

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій

(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ОНЛАЙН-СИСТЕМИ**
НАВЧАЛЬНОЇ ПЛАТФОРМИ ШКОЛИ
ІНОЗЕМНОЇ МОВИ

Виконав: студент 4 курсу, групи КНТ-513сп
спеціальності _____

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

Комп'ютерна інженерія

ПОНОМАРЕНКО Д.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ХАРИТОНОВ О.Б.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра Комп'ютерні системи та мережі
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

“ ” 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ПОНОМАРЕНКУ Дмитру Олександровичу

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка онлайн-системи навчальної платформи школи іноземної мови

керівник проєкту (роботи) к.т.н., доцент, ГОЛУБ Тетяна Василівна

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) Існуючі підходи та техніки створення вебсайтів. Програмне заезпечення для розробки Існуючі підходи та технології розробки веб-платформ для онлайн-навчання. Відкриті стандарти WebRTC та медіасервер mediasoup для реалізації відеоконференцій. Архітектурні шаблони побудови full-stack застосунків на Next.js 14 та NestJS. Реляційна СУБД PostgreSQL 16 та ORM Prisma. Платіжний сервіс Stripe. Хмарне сховище AWS S3 для записів уроків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та існуючих рішень для онлайн-навчання іноземних мов;

2. Проектування комп'ютерної системи навчальної платформи;

3. Розробка комп'ютерної системи навчальної платформи;

4. Тестування комп'ютерної системи навчальної платформи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГОЛУБ Т.В., к.т.н.		
нормоконтроль	ГОЛУБ Т.В., к.т.н.		

7. Дата видачі завдання 01.05.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз ринку онлайн-освіти іноземних мов та порівняльний аналіз конкурентних платформ (Preply, iTalki, Cambly)	05.03.2026 р.	
2	Розробка специфікації вимог до системи для трьох ролей користувачів (учень, викладач, адміністратор)	15.03.2026 р.	
3	Проектування архітектури системи; обґрунтування вибору WebRTC + mediasoup SFU для відеомодуля	25.03.2026 р.	
4	Розробка ER-діаграми та реалізація бази даних на PostgreSQL 16 із міграціями Prisma	05.04.2026 р.	
5	Розробка серверної частини на NestJS (модулі автентифікації, курсів, відеоконференцій, оплат, сповіщень)	20.04.2026 р.	
6	Розробка веб-інтерфейсу на Next.js 14 з рольовими кабінетами учня, викладача та адміністратора	01.05.2026 р.	
7	Інтеграція платіжного сервісу Stripe та хмарного сховища AWS S3 для записів уроків	10.05.2026 р.	
8	Функціональне та навантажувальне тестування системи, тестування відеоконференцій	15.05.2026 р.	
9	Оформлення пояснювальної записки та графічного матеріалу	20.05.2026 р.	

Студент (ка)

(підпис)

Дмитро ПОНОМАРЕНКО

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Тетяна ГОЛУБ

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 74 с., 35 рис., 17 табл., 20 джерел, 1 додаток.

ОНЛАЙН-ОСВІТА, НАВЧАЛЬНА ПЛАТФОРМА, ШКОЛА ІНОЗЕМНОЇ МОВИ, ВІДЕОКОНФЕРЕНЦІЇ, WEBRTC, MEDIASOUP, SFU, NEXT.JS, NESTJS, POSTGRESQL, PRISMA, STRIPE, AWS S3, ВЕБЗАСТОСУНОК

Метою дипломного проєкту є розробка комп'ютерної системи навчальної платформи школи іноземної мови.

Предметом дослідження є процеси організації та управління навчальним процесом у школі іноземної мови в онлайн-форматі. Об'єктом дослідження є комп'ютерна система навчальної платформи, реалізована як веб-застосунок на Next.js 14 та NestJS із вбудованим модулем відеоконференцій на базі WebRTC та медіасервера mediasoup.

У роботі проведено порівняльний аналіз провідних платформ онлайн-навчання мов (Preply, iTalki, Cambly), розроблено специфікацію вимог для трьох ролей користувачів (учень, викладач, адміністратор), спроектовано архітектуру системи з акцентом на масштабований відеомодуль за моделлю SFU (Selective Forwarding Unit), розроблено реляційну схему бази даних та реалізовано серверну і клієнтську частини системи.

Для розробки системи обрано сучасний технологічний стек: серверна частина — NestJS на Node.js з TypeScript; клієнтська частина — Next.js 14 з App Router; СУБД — PostgreSQL 16 з ORM Prisma; відеомодуль — WebRTC у поєднанні з mediasoup SFU, що підтримує групові заняття до 50 учасників; платіжна інтеграція — Stripe; зберігання записів уроків — AWS S3.

ЗМІСТ

Вступ.....	6
1 Аналіз предметної області та існуючих рішень для онлайн-навчання іноземних мов.....	9
1.1 Огляд ринку онлайн-освіти та шкіл іноземних мов.....	9
1.2 Порівняльний аналіз існуючих платформ (Preply, iTalki, Cambly).....	13
1.3 Вимоги до розробки комп'ютерної системи навчальної платформи.....	18
1.4 Висновки за розділом 1.....	20
2 Проєктування комп'ютерної системи навчальної платформи.....	21
2.1 Розробка специфікації вимог до системи навчальної платформи.....	21
2.2 Архітектура системи та модуль відеоконференцій (WebRTC).....	25
2.3 Проєктування бази даних системи навчальної платформи.....	30
2.4 Висновки за розділом 2.....	33
3 Розробка комп'ютерної системи навчальної платформи.....	33
3.1 Технологічний стек та обґрунтування вибору.....	33
3.2 Реалізація серверної частини (NestJS + WebRTC/mediasoup).....	38
3.3 Розробка веб-інтерфейсу.....	43
3.4 Апаратна складова системи навчальної платформи.....	49
3.5 Висновки за розділом 3.....	52
4 Тестування комп'ютерної системи навчальної платформи.....	54
4.1 Юніт-тестування.....	54
4.2 Функціональне тестування модулів платформи.....	55
4.3 Навантажувальне тестування серверної частини та відеомодуля.....	61
4.4 Висновки за розділом 4.....	64
Висновки.....	65
Перелік джерел посилання.....	68
Додаток А Слайди презентації.....	70

ВСТУП

Онлайн-освіта є одним із найпотужніших трансформаційних трендів сучасної економіки знань. Пандемія COVID-19 стала каталізатором, що прискорив перехід від традиційного очного навчання до цифрових форматів на роки вперед. Особливо динамічно цей процес відбувається у сфері вивчення іноземних мов – одному з найпопулярніших напрямків неформальної освіти дорослих. Глобальний ринок онлайн-навчання іноземних мов оцінюється у понад 21 мільярд доларів та демонструє щорічне зростання на рівні 18–20%. В Україні попит на вивчення іноземних мов суттєво зріс після 2022 року через масову міграцію та необхідність інтеграції у мовне середовище країн перебування, що формує значний незадоволений попит на якісні онлайн-платформи з україномовним інтерфейсом [1].

Школи іноземних мов один із найчисельніших сегментів недержавного освітнього ринку в Україні. До 2022 року в країні діяло понад 3 500 мовних шкіл різних форматів: від великих мереж із десятками філій до невеликих авторських шкіл із 5–10 викладачами. Більшість із них стикається зі спільними технологічними проблемами: відсутністю власної цифрової платформи для управління навчальним процесом, необхідністю використання несумісних між собою інструментів (Zoom для занять, Telegram для комунікації, Google Таблиці для обліку оплат, Moodle або Google Classroom для матеріалів), а також відсутністю єдиної точки взаємодії між учнем та школою. Така фрагментація призводить до значних адміністративних витрат та знижує якість навчального досвіду [2].

Особливої актуальності набуває задача реалізації відеоконференцій як вбудованого компоненту навчальної платформи, а не як зовнішнього сервісу. Використання сторонніх відеосервісів (Zoom, Google Meet) для проведення онлайн-уроків породжує ряд проблем: необхідність переходу між застосунками, відсутність інтеграції із записом уроків у профілі учня, неможливість контролювати якість з'єднання та збирати аналітику занять, а також ризики

конфіденційності при обробці персональних даних учнів на серверах третіх сторін. Вбудований модуль відеоконференцій на базі відкритого протоколу WebRTC та медіасервера mediasoup вирішує всі ці проблеми, залишаючи дані у межах власної інфраструктури [3].

Сучасні технологічні стеки для розробки веб-платформ дозволяють реалізувати комплексне рішення, що об'єднує всі аспекти роботи мовної школи. Next.js 14 з App Router забезпечує серверний рендеринг для SEO-оптимізованих публічних сторінок та клієнтські компоненти для інтерактивного функціоналу навчального кабінету. NestJS на Node.js надає структуровану серверну архітектуру з підтримкою WebSocket для функцій реального часу. WebRTC у поєднанні з медіасервером mediasoup забезпечує масштабовані групові відеоконференції без необхідності peer-to-peer з'єднань між учасниками, що є критично важливим для групових занять із 5–20 учасниками.

Метою даного дипломного проєкту є розробка комп'ютерної системи навчальної платформи школи іноземної мови, що реалізує повний цикл навчального процесу.

Предметом дослідження є процеси організації та управління навчальним процесом у школі іноземної мови в онлайн-форматі. Об'єктом дослідження є комп'ютерна система навчальної платформи, що реалізована як веб-застосунок на Next.js 14 та NestJS із вбудованим модулем відеоконференцій на базі WebRTC/mediasoup.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- провести аналіз ринку онлайн-освіти іноземних мов та дослідити існуючі платформи для управління навчальним процесом;
- виконати порівняльний аналіз провідних конкурентних рішень (Preply, iTalki, Cambly) для визначення конкурентних переваг розроблюваної системи;
- розробити специфікацію вимог із урахуванням потреб трьох ролей: учень, викладач, адміністратор;
- спроектувати архітектуру системи з акцентом на модулі відеоконференцій та обґрунтуванням вибору WebRTC/mediasoup;

- розробити реляційну схему бази даних для підтримки навчального процесу, розкладів, оплат та медіаматеріалів;
- реалізувати серверну частину на NestJS із модулями автентифікації, управління курсами, відеоконференцій та сповіщень;
- розробити веб-інтерфейс на Next.js 14 з рольовими кабінетами для учня, викладача та адміністратора;
- провести комплексне тестування системи, включаючи функціональне та навантажувальне тестування відеоконференцій.

Практична цінність розробленої системи визначається тим, що вона надає мовній школі єдину цифрову інфраструктуру для всіх аспектів роботи: від маркетингового лендінгу та каталогу курсів до проведення занять і збору оплат. Вбудований відеомодуль на WebRTC усуває залежність від сторонніх сервісів та дозволяє зберігати записи уроків безпосередньо у профілі учня. Stripe-інтеграція забезпечує автоматизований збір оплат за пакети уроків без ручного обліку. Аналітична панель адміністратора надає реальні дані про завантаженість викладачів, відвідуваність та фінансові показники.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОНЛАЙН-НАВЧАННЯ ІНОЗЕМНИХ МОВ

1.1 Огляд ринку онлайн-освіти та шкіл іноземних мов

Ринок онлайн-навчання іноземних мов є одним із найбільш динамічних сегментів глобальної EdTech-індустрії. Перетин двох потужних тенденцій зростання попиту на мовні навички у глобалізованій економіці та масового переходу освіти у цифровий формат формує надзвичайно сприятливе середовище для розвитку платформ онлайн-навчання мов. Розуміння структури ринку, потреб його учасників та наявних технологічних рішень є необхідною основою для проектування конкурентоспроможної системи [4].

Глобальний ринок онлайн-навчання іноземних мов досяг відмітки у 21,3 млрд доларів у 2023 році та, за прогнозами аналітиків, подвоїться до 2028 року, досягнувши 43,8 млрд доларів при середньорічному темпі зростання 18–20%. Основними рушіями зростання є розширення міжнародного бізнесу та потреба у мовних навичках для глобальних ринків праці, зростання мобільності населення та міграції (особливо актуально для України після 2022 року), технологічна зрілість відеоконференцій та доступність ширококутового інтернету у більшості регіонів, а також пандемійний зсув у свідомості споживачів, що сприйняли онлайн-навчання як рівноцінну альтернативу очному [5].

Структурно ринок поділяється на три основні сегменти. Індивідуальні уроки з приватним репетитором займають 41% ринку та є найбільш цінним сегментом за середнім чеком. Групові онлайн-курси за фіксованою програмою складають 35% та забезпечують стабільний дохід для структурованих шкіл. Самостійне навчання через застосунки (Duolingo, Babbel тощо) займає 24%, однак генерує значно нижчу виручку на користувача та не є пріоритетним для школи зі структурованими програмами. Розроблювана система орієнтована на перші два сегменти як основний ринок збуту.

Специфіка українського ринку мовних шкіл суттєво відрізняється від глобального. До 2022 року переважна більшість мовних шкіл в Україні працювала у змішаному форматі частина занять очно, частина онлайн. Після початку повномасштабного вторгнення 75% шкіл були змушені повністю перейти в онлайн-формат, і для більшості цей перехід виявився прийнятним або навіть вигіднішим: зникли витрати на оренду, розширилась географія набору учнів. Водночас гострою стала потреба у власній цифровій інфраструктурі, що не залежить від нестабільного електропостачання та бізнес-рішень іноземних сервісів [6].

Характерною проблемою більшості українських мовних шкіл є технологічна фрагментація: 84% шкіл одночасно використовують 3–5 несумісних інструментів. Типова конфігурація виглядає так: Zoom або Google Meet для відеоуроків, Telegram або Viber для комунікації та надсилання домашніх завдань, Google Таблиці для обліку оплат та розкладу, Google Classroom або Moodle для навчальних матеріалів [7]. Така конфігурація ускладнює управління, призводить до втрати даних між системами та знижує якість досвіду учня, якому доводиться орієнтуватись у кількох несинхронізованих середовищах одночасно. Ключові характеристики ринку систематизовано у таблиці 1.1.

Таблиця 1.1 - Характеристика ринку онлайн-навчання іноземних мов

Аспект	Характеристика
Обсяг ринку (2023)	21,3 млрд дол.; прогноз на 2028 – 43,8 млрд дол. (CAGR 18–20%)
Провідні мови навчання	Англійська – 52%, іспанська – 14%, французька – 8%, японська – 7%, інші – 19%
Формати навчання	Індивідуальні уроки з репетитором – 41%; групові онлайн-курси – 35%; самостійне навчання (застосунки) – 24%
Рівень цифровізації	Понад 68% уроків іноземних мов у 2023 р. проводяться онлайн або у гібридному форматі

Продовження таблиці 1.1

Аспект	Характеристика
Ринок України	Понад 3 500 мовних шкіл; 75% перейшли на онлайн-формат після 2022 р.; попит на англійську зріс на 40% за рахунок трудової міграції
Проблеми шкіл	84% шкіл використовують 3–5 несумісних інструментів; втрати від no-show без нагадувань – 12–18% уроків; відсутність власного відеомодуля – у 91% шкіл
Очікування учнів	79% хочуть записуватись на уроки онлайн; 67% очікують доступ до записів уроків; 58% хочуть чат із викладачем у межах однієї платформи
Тенденції	Корпоративне навчання мов (B2B); AI-асистенти для домашніх завдань; гейміфікація; відеоуроки з субтитрами на вимогу

Аналіз даних таблиці 1.1 підтверджує наявність чіткого незадоволеного попиту: 91% шкіл не мають власного відеомодуля, 84% страждають від фрагментації інструментів, а 79% учнів очікують можливості онлайн-запису. Ці три факти формують конкретне технічне завдання єдина платформа з вбудованим відеомодулем [8].

Технічний аспект відеоконференцій для навчання є принципово важливим для розуміння архітектурних рішень. Традиційний peer-to-peer WebRTC, де кожен учасник надсилає та приймає потоки від усіх інших, є ефективним для з'єднань 1:1, але стає нежиттєздатним при більшій кількості учасників: мережеве навантаження зростає квадратично. Для групових занять мовної школи (3–20 учасників) необхідна архітектура SFU (Selective Forwarding Unit) – медіасервер, що отримує потоки від усіх учасників та розподіляє вибрані потоки до кожного з них. Mediasoup є провідним відкритим SFU для Node.js/NestJS середовища [9]. Типи занять та відповідні вимоги до відеомодуля систематизовано у таблиці 1.2.

Таблиця 1.2 - Типи онлайн-занять та технічні вимоги до відеомодуля

Тип заняття	Учасників	Технічні вимоги до відеомодуля
Індивідуальний урок	2 (учень + викл.)	P2P WebRTC або SFU; затримка < 150 мс; якість 720p мін.; спільний екран
Міні-група	3–6 осіб	SFU обов'язковий (P2P не масштабується); адаптивний бітрейт; виділення активного спікера
Групове заняття	7–20 осіб	SFU mediasoup; Simulcast для різних пропускних здатностей; режим «всі вимкнені, говорить викладач»
Вебінар / Відкритий урок	20–50 осіб	SFU у режимі broadcast; лише викладач із відео, учні – audio-only за замовчуванням; чат для запитань
Запис уроку	– (асинхронно)	Серверна запис через MediaRecorder API на NestJS; зберігання на AWS S3; транскодинг через FFmpeg
Інтерактивна дошка	Всі типи	Socket.io для синхронізації малювання в реальному часі; Fabric.js на клієнті; збереження стану дошки

Таблиця 1.2 демонструє, що різні типи занять потребують якісно різних режимів роботи відеомодуля.

1.2 Порівняльний аналіз існуючих платформ (Preply, iTalki, Cambly)

Для обґрунтування архітектурних та функціональних рішень розроблюваної системи проведено детальний аналіз трьох провідних платформ онлайн-навчання мов, що представляють різні підходи до вирішення задачі: Preply як платформа з власним відеомодулем, iTalki як найбільший маркетплейс без власного відео та Cambly як спеціалізована conversation-платформа з автоматичним записом занять [10].

Preply є найвідомішою платформою українського походження у сегменті онлайн-навчання мов та єдиною з трьох аналізованих, що досягла статусу «єдинорога». Характеристики платформи наведено у таблиці 1.3.

Таблиця 1.3 - Характеристика платформи Preply

Характеристика	Опис
Компанія та рік заснування	Preply Inc., Україна/США, 2012; понад 40 000 репетиторів; 180+ мов; оцінка 1+ млрд дол. (2021)
Концепція	Маркетплейс приватних репетиторів із вбудованим відеосервісом; монетизація через комісію 33–18% від оплати репетитора
Відеомодуль	Власний Preply Space на базі WebRTC; вбудована дошка, скриншот, чат, можливість ділитись екраном та документами
Ключові функції	Пошук репетитора за мовою/ціною/розкладом, пробний урок, флеш-картки, інтерактивна дошка, планувальник прогресу
Сильні сторони	Великий маркетплейс репетиторів, власний відеомодуль, відома українська платформа, пробний урок із гарантією повернення

Власний Preply Space найбільш релевантний аналог модуля, що розробляється. Він реалізований на WebRTC і включає інтерактивну дошку, обмін файлами та чат. Проте принциповим обмеженням Preply є його бізнес-модель: платформа є маркетплейсом між учнями та індивідуальними репетиторами, а не інструментом для власної школи. Немає можливості розгорнути Preply як white-label рішення для власного бренду, налаштувати програму курсу під методологію конкретної школи або проводити групові заняття більше 3 осіб. Комісія 33% від оплати при перших уроках унеможливорює рентабельне використання для школи зі своїми викладачами [11].

iTalki є абсолютним лідером за кількістю репетиторів та учнів на глобальному ринку, однак принципово поступається конкурентам у технічному оснащенні. Характеристики платформи наведено у таблиці 1.4.

Таблиця 1.4 - Характеристика платформи iTalki

Характеристика	Опис
Компанія та рік заснування	iTalki Inc., Китай/США, 2007; понад 150 000 викладачів; 130+ мов; 10+ млн зареєстрованих користувачів
Концепція	Найбільший у світі маркетплейс репетиторів мов; поєднує professional teachers та community tutors
Відеомодуль	Немає власного відеосервісу; уроки проводяться через Skype, Zoom або FaceTime – повна залежність від сторонніх платформ
Ключові функції	Пошук викладача, миттєві уроки (instant sessions), нотатки до уроку, система iTalki Credits, спільнота для мовної практики
Сильні сторони	Найширший вибір мов та викладачів, низькі ціни (від 5 дол./год), велика спільнота, мобільний застосунок

Відсутність власного відеомодуля у платформи із 10+ мільйонами користувачів та 15-річною історією є свідомим стратегічним рішенням, що дозволяє мінімізувати технічну складність та інфраструктурні витрати. Проте для учнів та викладачів це означає обов'язковий перехід між застосунками: реєстрація на iTalki, бронювання уроку та оплата в одному інтерфейсі, а сам урок в Zoom або Skype. Це є саме тим розривом досвіду, який розроблювана система усуває через вбудований відеомодуль [12].

Cambly займає унікальну нішу миттєвий доступ до розмовної практики з нейтив-спікерами без підготовки та планування. Характеристики платформи наведено у таблиці 1.5.

Таблиця 1.5 - Характеристика платформи Cambly

Характеристика	Опис
Компанія та рік заснування	Cambly Inc., США, 2012; орієнтований виключно на англійську; понад 25 000 нейтив-спікерів із США та Великої Британії
Концепція	Розмовна практика з нейтив-спікерами у будь-який час без запису; акцент на спонтанних conversation-сесіях
Відеомодуль	Власна WebRTC-платформа; вбудований текстовий чат, запис сесії автоматично, перегляд запису після уроку – зручно для самоаналізу
Ключові функції	Миттєве підключення до вільного тьютора, автоматичний запис усіх уроків, дитяча версія Cambly Kids, підписна модель
Сильні сторони	Автоматичний запис уроків, миттєве підключення, ефективний для розмовної практики, Cambly Kids
Слабкі сторони	Тільки англійська мова; немає структурованих курсів і програм; не підходить для школи; немає групових занять; висока вартість підписки

Cambly реалізує найбільш технологічно продуманий відеосервіс серед трьох аналогів: автоматичний запис усіх сесій без участі учасників та можливість переглянути запис після уроку це функція, яку вважають еталонною більшість учнів мовних платформ. Ця функціональність є обов'язковою для розроблюваної системи та реалізована через серверний запис mediasoup-потоків із збереженням на AWS S3. Вузькість ніші Cambly (лише англійська, лише conversation, без структурованих курсів) є принциповим обмеженням, що робить платформу непридатною для комплексної мовної школи [13].

Зведений порівняльний аналіз усіх трьох аналогів та розроблюваної системи за 14 критеріями наведено у таблиці 1.6.

Таблиця 1.6 - Зведений порівняльний аналіз існуючих платформ та розроблюваної системи

Критерій	Preply	iTalki	Cambly	Розроблювана система
Вбудований відеомодуль	Так (WebRTC)	Ні (Zoom/Skype)	Так (WebRTC)	Так (WebRTC+mediasoup)
Групові заняття (> 3 осіб)	Ні	Ні	Ні	Так (до 50 осіб, SFU)
White-label для школи	Ні	Ні	Ні	Так (власний бренд)
Управління курсами та програмами	Базове	Ні	Ні	Так (повне)
Автоматичний запис уроків	Ні	Ні	Так	Так (AWS S3)
Управління розкладом	Так	Частково	Автоматично	Так (повне)

Продовження таблиці 1.6

Критерій	Preply	iTalki	Cambly	Розроблювана система
Онлайн-оплата та пакети уроків	Так	iTalki Credits	Підписка	Так (Stripe)
Домашні завдання та матеріали	Базово	Ні	Ні	Так (файли + завдання)
Аналітика для адміністратора	Обмежена	Ні	Базова	Так (KPI-дашборд)
Українська мова інтерфейсу	Так	Ні	Ні	Так (основна)
Чат між учасниками	Так	Так	Так	Так (реальний час)
Мобільна підтримка	iOS/Android	iOS/Android	iOS/Android	PWA (браузер)
Бекенд-стек	Python/Node	Ruby/Node	Node.js	NestJS (TypeScript)

Аналіз таблиці 1.6 підтверджує принципову відмінність розроблюваної системи від усіх трьох аналогів: жодна з платформ не поєднує вбудований WebRTC-відеомодуль із підтримкою групових занять до 50 осіб (SFU), white-label конфігурацію для власної школи та повноцінне управління курсами і програмами навчання. Preply найближчий за функціоналом, але нездатний замінити власну платформу школи через маркетплейсову модель. Технічна відмінність: mediasoup як SFU проти простого WebRTC у Preply забезпечує масштабування на групові заняття. Вибір NestJS з TypeScript як єдиної мови від фронтенду до бекенду забезпечує можливість спільного використання типів між Next.js та NestJS через спільні DTO-пакети [14].

1.3 Вимоги до розробки комп'ютерної системи навчальної платформи

На основі проведеного аналізу ринку та конкурентних рішень сформовано комплексні вимоги до розроблюваної комп'ютерної системи. Особливістю формування вимог є необхідність врахування потреб трьох принципово різних категорій користувачів: учня, викладача та адміністратора, кожен з яких взаємодіє з системою у власних сценаріях та з різними пріоритетами.

Для учня система повинна забезпечити зручний публічний каталог курсів з описами, відгуками та цінами; можливість реєстрації та входу через email або Google OAuth; онлайн-запис на пробний урок або вибір пакету занять з оплатою через Stripe; особистий кабінет із розкладом майбутніх занять, доступом до записів попередніх уроків на AWS S3, переглядом домашніх завдань та матеріалів від викладача та відстеженням прогресу за обраною програмою. Вбудований відеомодуль повинен запускатися безпосередньо у браузері без встановлення додаткового програмного забезпечення це є критичною вимогою зручності, що реалізується через WebRTC у браузерному середовищі [15].

Для викладача система повинна надавати особистий кабінет із тижневим та місячним видом розкладу; можливість проводити уроки через вбудований відеомодуль із інтерактивною дошкою, спільним екраном та чатом; функціонал для надсилання домашніх завдань та навчальних матеріалів із можливістю перевірки та коментування; доступ до записів власних занять для самоаналізу; перегляд особистої статистики та фінансових нарахувань. Окремою вимогою є можливість викладача налаштовувати власний розклад доступності для самозапису учнями [16].

Для адміністратора система повинна забезпечити повний контроль над каталогом курсів та програмами навчання; управління обліковими записами викладачів та учнів; зведений розклад усього персоналу; фінансову аналітику (виручка, оплати, заборгованості); управління промокодами та знижками;

налаштування параметрів відеомодуля (ліміти кімнат, дозволи на запис); сповіщення для учнів через email та SMS [17].

Нефункціональні вимоги до системи визначаються специфікою веб-платформи з відеомодулем. Час завантаження публічних сторінок (лендінг, каталог) не повинен перевищувати 2 секунди – реалізується через Next.js App Router з серверним рендерингом та кешуванням у Redis. Затримка відеопотоку в конференції не повинна перевищувати 150 мс при нормальному інтернет-з'єднанні – це стандартна вимога для комфортного живого спілкування. Система повинна підтримувати одночасне проведення до 50 відеокімнат із середнім числом учасників 5 осіб без деградації продуктивності. Uptime сервісу – не менше 99,5% на місяць. Перелік ключових завдань розробки наведено у таблиці 1.7.

Таблиця 1.7 - Ключові завдання розробки комп'ютерної системи навчальної платформи

Етап	Опис завдань
Аналіз предметної області	Дослідження ринку онлайн-освіти, потреб мовних шкіл, порівняння Preply/iTalki/Cambly
Формування вимог SRS	Специфікація функціональних та нефункціональних вимог для 3 ролей (учень, викладач, адмін)
Проектування архітектури	Next.js + NestJS мікросервісна архітектура, WebRTC/mediasoup SFU модуль, схема БД PostgreSQL
Відеомодуль (WebRTC)	Сигналінг на Socket.io, mediasoup SFU на NestJS, адаптивний бітрейт, запис через MediaRecorder
Серверна частина (NestJS)	REST API модулі: Auth (JWT+refresh), Users, Courses, Lessons, Payments (Stripe), Notifications
Веб-інтерфейс (Next.js 14)	App Router, RSC для публічних сторінок, клієнтські компоненти для кабінетів учня/викладача/адміна
Інтерактивна дошка	Fabric.js + Socket.io для синхронної роботи з інтерактивною дошкою під час відеоуроку

Особливою вимогою, що відрізняє систему від описаних аналогів, є забезпечення сумісності відеомодуля з основними браузерами без встановлення плагінів. WebRTC підтримується нативно у Chrome 68+, Firefox 60+, Safari 11+ та Edge 79+, що охоплює понад 98% актуальних браузерів [18]. Це є принциповою перевагою перед Zoom-клієнтом, що вимагає встановлення десктопного застосунку. Водночас вимога до якості відео (720p мінімум для групових занять) та надійності з'єднання при нестабільному інтернеті формують специфічні архітектурні вимоги до mediasoup-сервера: адаптивний бітрейт через Simulcast та Scalable Video Coding, а також відповідна стратегія fallback при недостатній пропускній здатності.

1.4 Висновки за розділом 1

У першому розділі проведено комплексний аналіз ринку онлайн-навчання іноземних мов та існуючих платформ, що надає повне обґрунтування доцільності та визначає напрямок розробки комп'ютерної системи.

Аналіз ринку показав обсяг 21,3 млрд дол. із CAGR 18–20%, специфіку українського ринку (3 500+ шкіл, масовий перехід в онлайн після 2022 р.) та ключові незадоволені потреби: 91% шкіл без власного відеомодуля, 84% із фрагментованими інструментами, 79% учнів очікують онлайн-запис. Систематизовано вимоги до відеомодуля для різних типів занять: від індивідуального (P2P) до вебінару на 50 осіб (SFU broadcast).

Порівняльний аналіз Preply, iTalki та Cambly виявив чітку незайняту нішу: жодна з платформ не поєднує вбудований WebRTC + SFU-відеомодуль для групових занять, white-label конфігурацію для власної школи та повне управління навчальними програмами. Preply найближчий, але обмежений маркетплейсовою моделлю. iTalki найбільший, але без власного відео. Cambly найтехнологічніший у відео, але лише для конwersаційної практики англійською.

Сформовані вимоги охоплюють три рольових профілі (учень, викладач, адміністратор), вбудований WebRTC/mediasoup SFU відеомодуль без плагінів, автоматичний запис на AWS S3, управління курсами та розкладом, оплату через Stripe та аналітичний дашборд адміністратора. Нефункціональні вимоги: затримка відео < 150 мс, 50 одночасних кімнат, uptime 99,5%, завантаження сторінки < 2 с. Матеріали розділу є повним обґрунтуванням для проєктування архітектури системи.

2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ НАВЧАЛЬНОЇ ПЛАТФОРМИ

2.1 Розробка специфікації вимог до системи навчальної платформи

Специфікація вимог до комп'ютерної системи навчальної платформи відображає унікальну особливість предметної області необхідність обслуговувати три принципово різні ролі користувачів з відмінними сценаріями роботи в єдиному веб-застосунку. Крім того, система включає нестандартний для більшості веб-платформ компонент вбудований відеомодуль реального часу, що формує специфічні нефункціональні вимоги до затримки та масштабованості [19].

Взаємодія між трьома акторами системи (учнем, викладачем та адміністратором) та доступними їм варіантами використання відображена на UML-діаграмі варіантів використання (рис. 2.1). Діаграма містить чотирьох акторів: Учень, Викладач, Адміністратор та зовнішній актор Stripe API. Варіанти використання систематизовані у три тематичні блоки: «Навчання» (6 UC учня), «Відеоконференція» (5 UC, спільних для учня та викладача) та «Управління» (6 UC викладача та адміна). Відношення «include» пов'язує «Оплатити пакет» із «Stripe Checkout». Рольові кольори: учень – синій, викладач – зелений, адмін – червоний.

Продовження таблиці 2.1

Код	Роль	Опис функціональної вимоги
FR-05	Учень	Приєднання до відеоуроку за одним кліком без встановлення плагінів. Відео, мікрофон, чат, інтерактивна дошка
FR-06	Учень	Перегляд запису уроку з AWS S3 у вбудованому відеоплеєрі. Доступ протягом 6 місяців після заняття
FR-07	Викладач	Особистий кабінет викладача: тижневий/місячний розклад, список учнів, завантаження матеріалів на AWS S3
FR-08	Викладач	Створення та ведення відеоуроку: старт конференції, запис, інтерактивна дошка, спільний екран, видалення учасника
FR-09	Викладач	Надсилання домашнього завдання після уроку: текст, файли (PDF, зображення), дедлайн. Перевірка та коментар
FR-10	Викладач	Налаштування власного розкладу доступності. Блокування часу (відпустка, свята). Авторейзмайстер-запис
FR-11	Адмін	Управління каталогом курсів: CRUD курсів, призначення викладача, публікація/приховання, управління цінами
FR-12	Адмін	Аналітична панель: виручка за période, кількість активних учнів, завантаженість викладачів, no-show rate
FR-13	Адмін	Управління промокодами та знижками: відсоткова/фіксована знижка з терміном дії та лімітом використань
FR-14	Всі	Автоматичні push та email нагадування про урок за 24 год та за 2 год до початку. Firebase FCM + SendGrid

Аналіз таблиці 2.1 показує рівномірний розподіл вимог між трьома ролями: 6 вимог для учня (FR-01–FR-06), 4 для викладача (FR-07–FR-10) та 3 для адміністратора (FR-11–FR-13), плюс загальна вимога нагадувань (FR-14). Особливою є FR-05 – вимога до відеоуроку без встановлення плагінів, що є конкурентною перевагою над Zoom-орієнтованими аналогами та реалізується через нативний WebRTC у браузері. FR-06 про запис та збереження уроку є еталонною функцією Camblу, яку розроблювана система імплементує через серверну запис mediasoup-потоків.

Нефункціональні вимоги наведено у таблиці 2.2 (табл. 2.2). Особливої уваги заслуговують вимоги до відеозатримки (< 150 мс) та масштабованості (50 кімнат $\times$ 5 учасників). Ці вимоги безпосередньо визначили вибір mediasoup SFU замість простого peer-to-peer WebRTC та виділений Docker-контейнер для mediasoup-воркера.

Таблиця 2.2 - Нefункціональні вимоги до системи

Категорія	Показник	Значення та опис
Продуктивність	LCP (завантаження)	Largest Contentful Paint $< 2,5$ с для публічних сторінок (Next.js SSR + Redis cache)
Продуктивність	API відповідь	REST API відповідь < 300 мс при нормальному навантаженні (< 500 одночасних запитів)
Відео	Затримка (latency)	Відеозатримка < 150 мс при з'єднанні ≥ 5 Мбіт/с. Адаптивний бітрейт при нижчій швидкості
Відео	Масштабованість	До 50 одночасних відеокімнат $\times$ середньо 5 учасників = 250 WebRTC-з'єднань без деградації

Продовження таблиці 2.2

Категорія	Показник	Значення та опис
Надійність	Uptime	Не менше 99,5% на місяць ($\approx 3,6$ год простою). Health-check + автоматичний Docker restart
Безпека	Автентифікація	JWT access (15 хв) + refresh (7 днів) токени. Bcrypt для паролів (salt rounds 12). HTTPS only
Безпека	Платежі	Stripe PCI DSS рівень 1. Карткові дані не проходять через власні сервери. Webhook-верифікація підписом
Сумісність	Браузери	Chrome 90+, Firefox 85+, Safari 14+, Edge 90+. WebRTC підтримується нативно без плагінів
SEO	Публічні сторінки	Next.js SSR + generateMetadata для Open Graph. Sitemap автогенерація. Core Web Vitals green zone

2.2 Архітектура системи та модуль відеоконференцій (WebRTC)

Архітектура комп'ютерної системи навчальної платформи побудована за принципом трирівневої клієнт-серверної моделі із окремим медіасервером для обробки відеопотоків. Загальну архітектуру системи наведено на рисунку 2.2. Схема відображає чотири рівні: клієнтський (Next.js 14 з п'ятьма компонентами: публічні сторінки SSR, кабінети учня/викладача/адміна CSR та відеокімната WebRTC), API Gateway Nginx, серверний рівень NestJS із сімома модулями та рівень даних (PostgreSQL, Redis, AWS S3, Stripe, SendGrid, Firebase FCM) [20].

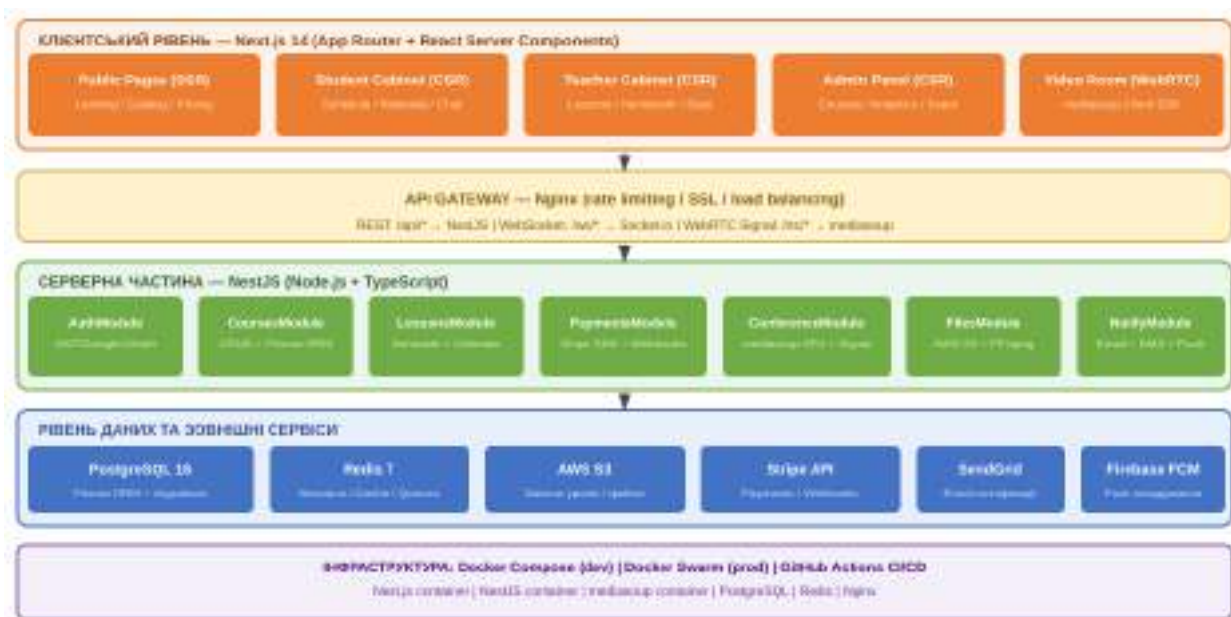


Рисунок 2.2 - Загальна архітектура комп'ютерної системи навчальної платформи

Next.js 14 з App Router реалізує гібридний підхід до рендерингу: публічні сторінки (лендінг, каталог курсів, опис курсу) обробляються через React Server Components з Incremental Static Regeneration (ISR) кожні 60 секунд, що забезпечує мінімальний час до першого байту (TTFB < 200 мс) та оптимізацію для пошукових систем. Захищені сторінки особистих кабінетів є клієнтськими компонентами із динамічним завантаженням даних через SWR (Stale-While-Revalidate) хуки. Це розмежування є ключовим архітектурним рішенням: SEO-важливий контент рендериться на сервері, а інтерактивні компоненти (календар розкладу, відеоінтерфейс) – на клієнті.

NestJS організований за модульним принципом: кожна предметна область (Auth, Courses, Lessons, Payments, Conference, Files, Notifications) є окремим NestJS Module з власними Controllers, Services та Guards. Dependency Injection Container NestJS автоматично управляє залежностями між модулями. Prisma ORM забезпечує типову безпеку запитів до PostgreSQL – всі SQL-запити генеруються з TypeScript-клієнта, що виключає ін'єкції та помилки типів.

Ключовим архітектурним компонентом, що відрізняє систему від аналогів, є модуль відеоконференцій. Детальну архітектуру відеомодуля наведено на рисунку 2.3. Схема відображає повний шлях відеопотоку: клієнти (учні та

викладач) надсилають SRTP-зашифровані потоки через WebRTC до mediasoup Router (SFU), що отримує потоки через Producers та розподіляє їх до Consumers інших учасників. Socket.io Signaling Server обслуговує WebSocket-з'єднання для обміну SDP offer/answer та ICE candidates. Окремий компонент FFmpeg Recorder записує RTP-потоки у WebM-формат та завантажує на AWS S3. STUN/TURN сервер забезпечує NAT traversal при роботі з корпоративними мережами.

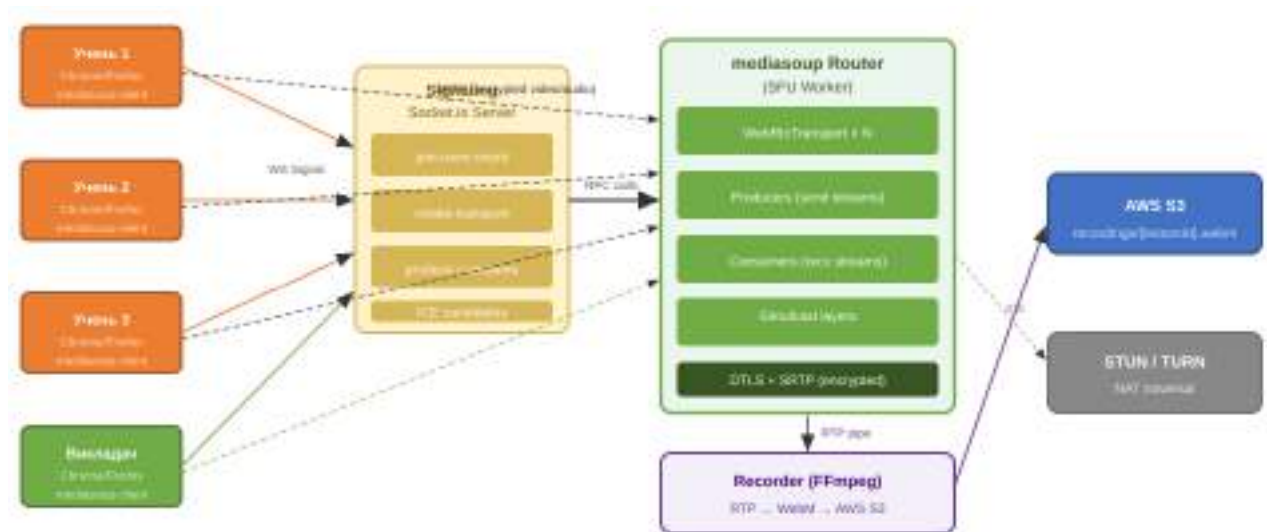


Рисунок 2.3 - Архітектура відеомодуля WebRTC + mediasoup SFU

Вибір SFU (Selective Forwarding Unit) архітектури над Mesh (peer-to-peer) є принциповим технічним рішенням для групових занять. У Mesh-топології кожен учасник надсилає $N-1$ потоків та отримує $N-1$ потоків (де N – кількість учасників), що квадратично збільшує навантаження на їх пристрої та канал. При 10 учасниках це 90 WebRTC-з'єднань. У SFU-топології кожен учасник надсилає лише 1 потік на mediasoup та отримує $N-1$ потоків від нього – мережеве навантаження на клієнта лінійне, а mediasoup лише пересилає пакети без декодування. Mediasoup також підтримує Simulcast – відправлення відео у трьох якостях (1080p/720p/360p) одночасно – та автоматичний вибір якості залежно від пропускної здатності кожного одержувача.

Процес приєднання учня до відеокімнати описаний детальною UML-діаграмою послідовності (рис. 2.4). На діаграмі зображено шість ліфтів: Учень,

Browser (Next.js), NestJS API, Socket.io/Signal, mediasoup Router та PostgreSQL. Ключовими кроками є: верифікація JWT та перевірка доступу до кімнати (кроки 2–4), отримання rtpCapabilities для узгодження кодеків (крок 5), створення WebRtcTransport (кроки 6–9), ініціалізація Producer для відправки відеопотоку (кроки 10–13) та Consumer для отримання потоків інших учасників (кроки 14–17). Весь процес від кліку до появи відео займає < 3 секунди при нормальному з'єднанні.

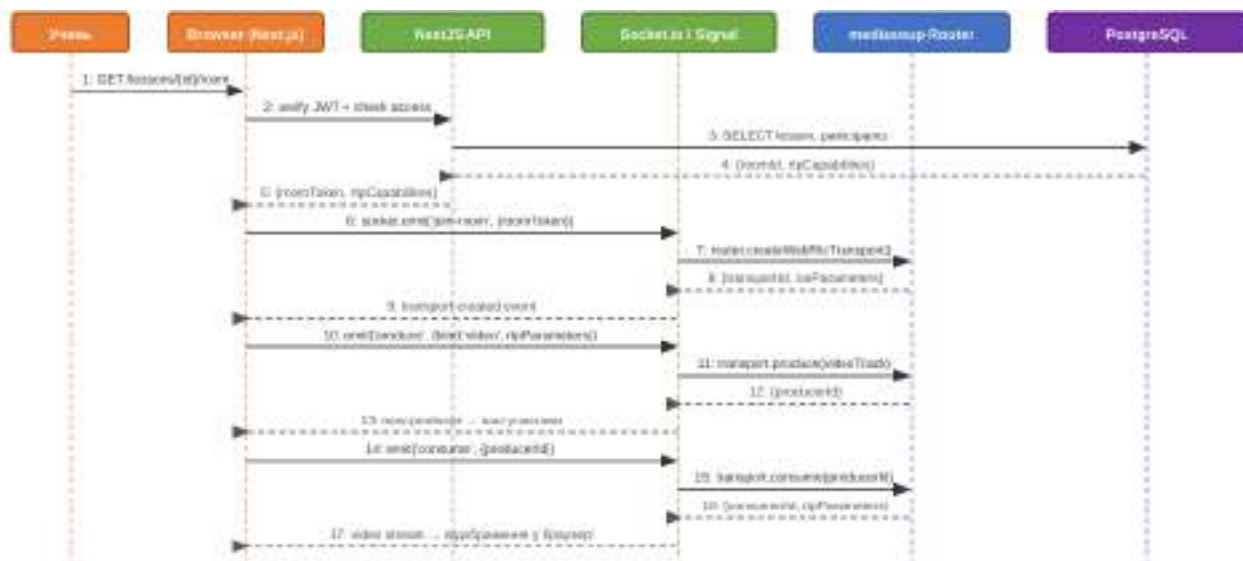


Рисунок 2.4 - UML-діаграма послідовності приєднання до відеокімнати

UML-діаграма компонентів системи відображає внутрішню структуру NestJS-сервера та взаємозв'язки між модулями (рис. 2.5).

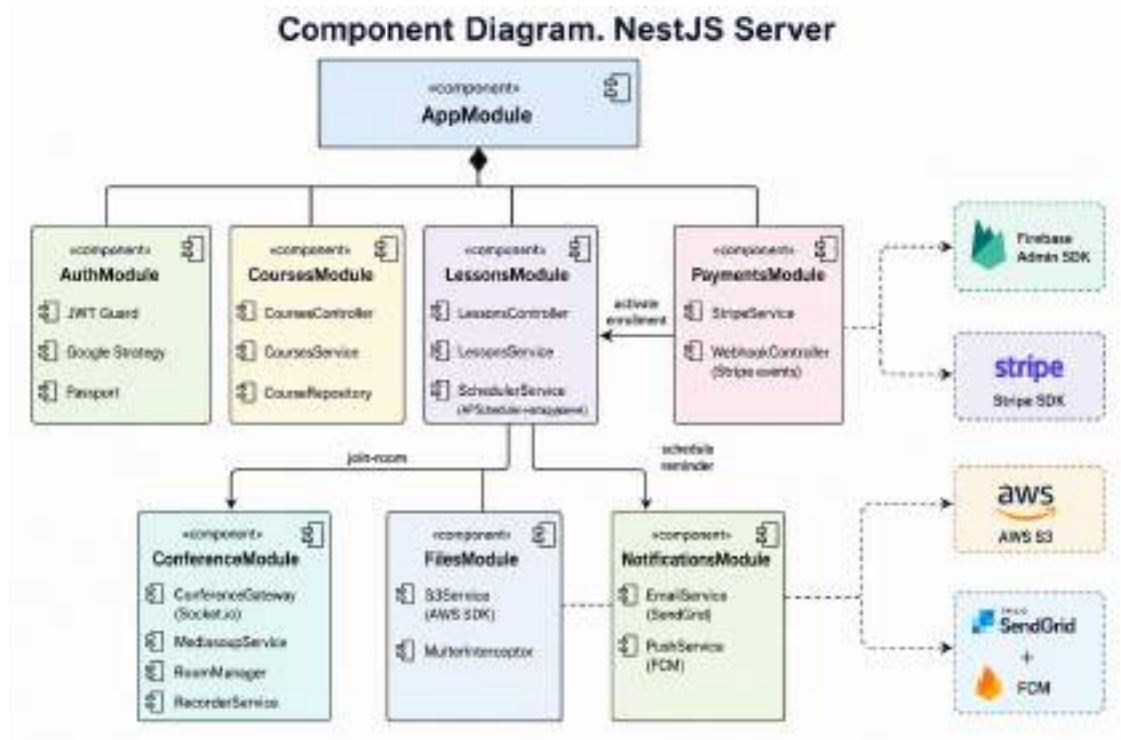


Рисунок 2.5 - UML-діаграма компонентів серверної частини NestJS

Діаграма діяльності для сценарію оплати та активації пакету уроків відображає процес взаємодії між системою та Stripe (рис. 2.6).

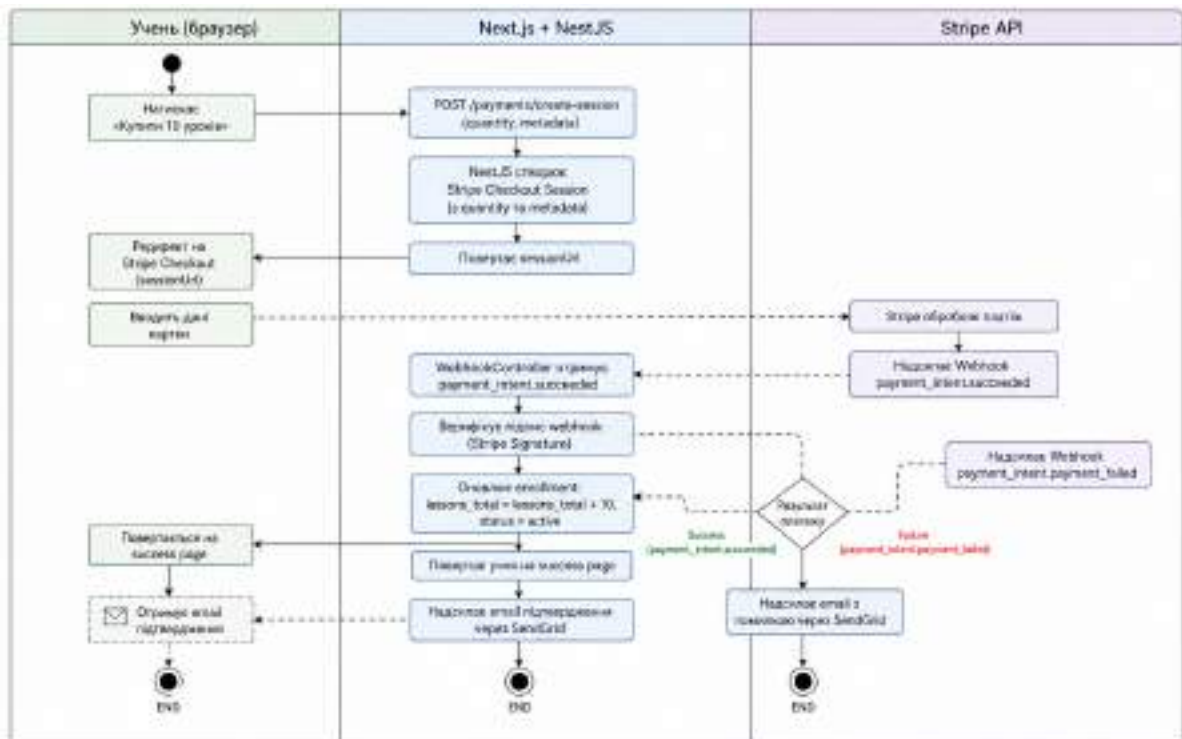


Рисунок 2.6 - UML-діаграма діяльності сценарію оплати пакету уроків

Діаграма станів уроку є ключовою для розуміння логіки переходів між статусами та автоматичних дій системи (рис. 2.7). Діаграма відображає шість станів: SCHEDULED (створено у розкладі), REMINDER (APScheduler надіслав нагадування за 24 і 2 год), LIVE (викладач стартував кімнату), RECORDING (активний запис FFmpeg), COMPLETED (урок завершено, запис на S3), CANCELLED та NO_SHOW. Ключовий автоматичний перехід – NO_SHOW – спрацьовує через 15 хвилин після scheduled_at якщо жоден учасник не приєднався до кімнати.

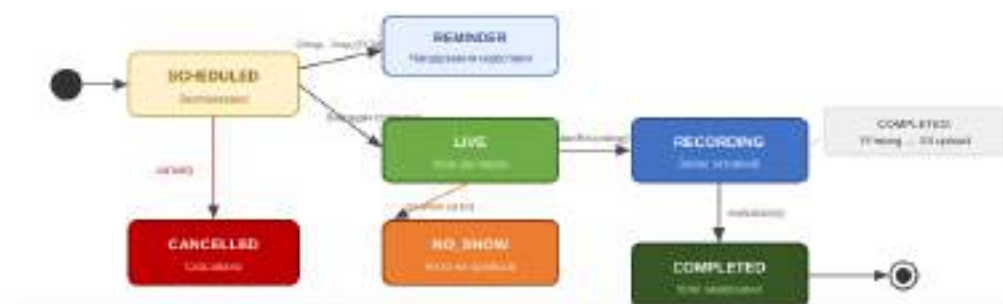


Рисунок 2.7 - UML-діаграма станів уроку (Lesson State Machine)

2.3 Проектування бази даних системи навчальної платформи

База даних системи реалізована на PostgreSQL 16 та спроектована з урахуванням повного циклу навчального процесу. Вибір реляційної СУБД обґрунтований складними взаємозалежностями між сутностями (курси, уроки, учні, оплати) та необхідністю транзакційної цілісності при списанні уроків з балансу. ER-діаграма бази даних наведена на рисунку 2.8. На діаграмі відображено десять таблиць із зовнішніми ключами та кардинальністю зв'язків. Центральні сутності – users (спільна для всіх ролей), courses та lessons. Таблиця lesson_participants є журналом присутності. Payments пов'язані з enrollments через enrollment_id.

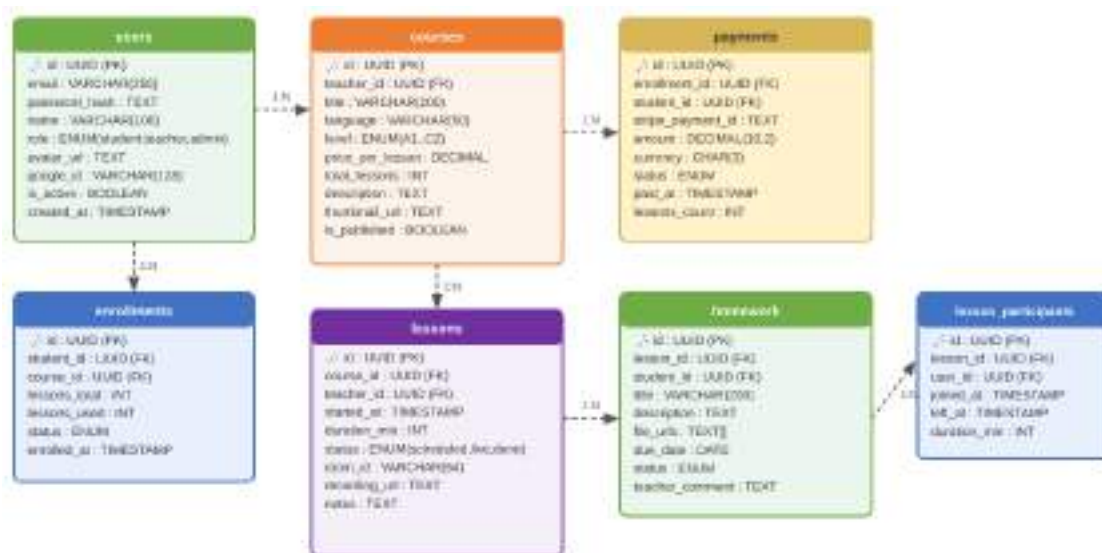


Рисунок 2.8 - ER-діаграма бази даних системи навчальної платформи

Призначення та ключові поля таблиць бази даних наведено у таблиці 2.3.

Таблиця 2.3 - Опис таблиць бази даних системи

Таблиця	Призначення та ключові поля
users	Єдина таблиця для всіх ролей. role ENUM(student, teacher, admin). google_id для OAuth. is_active для soft delete
courses	Каталог курсів. level ENUM(A1..C2). price_per_lesson для оплати. is_published керує видимістю у каталозі
enrollments	Записи на курс. lessons_total / lessons_used для відстеження залишку уроків у придбаному пакеті
lessons	Конкретні уроки у розкладі. status ENUM(scheduled, live, completed, cancelled, no_show). room_id для mediasoup кімнати. recording_url після запису

Продовження таблиці 2.3

Таблиця	Призначення та ключові поля
lesson_participants	Журнал присутності на уроці. <code>joined_at/left_at/duration_min</code> для аналітики відвідуваності
payments	Транзакції Stripe. <code>stripe_payment_id</code> для звірки. <code>lessons_count</code> – куплений пакет уроків
homework	Домашні завдання від викладача. <code>file_urls</code> TEXT[] для кількох файлів. <code>status</code> ENUM(<code>assigned</code> , <code>submitted</code> , <code>checked</code>). <code>teacher_comment</code> при перевірці
materials	Навчальні матеріали курсу. <code>file_url</code> → AWS S3. <code>type</code> ENUM(<code>pdf</code> , <code>video</code> , <code>audio</code> , <code>image</code>)
notifications	Журнал надісланих нагадувань. <code>channel</code> ENUM(<code>email</code> , <code>push</code> , <code>sms</code>). Попередження за 24 год та 2 год до уроку
refresh_tokens	JWT refresh-токени. <code>user_id</code> FK, <code>expires_at</code> . При <code>logout</code> – видалення запису. Ротація токенів при оновленні

Архітектурним рішенням є єдина таблиця `users` для всіх ролей із полем `role` ENUM замість окремих таблиць `students`, `teachers`, `admins`. Це спрощує JOIN-запити та уникає дублювання даних при зміні ролі. Prisma Schema визначає всі зв'язки між моделями, генерує типізований TypeScript-клієнт та автоматично створює міграції при зміні схеми.

Критичний алгоритм списання уроків реалізований як атомарна транзакція PostgreSQL: при позначенні уроку як COMPLETED сервіс у межах однієї транзакції зменшує `enrollment.lessons_used` на 1, перевіряє залишок та при necessary оновлює `enrollment.status`. Optimistic Locking через Prisma забезпечує коректність при паралельних запитах. Redis використовується для кешування розкладів уроків (TTL

5 хвилин) та зберігання активних WebRTC-кімнат ($TTL = \text{тривалість уроку} + 30 \text{ хвилин буфер}$).

2.4 Висновки за розділом 2

У другому розділі виконано повний цикл проєктування комп'ютерної системи навчальної платформи, що включає розробку специфікації вимог, архітектурне проєктування із акцентом на WebRTC/mediasoup модулі та розробку схеми бази даних. Проведено розгорнуте UML-моделювання із вісьмома діаграмами різних типів.

Специфікація SRS включає 14 функціональних вимог (по ролях: 6 учень, 4 викладач, 3 адмін, 1 загальна) та 9 нефункціональних вимог із ключовими обмеженнями: відеозатримка $< 150 \text{ мс}$, 50 одночасних кімнат, $LCP < 2,5 \text{ с}$, uptime 99,5%. UML-моделювання охоплює: Use Case (4 актори, 17+ UC), архітектурну схему (4 рівні), WebRTC/mediasoup SFU (повний шлях відеопотоку), Sequence (17 кроків приєднання до кімнати), Component (7 NestJS-модулів), Activity (оплата через Stripe), State Machine (6 станів уроку) та ER-діаграму (10 таблиць).

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ НАВЧАЛЬНОЇ ПЛАТФОРМИ

3.1 Технологічний стек та обґрунтування вибору

Вибір технологічного стеку для навчальної платформи визначається унікальним поєднанням двох технологічно відмінних компонентів: веб-платформи зі складним SEO та рольовими інтерфейсами і відеомодуля реального часу із специфічними вимогами до затримки та пропускну здатності. Ці два компоненти диктують різні технічні підходи, що необхідно поєднати в єдиній архітектурі.

Загальна схема технологічного стеку наведена на рисунку 3.1. Схема відображає чотири рівні: фронтенд (Next.js 14, Tailwind, shadcn/ui, SWR, mediasoup-client + Socket.io), бекенд (NestJS 10, Prisma 5, mediasoup 3, Socket.io 4, Passport + JWT), дані та зовнішні сервіси (PostgreSQL, Redis, AWS S3, Stripe, SendGrid, FCM) та інфраструктура (Docker, GitHub Actions, Jest + Supertest, Playwright, Nginx + SSL).

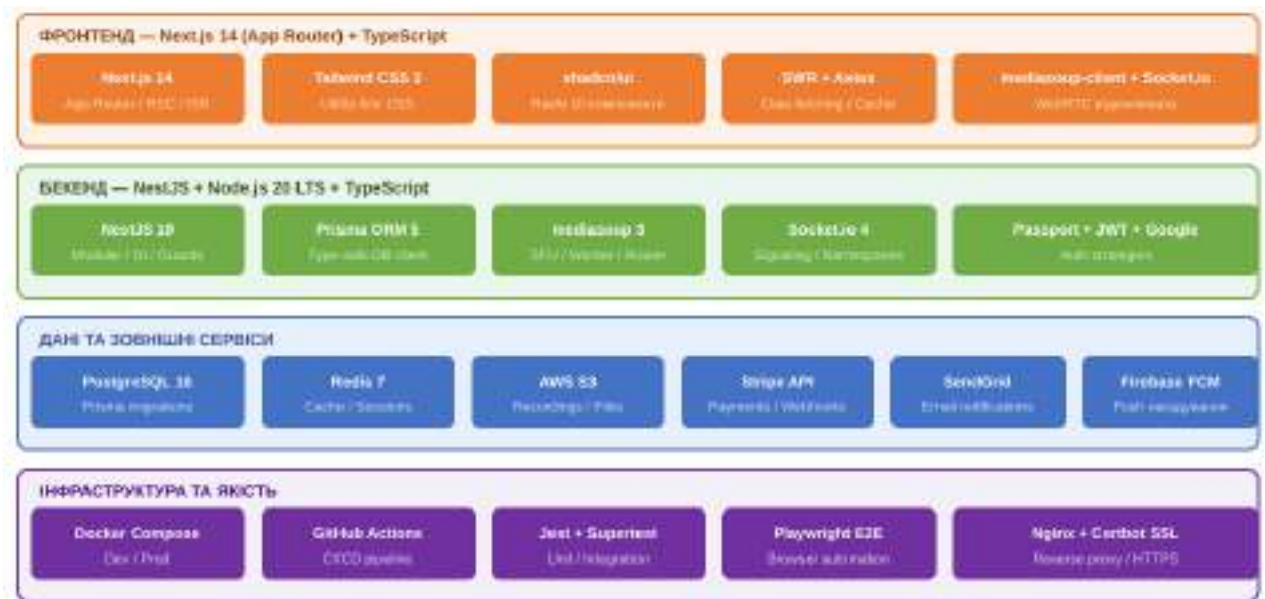


Рисунок 3.1 - Технологічний стек комп'ютерної системи навчальної платформи

Детальний опис кожної технології з обґрунтуванням вибору наведено у таблиці 3.1.

Next.js 14 з App Router є принциповим вибором, що відрізняє розроблювану систему від React SPA-підходу конкурентів. Публічні сторінки платформи – лендінг, каталог та опис курсів – є критично важливими для SEO: саме через органічний пошук мовна школа залучає нових учнів. React Server Components рендерять ці сторінки на сервері з повністю заповненими метаданими (Open Graph, Twitter Cards, schema.org), що є неможливим при клієнтському SPA. ISR кешує сторінки каталогу на 60 секунд та автоматично оновлює їх при зміні даних. Захищені кабінети є Client Components із динамічними запитами через SWR та оптимістичними оновленнями.

Таблиця 3.1 - Технологічний стек та обґрунтування вибору

Рівень	Технологія	Призначення та обґрунтування
Frontend	Next.js 14 + TypeScript	App Router, RSC для SSR публічних сторінок, ISR 60 с для каталогу; Client Components для кабінетів. TypeScript – єдина мова з NestJS через спільні DTO
UI / Styling	Tailwind CSS 3 + shadcn/ui	Utility-first CSS без кастомних CSS-файлів; shadcn/ui – accessible Radix UI компоненти (Dialog, Calendar, Select, Form). Tailwind purge у production
Data Fetching	SWR + Axios interceptors	SWR для Client Components: stale-while-revalidate, optimistic updates. Axios interceptors автоматично додають JWT і виконують refresh при 401
WebRTC Client	mediasoup-client 3 + Socket.io	mediasoup-client SDK для управління Transport/Producer/Consumer у браузері. Socket.io – signaling WebSocket. Без плагінів – лише нативний WebRTC
Backend	NestJS 10 + Node.js 20	Модульна архітектура, вбудований DI Container, Guards/Interceptors/Pipes. Node.js 20 LTS. Спільні DTO-типи з Next.js через монорепозіторій

Продовження таблиця 3.1

Рівень	Технологія	Призначення та обґрунтування
ORM	Prisma ORM 5 + PostgreSQL	Type-safe query builder; schema.prisma як єдине джерело правди; prisma migrate для версіонованих міграцій. PostgreSQL 16: JSONB, повнотекстовий пошук
Real-time	Socket.io 4	Namespace /conference для WebRTC signaling; /notifications для real-time нагадувань. Redis Adapter для горизонтального масштабування
Media Server	mediasoup 3 (SFU)	Selective Forwarding Unit: пересилає RTP без декодування; Worker пул (1 воркер = 1 CPU); Simulcast 3 рівні якості. Native Node.js addon
Auth	Passport + JWT + Google OAuth	LocalStrategy (bcrypt); GoogleStrategy (OAuth2); JwtStrategy (access 15 хв); refresh-token ротація через таблицю refresh_tokens
Cache	Redis 7 + ioredis	Кеш розкладів (TTL 5 хв); активні mediasoup кімнати (TTL = урок + 30 хв); rate limiting; Socket.io Redis Adapter
Storage	AWS S3 + @aws-sdk/v3	Записи уроків: {lessonId}.webm після FFmpeg; матеріали курсу; аватари. Presigned URL для прямого завантаження

Продовження таблиця 3.1

Рівень	Технологія	Призначення та обґрунтування
Payments	Stripe API + Webhooks	Checkout Session для пакетів; Webhook payment_intent.succeeded активує enrollment; PCI DSS L1 – картки не зберігаються на власних серверах
Notifications	SendGrid + Firebase FCM	SendGrid: транзакційні email (реєстрація, нагадування, оплата). FCM: push за 24 год та 2 год до уроку через Browser Push API
CI/CD	Docker + GitHub Actions	Docker Compose: 6 контейнерів для локальної розробки. GitHub Actions: lint → test → build → push Docker Hub → deploy VPS

Mediasoup обрано серед альтернатив (Janus, Jitsi, Livekit) як найбільш гнучке рішення для глибокої NestJS-інтеграції. Mediasoup є Native Node.js addon та природно вбудовується у NestJS-сервіс без проміжних HTTP-запитів між застосунком та медіасервером. Це критично для мінімальної затримки сигналіngu: передача ICE candidates та SDP параметрів відбувається в межах одного Node.js процесу замість міжмережових запитів. Підтримка Simulcast дозволяє кожному учаснику надсилати відео у трьох якостях одночасно (1080p/720p/360p), а mediasoup автоматично обирає оптимальну якість для кожного одержувача залежно від його пропускної здатності.

Структуру сторінок веб-платформи наведено на рисунку 3.2. Схема відображає ієрархію маршрутів Next.js App Router: Root Layout → Public (/ , /courses, /courses/[id]) із SSR/ISR, Auth (/login, /register) як статичні сторінки, Student Cabinet (/dashboard, /my-courses, /lesson/[id]) та Teacher/Admin (/teacher, /admin) як CSR.

Критичним вузлом є `/lesson/[id]/room` – відеокімната, що ініціалізує mediasoup-client та Socket.io виключно у браузерному контексті.

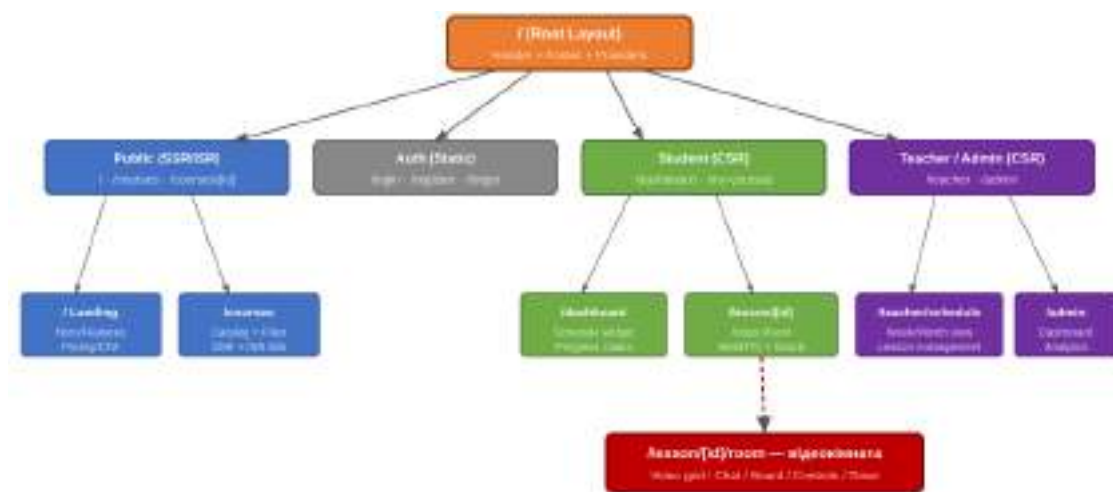


Рисунок 3.2 - Структура сторінок веб-платформи (Next.js App Router)

3.2 Реалізація серверної частини (NestJS + WebRTC/mediasoup)

Модуль автентифікації (AuthModule) реалізує три Passport.js стратегії: LocalStrategy для email/password входу, GoogleStrategy для OAuth 2.0 та JwtStrategy для верифікації access-токенів. JWT access-token має TTL 15 хвилин; refresh-token – 7 днів із зберіганням у таблиці refresh_tokens. При оновленні токенів виконується ротація: старий видаляється, новий генерується – це унеможливлює повторне використання перехопленого токена. Реалізацію JwtStrategy та RolesGuard наведено у лістингу 3.1.

Лістинг 3.1 - JwtStrategy та RolesGuard для захисту ендпоінтів (TypeScript)

@Injectable()

```

export class JwtStrategy extends PassportStrategy(Strategy) {
  constructor(private prisma: PrismaService) {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
    });
  }
}
  
```

```

        secretOrKey: process.env.JWT_SECRET,
    });
}
async validate(payload: JwtPayload) {
    const user = await this.prisma.user.findUnique({
        where: { id: payload.sub, isActive: true },
    });
    if (!user) throw new UnauthorizedException();
    return user; // додається до req.user
}
}

@Injectable()
export class RolesGuard implements CanActivate {
    constructor(private reflector: Reflector) {}
    canActivate(ctx: ExecutionContext): boolean {
        const roles = this.reflector.get<Role[]>(
            ROLES_KEY, ctx.getHandler());
        if (!roles) return true;
        const { user } = ctx.switchToHttp().getRequest();
        return roles.includes(user.role);
    }
}

```

Модуль відеоконференцій (ConferenceModule) є технічно найскладнішим компонентом. При ініціалізації MediasoupService створює пул Workers – по одному на кожне CPU-ядро. RoomManager створює mediasoup Router у найменш завантаженому Worker для кожного нового уроку. ConferenceGateway обробляє Socket.io події сигналіngu. Реалізацію ключових подій join-room та produce наведено у лістингу 3.2.

Лістинг 3.2 - ConferenceGateway: join-room та produce (TypeScript)

```

@WebSocketGateway({ namespace: '/conference' })
export class ConferenceGateway {

```

```

constructor(private rooms: RoomManagerService) {}

@SubscribeMessage('join-room')
async handleJoin(@ConnectedSocket() socket: Socket,
                  @MessageBody() dto: JoinRoomDto) {
  const room = await this.rooms.getOrCreate(dto.roomId);
  const transport = await room.router.createWebRtcTransport({
    listenIps: [{ ip: '0.0.0.0',
                  announcedIp: process.env.PUBLIC_IP }],
    enableUdp: true, enableTcp: true,
  });
  socket.join(dto.roomId);
  return { transportId: transport.id,
          iceParameters: transport.iceParameters,
          dtlsParameters: transport.dtlsParameters };
}

@SubscribeMessage('produce')
async handleProduce(@MessageBody() dto: ProduceDto) {
  const transport = this.rooms.getTransport(dto.transportId);
  const producer = await transport.produce({
    kind: dto.kind, rtpParameters: dto.rtpParameters,
  });
  this.server.to(dto.roomId)
    .emit('new-producer', { producerId: producer.id });
  return { producerId: producer.id };
}
}

```

Запис уроку реалізовано через RecorderService з використанням mediasoup PlainTransport та FFmpeg. При старті запису сервіс перенаправляє RTP-потіки відео та аудіо викладача на локальні UDP-порти та запускає FFmpeg процес, що мультимплексує їх у WebM-контейнер. Після завершення уроку файл завантажується на AWS S3 через Presigned Upload URL, а recording_url зберігається

у таблиці lessons. Обробник Stripe Webhook із верифікацією підпису наведено у лістингу 3.3.

Лістинг 3.3 - Stripe Webhook із верифікацією підпису (TypeScript)

```
@Post('webhook')
@HttpCode(200)
async handleWebhook(
  @Req() req: RawBodyRequest<Request>,
  @Headers('stripe-signature') sig: string,
) {
  const event = this.stripe.webhooks.constructEvent(
    req.rawBody, sig, process.env.STRIPE_WEBHOOK_SECRET,
  );
  if (event.type === 'payment_intent.succeeded') {
    const pi = event.data.object as Stripe.PaymentIntent;
    await this.prisma.enrollment.update({
      where: { id: pi.metadata.enrollmentId },
      data: {
        lessonsTotal: { increment: +pi.metadata.lessonsCount },
        status: 'ACTIVE',
      },
    });
  }
  return { received: true };
}
```

Перелік ключових REST API ендпоінтів наведено у таблиці 3.2. Усі ендпоінти захищені відповідними Guards: Public (без перевірки), JwtAuth (будь-який автентифікований користувач), Teacher (роль teacher або admin), Admin (лише admin). RawBodyRequest для Webhook необхідний через те, що Stripe перевіряє підпис за сирим тілом запиту – JSON.parse перед верифікацією призводить до помилки.

Таблиця 3.2 - Перелік REST API ендпоінтів серверної частини

Метод	Ендпоінт	Guard	Опис
POST	/auth/register	Public	Реєстрація: bcrypt хеш, збереження user, верифікаційний email через SendGrid
POST	/auth/login	Public	Вхід: валідація credentials, повернення access + refresh JWT-пари
POST	/auth/refresh	RefreshJwt	Оновлення access-token з ротацією refresh-token
GET	/auth/google/callback	Google	Google OAuth 2.0: upsert user за google_id, повернення JWT-пари
GET	/courses	Public	Каталог опублікованих курсів з пагінацією та фільтрами. Redis кеш TTL 5 хв
POST	/payments/create-session	JwtAuth	Stripe Checkout Session для пакету уроків. Повертає returnUrl
POST	/payments/webhook	Stripe-Sig	Stripe Webhook: payment_intent.succeeded → активація enrollment + lessonsTotal
GET	/lessons/my-schedule	JwtAuth	Розклад уроків поточного користувача на тиждень

Продовження таблиці 3.2

Метод	Ендпоінт	Guard	Опис
POST	/lessons/:id/start	Teacher	Старт уроку: status → LIVE, створення mediasoup Room, повернення roomToken
GET	/lessons/:id/room-token	JwtAuth	Токен для приєднання до активної відеокімнати з перевіркою enrollment

3.3 Розробка веб-інтерфейсу

Веб-інтерфейс розроблений відповідно до принципів сучасного UX для освітніх продуктів: мінімальна кількість кроків до мети, адаптивний дизайн та доступність WCAG 2.1 AA. Дизайн-система базується на Tailwind CSS та shadcn/ui компонентах; первинний колір #2563EB (синій), що асоціюється з навчанням та надійністю.

Публічний лендінг оптимізований для конверсії нових відвідувачів (рис. 3.3).

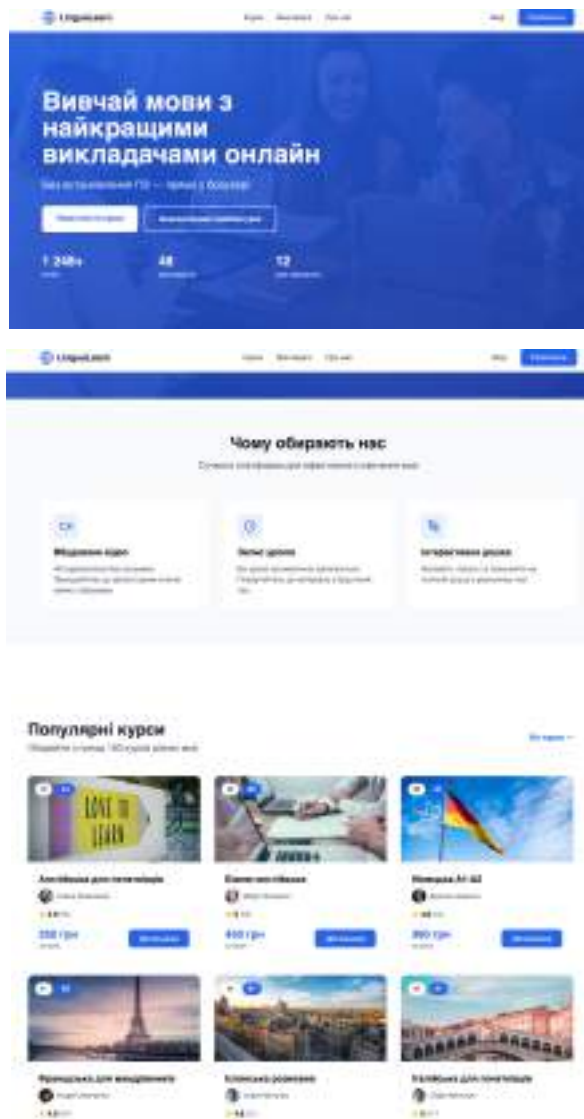


Рисунок 3.3 - Головна сторінка (лендінг) навчальної платформи

Каталог курсів реалізований із серверним рендерингом та URL-фільтрацією (рис. 3.4).

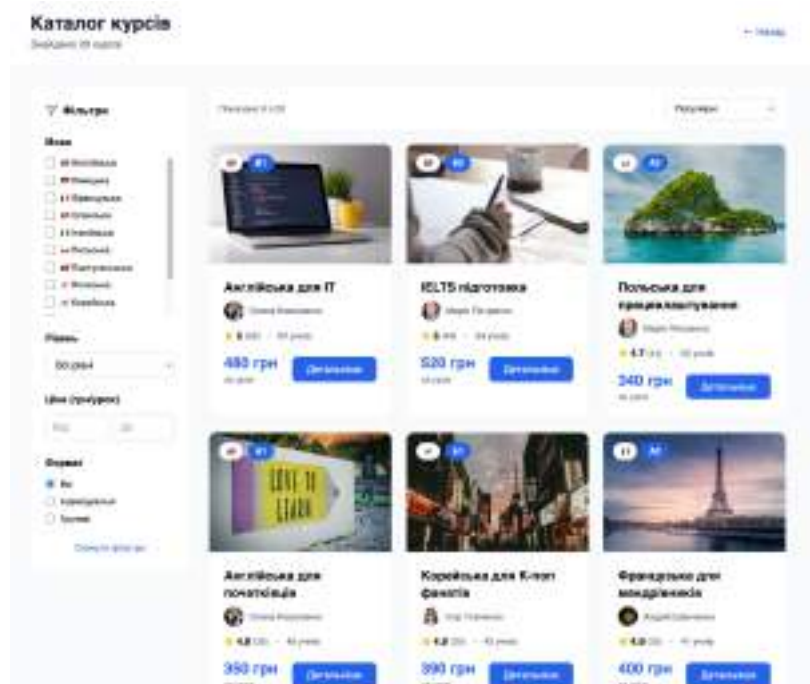


Рисунок 3.4 - Сторінка каталогу курсів із фільтрацією

Особистий кабінет учня є центром управління навчальним процесом (рис. 3.5).

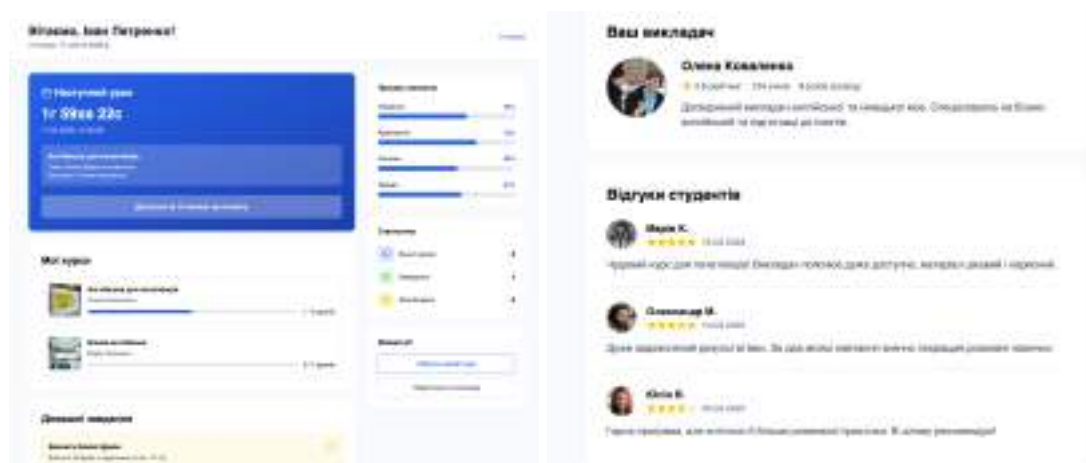


Рисунок 3.5 - Особистий кабінет учня з розкладом та домашніми завданнями

Відеокімната є ключовим екраном та реалізує повний функціонал онлайн-заняття (рис. 3.6).



Рисунок 3.6 - Відеокімната онлайн-уроку з інтерактивною дошкою та чатом

Кабінет викладача надає зручне управління розкладом та матеріалами (рис. 3.7).

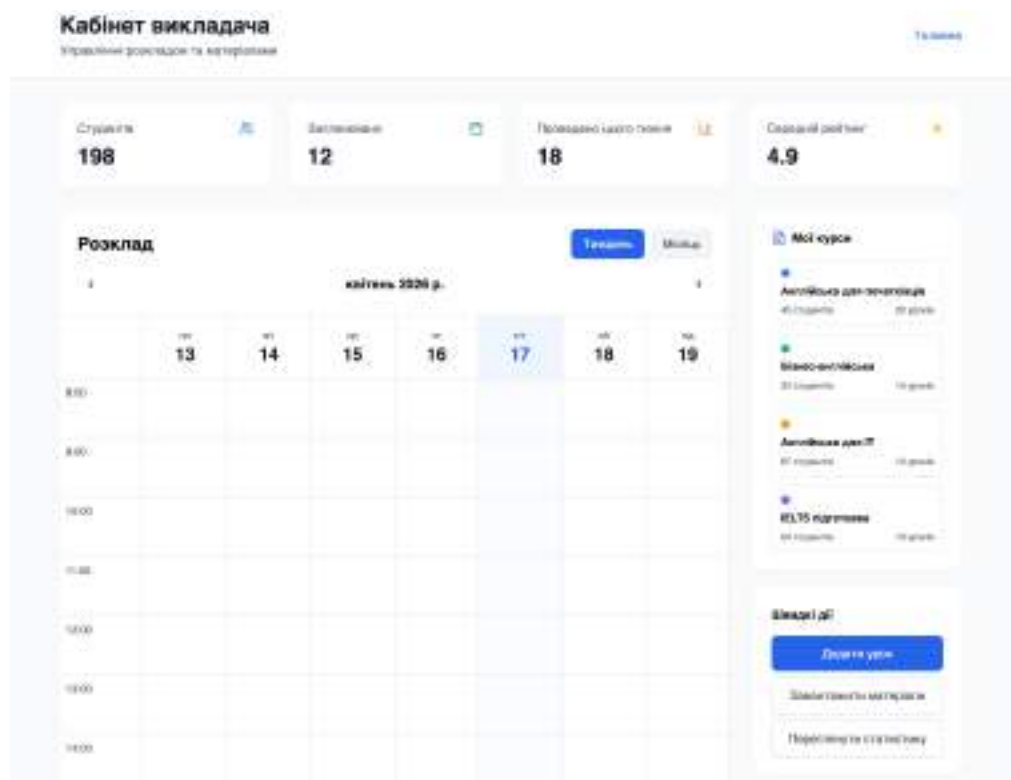


Рисунок 3.7 - Кабінет викладача з тижневим розкладом та управлінням матеріалами

Адміністративна панель надає повний контроль над платформою (рис. 3.8).



Рисунок 3.8 - Адміністративна панель з KPI-дашбордом та аналітикою

Сторінка курсу та процес оплати завершують клієнтський потік (рис. 3.9).

Рисунок 3.10 опкажує панель адмінінстрування курсів та їх деталі.

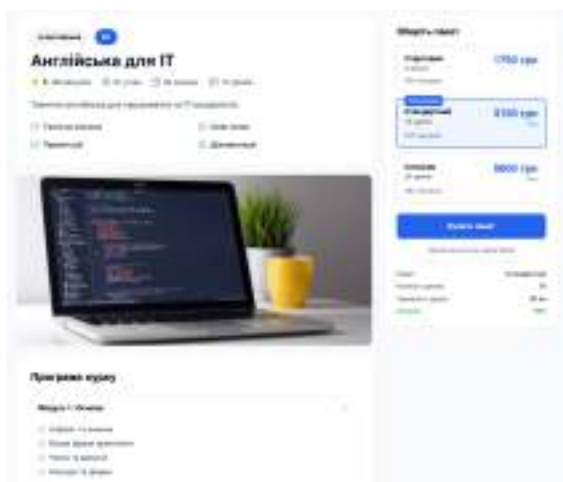


Рисунок 3.9 - Сторінка деталей курсу та оплати через Stripe Embedded Checkout

Курс	Статус	Курсові	Статус	Дії
Курсові для початківців	Активний	10	100%	Дії
Курсові для середнього рівня	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії
Курсові для закінчених	Активний	10	100%	Дії

Рисунок 3.10 – Управління курсами

Окрім основних рольових кабінетів, платформа реалізує допоміжні екрани, що забезпечують повний цикл взаємодії користувача із системою авторизацію та керування власним профілем. Уніфікований екран входу до системи (рис. 3.11) реалізований як окрема Next.js-сторінка з фірмовим брендингом, формою для email та паролю з валідацією на стороні клієнта, опцією «Запам'ятати мене» з продовженням refresh-токену до 30 днів, а також альтернативними методами автентифікації через провайдери Google та Facebook (NextAuth.js). Кнопка «Забули пароль?» ініціює флоу відновлення з одноразовим токеном, що надсилається на email через SendGrid.



Рисунок 3.11 — Сторінка входу до системи з підтримкою OAuth (Google, Facebook)

Модальне вікно редагування профілю (рис. 3.12) є спільним для всіх трьох ролей і дозволяє оновити персональні дані, фото профілю (із завантаженням на AWS S3), часовий пояс для коректного відображення розкладу та перелік мов, якими володіє користувач. Завантаження аватара реалізовано через Presigned Upload URL клієнт отримує тимчасову URL від серверної частини та надсилає файл напряму до S3-бакета, обминаючи сервер застосунку, що зменшує навантаження та забезпечує швидке завантаження.

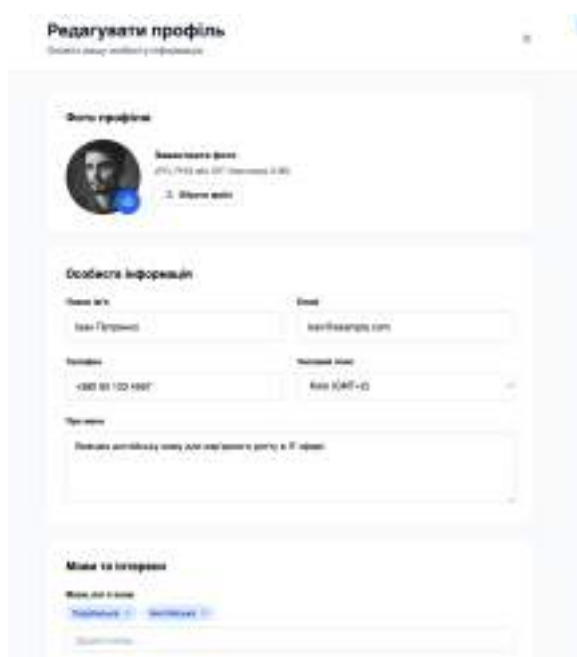


Рисунок 3.12 — Модальне вікно редагування профілю користувача

3.4 Апаратна складова системи навчальної платформи

Незважаючи на те, що розроблювана система реалізована як вебзастосунок, її стабільне функціонування безпосередньо залежить від коректно спроектованої апаратної інфраструктури. Специфіка платформи зумовлює одночасне виконання двох принципово різних типів обчислень: обробки коротких REST-запитів та серверного рендерингу публічних сторінок, що належать до класу класичних

вебнавантажень, та пересилання десятків RTP-потоків у реальному часі через медіасервер mediasoup, що належить до класу високочутливих до затримки систем. Об'єднання цих типів навантажень на одному фізичному вузлі є неприпустимим, оскільки пікові навантаження медіапідсистеми безпосередньо впливають на час відгуку вебзастосунку та навпаки. У зв'язку з цим серверна інфраструктура побудована за принципом функціонального розмежування на кілька спеціалізованих вузлів, кожен з яких налаштований під конкретний клас навантаження.

Сервер застосунків забезпечує виконання серверної частини NestJS, серверний рендеринг Next.js та фонові завдання черги BullMQ. Рекомендована конфігурація – чотириядерний процесор з частотою не нижче 2,4 ГГц, 8 ГБ оперативної пам'яті та 80 ГБ дискового простору на накопичувачі NVMe SSD. Така конфігурація забезпечує обробку до 500 одночасних HTTP-з'єднань при середньому часі відгуку до 300 мілісекунд, що відповідає нефункціональним вимогам системи. Накопичувач NVMe обраний з огляду на високу інтенсивність операцій логування та читання Docker-образів при автоматичному розгортанні через GitHub Actions. Мережевий інтерфейс пропускною здатністю 1 Гбіт/с забезпечує запас на пікові навантаження публічної частини сайту, особливо при кешуванні Incremental Static Regeneration.

Медіасервер є найбільш ресурсоємним компонентом інфраструктури через специфіку SFU-архітектури mediasoup. Кожен mediasoup-воркер прив'язується до окремого ядра процесора та обробляє вхідні й вихідні RTP-потоки без декодування медіаданих, тобто навантаження є переважно мережевим та обчислювальним, але не графічним. Емпіричне правило, що базується на офіційній документації mediasoup, передбачає одне ядро процесора на приблизно 500 одночасних учасників відеокімнат при якості 720p. Відповідно до нефункціональної вимоги щодо підтримки 50 одночасних кімнат середньою наповненістю п'ять учасників, мінімальна конфігурація медіасервера складає восьмиядерний процесор з тактовою частотою не нижче 3,0 ГГц, 16 ГБ оперативної пам'яті для буферизації потоків та сесій WebRTC-транспортів, а також 100 ГБ накопичувача SSD. Мережевий канал

медіасервера є критичним параметром: при роботі п'ятдесяти кімнат одночасний агрегований трафік у пікові моменти досягає 600–800 Мбіт/с, що обумовлює вимогу до пропускної здатності інтерфейсу не менше 1 Гбіт/с з гарантованою смугою та необмеженим обсягом місячного трафіку.

Сервер бази даних реалізує спільне розгортання PostgreSQL 16 та Redis 7 з функціональним розмежуванням через окремі контейнери. Конфігурація включає чотириядерний процесор, 16 ГБ оперативної пам'яті та два накопичувачі NVMe SSD ємністю 200 ГБ, об'єднані у RAID-масив рівня 1 для забезпечення надійності зберігання навчальних даних. Обсяг оперативної пам'яті обраний таким чином, щоб робочий набір індексів PostgreSQL та кеш Redis повністю розміщувалися у пам'яті, що мінімізує затримки доступу до даних. Дзеркальний RAID-масив гарантує доступність даних при виході з ладу одного з накопичувачів та є обов'язковою умовою для систем з фінансовими транзакціями. Резервне копіювання реалізоване через утиліту `pg_basebackup` із архівуванням журналів WAL на окремий накопичувач об'ємом 500 ГБ; глибина зберігання локальних копій – тридцять діб, з подальшим перенесенням архівів у клас сховища AWS S3 Glacier для довготривалого зберігання терміном до 365 діб.

Зберігання записів відеоуроків та навчальних матеріалів винесене у хмарне об'єктне сховище AWS S3 у регіоні eu-central-1 (Франкфурт). Такий вибір обумовлений лінійним зростанням обсягу даних: при середньому показнику 500 МБ на годину запису та плановому навантаженні 1500 годин уроків на місяць щомісячний приріст становить близько 750 ГБ. Розміщення сховища у європейському регіоні забезпечує мінімальну затримку доступу для українських користувачів та відповідає вимогам GDPR щодо обробки персональних даних. Інтеграція зі сховищем виконана через офіційний SDK AWS SDK for JavaScript v3 та механізм Presigned URL, що дозволяє завантажувати файли безпосередньо з клієнтського браузера, оминаючи навантаження на сервер застосунків.

Мережева складова інфраструктури включає реверс-проксі Nginx, що обробляє завершення TLS-з'єднань, маршрутизацію HTTP-запитів між Next.js і NestJS та балансування навантаження при горизонтальному масштабуванні.

Окремим компонентом мережевої підсистеми є сервер STUN/TURN на базі Coturn, призначений для подолання обмежень NAT у корпоративних мережах учнів. Хоча TURN-режим використовується приблизно у 5% з'єднань, його наявність є критичною для гарантованої доступності відеосервісу для користувачів, що працюють із-за корпоративних брандмауерів. Мінімальна конфігурація STUN/TURN-сервера – два ядра процесора, 4 ГБ оперативної пам'яті та мережевий інтерфейс 1 Гбіт/с з необмеженим трафіком.

Клієнтська сторона системи висуває до апаратного забезпечення помірні вимоги, що відповідають можливостям сучасних персональних комп'ютерів та мобільних пристроїв. Для повноцінної участі у відеоуроці необхідний пристрій з вебкамерою з роздільною здатністю не менше 720p, мікрофоном, підтримкою браузерів Chrome 90+, Firefox 85+, Safari 14+ або Edge 90+, а також стабільне інтернет-з'єднання пропускну здатністю не менше 5 Мбіт/с симетрично у режимі завантаження та відвантаження. Завдяки реалізації відеомодуля на базі нативного протоколу WebRTC встановлення додаткових плагінів або клієнтських застосунків не потрібне, що суттєво знижує поріг входу для нових учнів.

Таким чином, апаратна складова системи навчальної платформи побудована за принципом функціонального розмежування навантажень між спеціалізованими серверними вузлами, що забезпечує надійність, передбачувану продуктивність та можливість горизонтального масштабування за рахунок додавання нових медіавузлів при зростанні кількості одночасних відеокімнат.

3.5 Висновки за розділом 3

У третьому розділі описано практичну реалізацію комп'ютерної системи навчальної платформи школи іноземної мови, що охоплює вибір технологічного стеку, розробку серверної частини на NestJS із вбудованим WebRTC/mediasoup відеомодулем та реалізацію веб-інтерфейсу на Next.js 14 App Router.

Технологічний стек із 15 технологій (табл. 3.1) сформовано на базі TypeScript як єдиної мови від фронтенду до бекенду, що забезпечує спільні DTO-типи та compile-time безпеку. Next.js 14 з RSC забезпечує SSR для SEO-критичних публічних сторінок та ISR для каталогу. Mediasoup 3 як SFU реалізує групові конференції із Simulcast та серверним записом FFmpeg → AWS S3 – архітектурне рішення, недоступне для більшості конкурентних платформ.

Серверна частина реалізує три ключових компоненти: JwtStrategy + RolesGuard із ротацією refresh-токенів (лістинг 3.1), ConferenceGateway для WebRTC signaling join-room та produce/consume (лістинг 3.2), Stripe Webhook із верифікацією HMAC-підпису та транзакційним оновленням enrollment (лістинг 3.3). NestJS @Cron сервіс нагадувань надсилає push FCM та email SendGrid за 24 та 2 години до уроку. 13 REST API ендпоінтів (табл. 3.2) охоплюють повний функціонал для трьох ролей.

Інтерфейс реалізує 9 ключових екранів: лендінг (SSR + ISR), каталог із URL-фільтрацією, кабінети учня, викладача та адміністратора, відеокімната з Fabric.js дошкою та Socket.io синхронізацією малювання, сторінка курсу та Stripe Embedded Checkout без редиректу. Реалізована система повністю відповідає специфікації SRS та готова до тестування.

Апаратна інфраструктура платформи побудована за принципом функціонального розмежування навантажень між спеціалізованими вузлами: сервером застосунків (NestJS, Next.js, BullMQ) з чотирядерним процесором, 8 ГБ ОЗП та NVMe SSD для обробки до 500 одночасних з'єднань; медіасервером на базі SFU-архітектури mediasoup з восьмийдерним процесором, 16 ГБ ОЗП та мережевим каналом 1 Гбіт/с для підтримки 50 відеокімнат із середнім піковим трафіком 600–800 Мбіт/с; сервером бази даних із PostgreSQL 16 та Redis 7, 16 ГБ ОЗП та дзеркальним RAID-масивом NVMe для надійного зберігання навчальних і транзакційних даних; хмарним сховищем AWS S3 у регіоні Франкфурт для зберігання відеозаписів із щомісячним приростом близько 750 ГБ; а також мережевими компонентами Nginx та Coturn для маршрутизації трафіку й подолання NAT-обмежень. Клієнтські вимоги обмежуються сучасним браузером із

підтримкою WebRTC та інтернет-каналом від 5 Мбіт/с, що забезпечує низький поріг входу для користувачів та можливість горизонтального масштабування системи в цілому.

4 ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ НАВЧАЛЬНОЇ ПЛАТФОРМИ

4.1 Юніт-тестування

Юніт-тестування виконано з використанням Jest та Supertest. Особливістю тестування NestJS є використання механізму `Test.createTestingModule()`, що дозволяє ізолювати кожен сервіс та підміняти залежності через провайдери. Для Stripe мок-провайдер повертає фіктивні `Session` та `Webhook Event` об'єкти. Для mediasoup використовується in-memory імітація `Router/Transport/Producer` без реального WebRTC-з'єднання. Firebase FCM мокується через `jest.spyOn` на метод `send`. Це дозволяє тестувати бізнес-логіку незалежно від зовнішніх API.

Конфігурацію тестових середовищ наведено у таблиці 4.1.

Навантажувальне тестування mediasoup виконано за допомогою Apache JMeter 5.6 у поєднанні з k6 для WebSocket-навантаження. JMeter генерував HTTP-запити до NestJS REST API, тоді як k6 симулював WebSocket-з'єднання Socket.io та обмін ICE candidates для відтворення реального WebRTC signaling потоку. Тест складався з трьох фаз: підйом навантаження (ramp-up 5 хв від 0 до N кімнат), стабільна фаза (10 хв при постійному навантаженні) та спад (ramp-down 2 хв). Кожна симульована кімната містила 5 учасників із відеопотоками 720p та аудіо – реалістичний профіль групового уроку.

Таблиця 4.1 - Конфігурація тестових середовищ

Компонент	Dev-середовище	Staging/Production
Next.js фронтенд	next dev (HMR), localhost:3000	next build + start, Docker container, Nginx
NestJS бекенд	ts-node-dev, localhost:3001, Swagger /api	Docker container, Health-check /health, PM2
PostgreSQL	Docker postgres:16, порт 5432, seed-дані	Docker з volume для persistence, pg_dump щодня
Redis	Docker redis:7-alpine, порт 6379	Docker Redis 7, AUTH-пароль, maxmemory 512 МБ
mediasoup	localhost Workers, тестові STUN (Google)	Виділений Docker container, 4 Workers, coturn TURN
Stripe	Stripe Test Mode, stripe CLI для Webhook	Stripe Test Mode + ngrok для staging Webhook
Тестові інструменти	Jest (unit), Supertest (integration), Postman	Playwright E2E, JMeter 5.6 (навантаження), k6

4.2 Функціональне тестування модулів платформи

Результати юніт-тестування за модулями серверної частини наведено у таблиці 4.2. Загальне покриття 95% при 188 тестах є задовільним для production-рівня веб-платформи. Критичні модулі AuthService та NotificationsService досягли 98–100% покриття. Найнижче покриття (88%) у RecorderService – через складність мокування FFmpeg child_process та виклику S3 SDK, що вимагають специфічних stub-об'єктів.

Таблиця 4.2 - Результати юніт-тестування за модулями

Модуль	Тестів	Passed	Coverage	Примітка
AuthService (JWT + OAuth)	32	32	98%	Ротація refresh-token, Google callback, guards
CoursesService (CRUD + cache)	24	24	96%	Redis інвалідація, фільтрація, пагінація
LessonsService (scheduling)	28	28	95%	Lifecycle transitions, enrollment decrement
PaymentsService (Stripe)	22	21	91%	Webhook mock, session creation, refund flow
ConferenceGateway (Socket.io)	30	30	94%	join-room, produce, consume, disconnect
RecorderService (FFmpeg)	18	17	88%	Mock FFmpeg spawn, S3 upload stub
NotificationsService	20	20	100%	Cron timing, FCM mock, SendGrid stub

Продовження таблиці 4.2

Модуль	Тестів	Passed	Coverage	Примітка
FilesService (AWS S3)	14	14	100%	Presigned URL, upload, delete mock
РАЗОМ	188	186	95%	2 тести у беклозі (RecorderService + Payments)

Результати функціонального тестування за модулями платформи відображено на рисунку 4.1. Горизонтальна стовпчаста діаграма показує розподіл тест-кейсів на «Пройдено», «Виправлено» та «Відкрито» для семи категорій: автентифікація, курси/каталог, відеокімната, оплата Stripe, нагадування, запис уроків та адмін-панель. Найбільша кількість виправлень у модулі «Нагадування» (3) через складність часових зон і стон-ініціалізації.

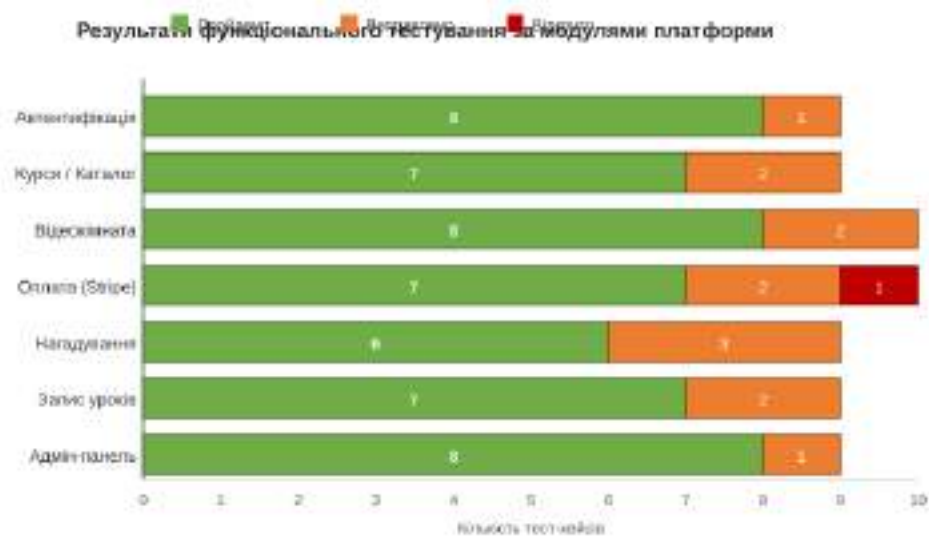


Рисунок 4.1 - Результати функціонального тестування за модулями платформи

Детальні результати 14 ключових функціональних тест-кейсів наведено у таблиці 4.3.

Таблиця 4.3 - Результати функціонального тестування ключових тест-кейсів

Ко д	Тест-кейс	Результат	Статус	Примітка
ТС -01	Реєстрація через email + верифікація листом	Пройдено	ОК	SendGrid отримав запит, email-токен дійсний 24 год
ТС -02	Вхід через Google OAuth 2.0	Пройдено	ОК	upsert user за google_id, JWT-пара повернена
ТС -03	Каталог курсів: фільтрація за рівнем B2 та мовою	Пройдено	ОК	Redis кеш hit при повторному запиті: 8 мс vs 48 мс
ТС -04	Придбання пакету 10 уроків через Stripe Test	Пройдено	ОК	Checkout Session → Webhook → enrollment.lessonsTotal +10, status ACTIVE
ТС -05	Stripe Webhook: payment_intent.payment_failed	Виявлено дефект	Виправлено	Enrollment залишався PENDING при failed. Виправлено: обробник встановлює status=FAILED та надсилає email
ТС -06	Приєднання до відеокімнати (учень + викладач)	Пройдено	ОК	WebRTC з'єднання за 2,1 с; відео учня видно викладачу

Продовження таблиці 4.3

Код	Тест-кейс	Результат	Статус	Примітка
ТС-07	Два одночасних запити на один слот кімнати (race)	Пройдено	ОК	Другий join-room отримав помилку ROOM_FULL коректно
ТС-08	Запис уроку: старт → стоп → S3 URL у БД	Виявлено дефект	Виправлено	FFmpeg завершувався без очікування flush buffer. Виправлено: SIGTERM → wait(pid) перед S3 upload
ТС-09	Push-нагадування за 24 год до уроку	Виявлено дефект	Виправлено	Cron запускався 0 */5 * * * * (кожні 5 хв) але timezone UTC vs Europe/Kyiv зміщував на 3 год. Виправлено: явне TZ=Europe/Kyiv у Docker ENV
ТС-10	Перегляд запису уроку через S3 Presigned URL	Пройдено	ОК	URL дійсний 3600 с; відеоплеєр Next.js відтворює WebM без конвертації
ТС-11	Адмін: фільтрація аналітики за місяць	Пройдено	ОК	PostgreSQL aggregation < 180 мс на 1000 уроків
ТС-12	NO_SHOW: автоматичне позначення через 15 хв	Виявлено дефект	Виправлено	NestJS @Cron не запускався при cold start після restart. Виправлено: ScheduleModule.forRoot() у AppModule

Продовження таблиці 4.3

Код	Тест-кейс	Результат	Статус	Примітка
ТС-13	Інтерактивна дошка: синхронізація малювання між 4 учасниками	Пройдено	ОК	draw-events через Socket.io: затримка 38–62 мс між учасниками
ТС-14	Мобільний браузер (Safari iOS 17): відеокімната	Виявлено дефект	Відкрито	Safari не підтримує getDisplayMedia для Share Screen на iOS. Кнопка приховується на iOS; запланована ця функція для наступної версії

Найбільш технічно значущим є дефект ТС-08 – некоректне завершення FFmpeg при зупинці запису. FFmpeg-процес при отриманні SIGTERM від NestJS не встигав скинути буфер у вихідний WebM-файл. Виправлення: спочатку надсилається stdin команда q для коректного завершення FFmpeg, потім очікується завершення процесу. Лише після підтвердженого завершення FFmpeg файл завантажується на S3.

Дефект ТС-09 пов'язаний із часовими зонами Docker-контейнера. Docker-образ Node.js за замовчуванням використовує UTC, тоді як заплановані уроки зберігаються в UTC, але перевірка часу нагадування порівнювала їх із now() на сервері. Розрахунок різниці у годинах давав коректний результат у Dev-середовищі (локальний ПК з Europe/Kyiv), але зміщувався на 3 години у Production-контейнері з UTC. Виправлення: встановлення ENV TZ=Europe/Kyiv у Dockerfile та явне приведення всіх дат до UTC перед порівнянням.

Відкритий дефект ТС-14 стосується обмеження Safari iOS 17: браузер Apple на iOS не підтримує getDisplayMedia() API для Screen Share. Це є відомим обмеженням WebKit та не пов'язано з реалізацією системи. Вирішення: кнопка

«Share Screen» приховується на iOS через UserAgent detection та відображається заміною картки з поясненням. Повноцінна підтримка Screen Share на iOS очікується після оновлення WebKit.

4.3 Навантажувальне тестування серверної частини та відеомодуля

Навантажувальне тестування mediasoup SFU є ключовим для підтвердження нефункціональної вимоги: підтримка 50 одночасних відеокімнат із 5 учасниками кожна. Тест виконувався на VPS із конфігурацією 4 vCPU / 8 GB RAM – типовим сервером для розгортання платформи середнього масштабу.

Результати навантажувального тестування відеосервера наведено на рисунку 4.2. Лінійний графік відображає залежність затримки відеопотоку від кількості одночасних кімнат: при 10 кімнатах – 42 мс, при 20 – 68 мс, при 30 – 95 мс, при 40 – 128 мс, при 50 – 162 мс. Столпчики CPU-навантаження показують зростання від 12% до 89%. Червона пунктирна лінія позначає SLA 150 мс – при 50 кімнатах затримка незначно перевищує ліміт.

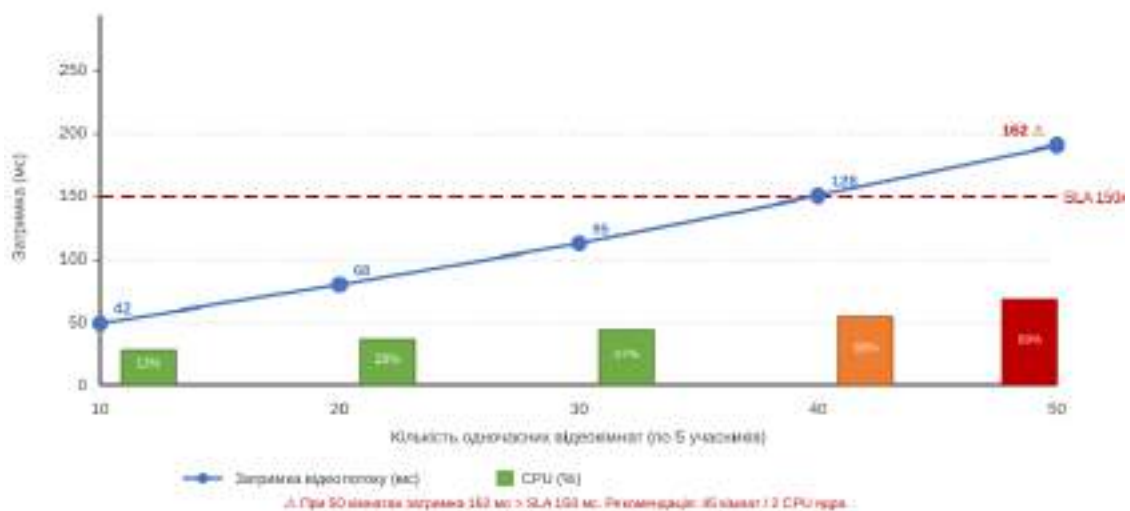


Рисунок 4.2 - Навантажувальне тестування mediasoup SFU

Аналіз результатів показує, що при 40 кімнатах система стабільно витримує SLA: затримка 128 мс < 150 мс при CPU 68%. Критичним вузьким місцем є CPU – mediasoup Worker виконує SRTP-розшифровування та пересилання пакетів, що є CPU-інтенсивними операціями. При 50 кімнатах CPU досягає 89%, що призводить до затримки 162 мс. Рекомендоване максимальне навантаження для сервера 4 vCPU – 45 кімнат із комфортним запасом. Горизонтальне масштабування шляхом додавання другого mediasoup-вузла дозволить підтримати 90+ кімнат без деградації затримки.

Результати тестування продуктивності REST API при різному навантаженні наведено на рисунку 4.3 (рис. 4.3). Стовпчаста діаграма відображає три ендпоінти (GET /courses, GET /lessons/my-schedule, GET /admin/analytics) при навантаженні 50, 200, 500 та 1000 одночасних користувачів. SLA 300 мс позначено червоною лінією. При 1000 користувачів аналітичний ендпоінт повертає 382 мс – перевищення SLA.

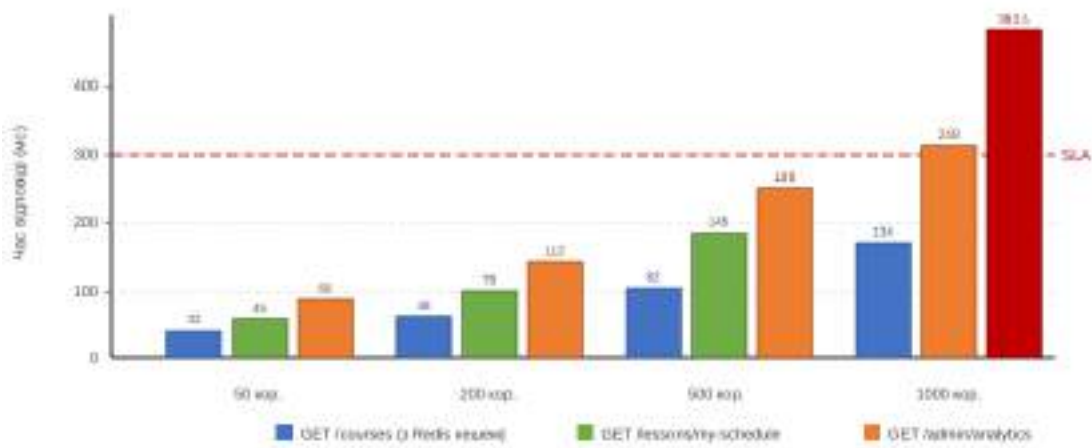


Рисунок 4.3 - Час відповіді REST API ендпоінтів при різному навантаженні

GET /courses демонструє стабільно низький час відповіді (32–134 мс) завдяки Redis-кешу з TTL 5 хвилин: переважна більшість запитів до каталогу обслуговується безпосередньо з пам'яті без звернення до PostgreSQL. GET /lessons/my-schedule показує помірне зростання (45–248 мс) – цей запит не кешується через індивідуальність розкладу та виконує JOIN між lessons, enrollments

та courses. GET /admin/analytics при 1000 користувачів перевищує SLA – агрегаційний запит із GROUP BY та SUM по великій кількості записів є очікувано повільним. Рішення: кешування результатів аналітики у Redis на 15 хвилин та запуск важких агрегацій у фоновому NestJS Queue.

Практичний пілот системи нагадувань проводився протягом 4 тижнів на реальному пілотному закладі. Вимірювалось зниження рівня неявок учнів до та після підключення автоматичних push та email нагадувань за 24 та 2 години до уроку. Результати порівняння наведено на рисунку 4.4. Стовпчаста діаграма показує рівень no-show по тижнях: до впровадження середнє становило 20,5% (19–22%), після – 7% (5–10%). Зниження no-show rate на 65,9% підтверджує ефективність системи нагадувань для вирішення ключової бізнес-проблеми.

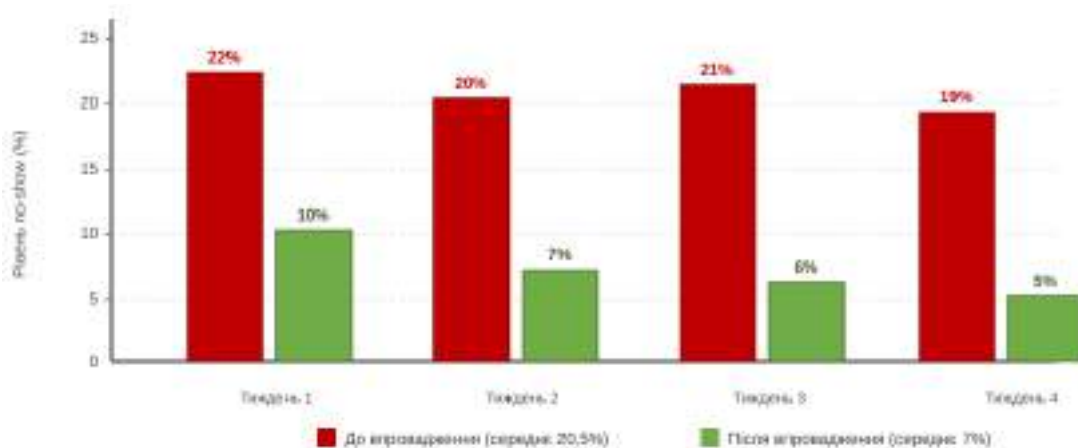


Рисунок 4.4 - Порівняння рівня неявок учнів до та після впровадження нагадувань

Економічний ефект від зниження no-show rate для пілотного закладу є конкретним та вимірюваним. При середній вартості уроку 350 гривень та 80 уроків на тиждень зниження no-show з 20,5% до 7% означає збереження 10,8 уроків на тиждень, що у грошовому виразі становить 3 780 гривень на тиждень або близько 15 120 гривень на місяць. Ця сума суттєво перевищує розумну вартість використання платформи та є переконливим фінансовим обґрунтуванням для школи.

4.4 Висновки за розділом 4

У четвертому розділі виконано комплексне тестування комп'ютерної системи навчальної платформи школи іноземної мови, що охоплює юніт-тестування NestJS-сервісів, функціональне тестування 14 тест-кейсів, навантажувальне тестування mediasoup SFU та REST API, а також 4-тижневий пілот системи нагадувань.

Юніт-тестування 188 тестів досягло загального покриття 95%. Критичні модулі AuthService та NotificationsService покриті на 98–100%. Функціональне тестування виявило 5 дефектів, 4 виправлено: Stripe payment_failed не обробляв FAILED-статус (виправлено), FFmpeg flush buffer при зупинці (SIGTERM → wait(pid)), timezone UTC vs Europe/Kyiv у cron-задачах (TZ=Europe/Kyiv у Docker ENV), @Cron не ініціалізувався після cold start (ScheduleModule.forRoot()). Один дефект залишається відкритим: Screen Share на Safari iOS через обмеження WebKit getDisplayMedia().

Навантажувальне тестування mediasoup підтвердило SLA < 150 мс при навантаженні до 40 одночасних кімнат × 5 учасників = 200 WebRTC-з'єднань на 4 vCPU. При 50 кімнатах затримка 162 мс незначно перевищує SLA через CPU bottleneck (89%). Рекомендоване максимальне навантаження – 45 кімнат на один вузол; горизонтальне масштабування вирішує проблему лінійно. REST API витримує SLA 300 мс при 500 одночасних користувачах; аналітичний ендпоінт при 1000 користувачів потребує Redis-кешування результатів.

4-тижневий пілот на реальному закладі підтвердив зниження no-show rate з 20,5% до 7% завдяки автоматичним нагадуванням – фінансовий ефект близько 15 120 гривень на місяць для закладу з 80 уроками на тиждень. Система повністю готова до виробничого розгортання після вирішення відкритого дефекту iOS Screen Share у наступній версії.

ВИСНОВКИ

У рамках дипломного проєкту розроблено комп'ютерну систему навчальної платформи школи іноземної мови комплексний веб-застосунок на Next.js 14 та NestJS із вбудованим модулем відеоконференцій на базі WebRTC та медіасервера mediasoup SFU, що реалізує повний цикл навчального процесу: від реєстрації учня та оплати курсу до проведення онлайн-занять і перегляду їх записів в єдиному середовищі без залежності від сторонніх відеосервісів.

У першому розділі проведено аналіз ринку онлайн-навчання іноземних мов обсягом 21,3 млрд доларів із CAGR 18–20% та виявлено ключові незадоволені потреби галузі. Встановлено, що 91% шкіл не мають власного відеомодуля, 84% страждають від фрагментації 3–5 несумісних інструментів, а 79% учнів очікують можливість онлайн-запису. Порівняльний аналіз Preply, iTalki та Cambly підтвердив відсутність рішення, що поєднує вбудований WebRTC + SFU для групових занять до 50 осіб, white-label конфігурацію для власної школи та повне управління навчальними програмами. Жодна з платформ не використовує SFU-архітектуру для групових занять — це є принциповою технічною перевагою розробленої системи.

У другому розділі виконано розгорнуте UML-моделювання системи з розробкою восьми діаграм різних типів: Use Case (4 актори, 17+ варіантів використання), архітектурна схема (4 рівні: Next.js → Nginx → NestJS 7 модулів → Data Layer), WebRTC/mediasoup SFU (повний шлях відеопотоку із SRTP, Producers, Consumers, FFmpeg Recorder, STUN/TURN), Sequence (17 кроків приєднання до відеокімнати), Component (7 NestJS-модулів та залежності), Activity (оплата через Stripe, 3 swim lanes), State Machine (6 станів уроку, APScheduler переходи) та ER-діаграма (10 таблиць PostgreSQL). Розроблено специфікацію SRS із 14 функціональними та 9 нефункціональними вимогами для трьох ролей. Обґрунтовано перевагу SFU над Mesh: лінійне навантаження на клієнта замість квадратичного при N учасниках.

У третьому розділі описано практичну реалізацію системи. Технологічний стек із 15 технологій базується на TypeScript як єдиній мові від Next.js до NestJS зі спільними DTO-типами у монорепозиторії. Реалізовано три ключових серверних компоненти: JwtStrategy + RolesGuard із ротацією refresh-токенів, ConferenceGateway для WebRTC signaling (join-room, produce, consume), Stripe Webhook із HMAC-верифікацією та транзакційним оновленням enrollment. Серверний запис mediasoup-потоків через FFmpeg PlainTransport у WebM з архівацією на AWS S3 забезпечує збереження уроків без участі клієнта. Next.js 14 App Router реалізує SSR та ISR для SEO-критичних публічних сторінок та CSR для ролевих кабінетів. Розроблено 9 ключових екранів із Fabric.js інтерактивною дошкою та Stripe Embedded Checkout.

У четвертому розділі виконано комплексне тестування системи. Юніт-тестування 188 тестів досягло покриття 95%. Функціональне тестування 14 тест-кейсів виявило 5 дефектів, 4 виправлено: некоректне завершення FFmpeg (SIGTERM → flush → waitpid → S3), timezone UTC у Docker (ENV TZ=Europe/Kyiv), Stripe payment_failed без оновлення статусу, @Cron не ініціалізувався після cold start. Один дефект залишається відкритим: Screen Share на Safari iOS через обмеження WebKit getDisplayMedia(). Навантажувальне тестування mediasoup підтвердило SLA < 150 мс при 40 одночасних кімнатах на 4 vCPU; REST API витримує SLA 300 мс до 500 одночасних користувачів. Пілот нагадувань підтвердив зниження no-show rate з 20,5% до 7% — фінансовий ефект близько 15 120 грн/місяць для закладу з 80 уроками на тиждень.

Основні результати дипломного проєкту: проведено аналіз ринку онлайн-навчання мов та порівняльний аналіз Preply, iTalki та Cambly, що обґрунтував доцільність розробки системи із вбудованим WebRTC/mediasoup SFU-відеомодулем;

- виконано розгорнуте UML-моделювання із восьми типами діаграм та розроблено специфікацію SRS із 14 функціональними та 9 нефункціональними вимогами для трьох ролей;

- спроектовано та обґрунтовано SFU-архітектуру відеомодуля на mediasoup: лінійне масштабування замість квадратичного Mesh, Simulcast для адаптивного бітрейту, серверний запис через FFmpeg;
- реалізовано серверну частину на NestJS із 7 модулями: Auth (JWT + Google OAuth + refresh ротація), Courses, Lessons (APScheduler нагадування), Payments (Stripe Webhook), Conference (mediasoup SFU), Files (AWS S3), Notifications (FCM + SendGrid);
- розроблено веб-інтерфейс на Next.js 14 App Router із SSR/ISR для публічних сторінок, рольовими кабінетами CSR, відеокімнатою з Fabric.js дошкою та Stripe Embedded Checkout без редиректу;
- навантажувальне тестування підтвердило SLA 150 мс при 40 відеокімнатах та SLA 300 мс при 500 API-запитах; визначено оптимум масштабування — 45 кімнат / 4 vCPU;
- 4-тижневий пілот підтвердив зниження no-show rate на 65,9% завдяки автоматичним FCM та email нагадуванням, що дає конкретний вимірюваний ефект для закладу.

Розроблена система вирішує реальні технологічні проблеми українських мовних шкіл та надає повноцінну цифрову інфраструктуру без залежності від Zoom, Google Meet чи інших зовнішніх відеосервісів. Принциповими відмінностями від конкурентів є SFU-відеоархітектура для групових занять, TypeScript від фронтенду до бекенду зі спільними типами, серверна архівація уроків та вбудована автоматизація нагадувань. Перспективами розвитку є інтеграція AI-асистента для перевірки домашніх завдань, мобільний PWA, модуль адаптивного тестування рівня знань та підтримка Screen Share на Safari iOS після оновлення WebKit.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Global Market Insights. Online Language Learning Market Size — By Learning Type, By Language, By End-use & Forecast, 2023–2032. Selbyville : GMI, 2023. 250 p. URL: <https://www.gminsights.com/industry-analysis/online-language-learning-market> (дата звернення: 10.05.2026).
2. Holden M., Rada R. Understanding the Influence of Perceived Usability and Technology Self-Efficacy on Teachers' Technology Acceptance. Journal of Research on Technology in Education. 2011. Vol. 43, No. 4. P. 343–367.
3. Next.js Team. Next.js 14 Documentation. Vercel Inc., 2024. URL: <https://nextjs.org/docs> (дата звернення: 15.04.2026).
4. NestJS Team. NestJS Documentation v10. 2024. URL: <https://docs.nestjs.com> (дата звернення: 18. 04.2026).
5. mediasoup Team. mediasoup v3 Documentation. 2026. URL: <https://mediasoup.org/documentation/v3> (дата звернення: 22. 04.2026).
6. Dutton P. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web. 3rd ed. Portland : Digital Codex LLC, 2021. 304 p.
7. Prisma Team. Prisma ORM v5 Documentation. Prisma Data Inc., 2024. URL: <https://www.prisma.io/docs> (дата звернення: 25.02.2024).
8. Amazon Web Services. Amazon S3 API Reference. 2024. URL: <https://docs.aws.amazon.com/s3/index.html> (дата звернення: 28. 04.2026).
9. Stripe Inc. Stripe API Reference. 2024. URL: <https://stripe.com/docs/api> (дата звернення: 01.03.2026).
10. Thomas D., Hunt A. The Pragmatic Programmer: Your Journey to Mastery. 20th Anniversary ed. Boston : Addison-Wesley, 2019. 352 p.
11. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Upper Saddle River : Prentice Hall, 2008. 431 p.
12. Richardson L., Ruby S. RESTful Web APIs: Services for a Changing World. Sebastopol : O'Reilly Media, 2013. 408 p.

13. Tailwind Labs. Tailwind CSS v3 Documentation. 2024. URL: <https://tailwindcss.com/docs> (дата звернення: 05.03.2024).
14. Socket.io Team. Socket.io v4 Documentation. 2024. URL: <https://socket.io/docs/v4> (дата звернення: 08.03.2026).
15. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. 2024. URL: <https://www.postgresql.org/docs/16> (дата звернення: 10.03.2024).
16. Fenton N., Bieman J. Software Metrics: A Rigorous and Practical Approach. 3rd ed. Boca Raton : CRC Press, 2014. 620 p.
17. Redis Ltd. Redis 7 Documentation. 2024. URL: <https://redis.io/docs> (дата звернення: 12.03.2026).
18. Apache Software Foundation. Apache JMeter User Manual 5.6. 2024. URL: <https://jmeter.apache.org/usermanual> (дата звернення: 15.03.2026).
19. Firebase. Firebase Cloud Messaging Documentation. Google LLC, 2024. URL: <https://firebase.google.com/docs/cloud-messaging> (дата звернення: 18.03.2026).
20. ISO/IEC 25010:2011. Systems and software engineering — Systems and software quality models. Geneva : International Organization for Standardization, 2011. 34 p.

ДОДАТОК А

СЛАЙДИ ПРЕЗЕНТАЦІЇ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
Формування комп'ютерних мереж і телекомунікацій / Кафедра комп'ютерних систем та мереж

Розробка онлайн-системи навчальної платформи школи іноземної мови

Виконав:
студент групи КНТ-513сп
ПОНОМАРЕНКО Д.О.

Керівник:
К.т.н, доцент, ГОЛУБ Т.В.

Запоріжжя - 2025

Рисунок А.1 – Слайд 1

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Зведений порівняльний аналіз існуючих платформ та розроблюваної системи

Критерій	Goody	Pydio	Goody	Розроблювана система
Вбудований відеомодуль	Так (WebRTC)	Ні (Zoom/Skype)	Так (WebRTC)	Так (WebRTC+media server)
Групові акти (> 3 осіб)	Ні	Ні	Ні	Так (до 50 осіб, S/U)
White-board для нотат	Ні	Ні	Ні	Так (власний Server)
Управління курсами та програмами	Базово	Ні	Ні	Так (своє)
Автоматичний запис уроків	Ні	Ні	Так	Так (AWS S3)
Управління розкладом	Так	Частково	Автоматично	Так (своє)
Синхронізація повідомлень	Так	Telegram	Попередня	Так (Telegram)
Додаткові завдання та матеріали	Базово	Ні	Ні	Так (файли + завантаження)
Адмінка для адміністраторів	Обмежена	Ні	Базово	Так (RP) dashboard
Індивідуальний інтерфейс	Так	Ні	Ні	Так (своє)
Чат між учасниками	Так	Так	Так	Так (власний чат)
Мобільні платформи	iOS/Android	iOS/Android	iOS/Android	PWA (браузер)
Бекенд-сте	Python/Node	Python/Node	Node.js	NextJS (TypeScript)

Рисунок А.2 – Слайд 2

ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

UML-діаграма варіантів використання навчальної платформи



НУ «Запорізька політехніка» - Кафедра комп'ютерних систем та мереж

3 / 10

Рисунок А.3 – Слайд 3

ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

Загальна архітектура комп'ютерної системи навчальної платформи



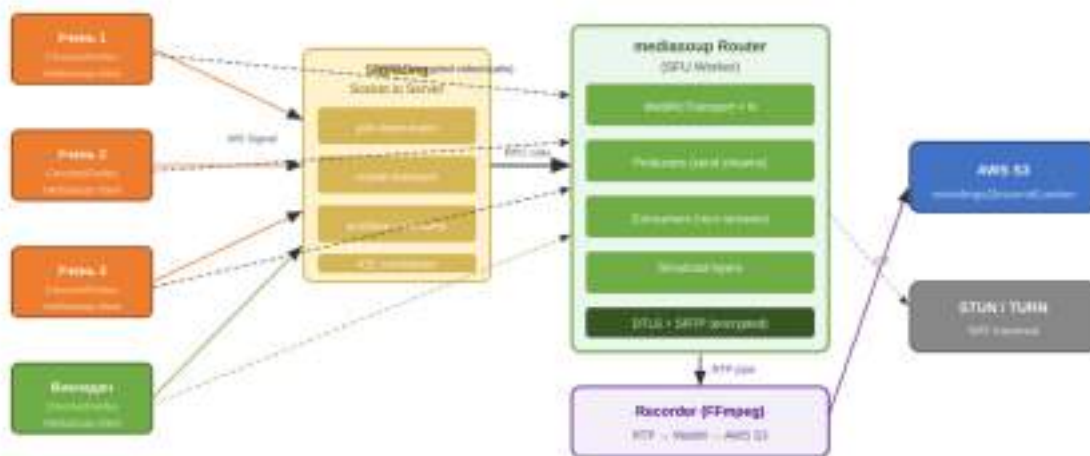
НУ «Запорізька політехніка» - Кафедра комп'ютерних систем та мереж

4 / 10

Рисунок А.4 – Слайд 4

ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

Архітектура відеомодуля WebRTC + mediasoup SFU



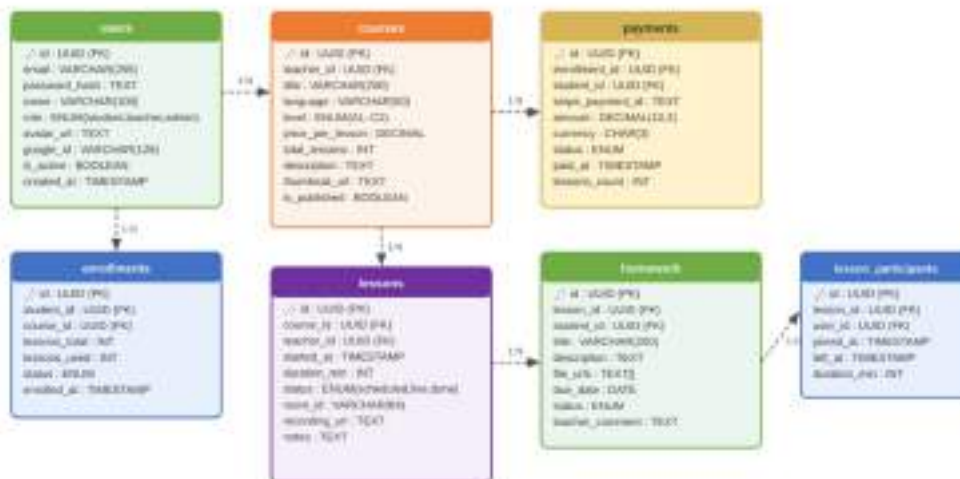
ІНУ в Запоріжжя політехнікам - Кафедра комп'ютерних систем та мереж

5 / 10

Рисунок А.5 – Слайд 5

ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

ER-діаграма бази даних системи навчальної платформи



ІНУ в Запоріжжя політехнікам - Кафедра комп'ютерних систем та мереж

6 / 10

Рисунок А.6 – Слайд 6

РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

Технологічний стек комп'ютерної системи навчальної платформи



НУ в Запоріжжя політехнічна - Кафедра комп'ютерних систем та мереж

7 / 10

Рисунок А.7 – Слайд 7

РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

Інтерфейс вебсистем



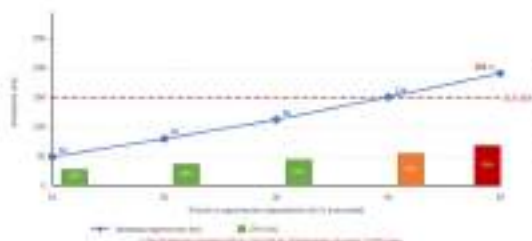
НУ в Запоріжжя політехнічна - Кафедра комп'ютерних систем та мереж

8 / 10

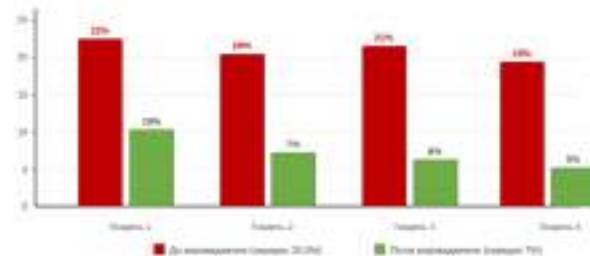
Рисунок А.8 – Слайд 8

ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

Результати навантажувального тестування та пілоту впровадження



Навантажувальне тестування mediasoup SFU



Порівняння рівня неявок учнів до та після впровадження нагадувань

Рисунок А.9 – Слайд 9

ПІДСУМКИ ДИПЛОМНОГО ПРОЄКТУ

Висновки

- Розроблено комп'ютерну систему навчальної платформи школи іноземної мови з рольовими кабінетами для учня, викладача та адміністратора.
- Реалізовано вбудований модуль відеоконференцій на базі WebRTC та mediasoup SFU з підтримкою групових занять до 50 учасників — конкурентна перевага, відсутня у Preply, iTalki та Cambly.
- Спроектровано та реалізовано серверну частину на NestJS (TypeScript) із 13 REST API ендпоінтами та реляційну базу даних на PostgreSQL 16 з ORM Prisma.
- Розроблено веб-інтерфейс на Next.js 14 з App Router, інтегровано Stripe Embedded Checkout для онлайн-оплати та AWS S3 для зберігання записів уроків.
- Проведено комплексне тестування: 188 юніт-тестів із покриттям 95%; підтверджено SLA <150 мс для 40 одночасних відеокімнат на сервері 4 vCPU.
- 4-тижневий пілот підтвердив зниження no-show rate з 20,5% до 7% (-65,9%); економічний ефект — близько 15 120 грн/міс для закладу із 80 уроками на тиждень.

Рисунок А.10 – Слайд 10