

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебплатформа організації навчального процесу
онлайн-школи»**

Виконав студент групи ІПЗ-22-2
БІЛИЙ Микола Миколайович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк
«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БІЛИЙ Микола Миколайович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Вебплатформа організації навчального процесу онлайн-школи»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: об'єктом дослідження є вебплатформа для організації навчального процесу онлайн-школи. Предметом дослідження є підходи та засоби розробки сучасного вебзастосунку, що забезпечує управління розкладом занять, ведення оцінок, моніторинг прогресу учнів, чат у реальному часі та адміністрування користувачів з чіткою рольовою моделлю.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	01.03.2026р.	01.03.2026р
2	Чижмотря О.Г.	01.03.2026р	01.03.2026р
3	Чижмотря О.Г.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра виконана на тему «Вебплатформа організації навчального процесу онлайн-школи». Випускна кваліфікаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 33 найменувань (3 стор.), 5 додатків (8 стор.), 11 таблиць та 31 ілюстрацій по тексту. Повний обсяг роботи становить 90 сторінок, з них 71 сторінок основного тексту.

У першому розділі проведено аналіз предметної галузі та виявлено ключові проблеми традиційних методів організації навчального процесу в онлайн-школах. Здійснено порівняльний аналіз існуючих аналогів — Google Classroom, Moodle та Coursera — за функціональними можливостями та визначено конкурентні переваги власної розробки. Обґрунтовано вибір клієнт-серверної архітектури на основі REST API та WebSocket, а також інструментальних засобів реалізації: TypeScript, Node.js, Express.js, PostgreSQL, Prisma, React.

У другому розділі визначено рольову модель системи з чотирма типами користувачів (адміністратор, куратор, викладач, учень) та побудовано діаграму варіантів використання. Розроблено структуру реляційної бази даних із дев'ятьма таблицями та її фізичну схему. Описано ключові алгоритми роботи системи — автентифікацію на основі JWT та Google OAuth 2.0, перевірку конфліктів розкладу занять, розрахунок прогресу навчання учня та реалізацію взаємодії в реальному часі через WebSocket — із наведенням відповідних діаграм послідовності. Описано структуру серверної та клієнтської частин програмного комплексу.

					ІПЗ.КР.Б-121-26-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Білий М.М.			Вебплатформа організації навчального процесу онлайн-школи. Пояснювальна записка	Літ.	Арк.
Керівник		Чижмотря О.Г.					3
						Аркуші	90
Рецензент		Вакалюк Т. А.				Житомирська політехніка група ІПЗ-22-2	
Зав. каф.		Вакалюк Т. А.					

У третьому розділі описано порядок розгортання вебплатформи у хмарному середовищі (Render, Netlify, Supabase) та порядок локального встановлення системи. Наведено структуру інтерфейсу користувача для кожної ролі (дашборди, розклад занять, домашні завдання, чат, профіль, адміністрування). Проведено функціональне тестування програмного комплексу за двадцятьма тестовими сценаріями, що охоплюють автентифікацію, управління розкладом, оцінювання, домашні завдання, реальнотчасову комунікацію та адміністрування користувачів.

Висновки містять результати аналізу предметної галузі та існуючих аналогів, обґрунтування архітектурних і технологічних рішень, опис розробленої структури бази даних та рольової моделі, перелік реалізованих функціональних можливостей системи, результати функціонального тестування, а також перспективи подальшого розвитку платформи.

У додатках наведено діаграму варіантів використання, фізичну схему бази даних, діаграми послідовності ключових процесів системи (автентифікація, створення заняття) та фрагмент програмної реалізації схеми бази даних.

Ключові слова: REACT, NODE.JS, EXPRESS, PRISMA, TYPESCRIPT, POSTGRESQL, WEBSOCKET, UML, ВЕБПЛАТФОРМА, ОНЛАЙН-ШКОЛА, АВТОМАТИЗАЦІЯ НАВЧАННЯ, REST API, JWT, GOOGLE OAUTH, СИСТЕМА УПРАВЛІННЯ НАВЧАННЯМ.

		Білий М.М.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Чижмотря О.Г.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ABSTRACT

The explanatory note for the bachelor's qualification thesis was completed on the topic "Web Platform for Organizing the Educational Process of an Online School". The graduation qualification thesis consists of an introduction, 3 chapters, conclusions, a list of references containing 33 sources (3 pages), 5 appendices (8 pages), 11 tables, and 31 illustrations throughout the text. The total volume of the thesis is 90 pages, including 71 pages of the main text.

The first chapter presents an analysis of the subject area and identifies the key problems of traditional methods of organizing the educational process in online schools. A comparative analysis of existing analogues — Google Classroom, Moodle, and Coursera — was carried out based on their functional capabilities, and the competitive advantages of the developed solution were determined. The choice of a client-server architecture based on REST API and WebSocket, as well as the implementation tools TypeScript, Node.js, Express.js, PostgreSQL, Prisma, and React, was substantiated.

The second chapter defines the system's role model with four types of users (administrator, curator, teacher, student) and presents a use case diagram. The structure of a relational database consisting of nine tables and its physical schema were developed. The key algorithms of the system operation — authentication based on JWT and Google OAuth 2.0, schedule conflict detection, student learning progress calculation, and real-time interaction implementation via WebSocket — are described with the corresponding sequence diagrams provided. The structure of the server-side and client-side components of the software system is described.

The third chapter describes the procedure for deploying the web platform in a cloud environment (Render, Netlify, Supabase) and the procedure for local system installation. The structure of the user interface for each role (dashboards, class schedule, homework, chat, profile, administration) is presented. Functional testing of the software system was carried out using twenty test scenarios covering authentication, schedule management, assessment, homework, real-time communication, and user administration.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

The conclusions contain the results of the subject area and existing analogues analysis, justification of architectural and technological solutions, a description of the developed database structure and role model, a list of the implemented functional capabilities of the system, the results of functional testing, as well as prospects for further platform development.

The appendices include a use case diagram, the physical database schema, sequence diagrams of the key system processes (authentication, lesson creation), and a fragment of the software implementation of the database schema.

Keywords: REACT, NODE.JS, EXPRESS, PRISMA, TYPESCRIPT, POSTGRESQL, WEBSOCKET, UML, WEB PLATFORM, ONLINE SCHOOL, LEARNING AUTOMATION, REST API, JWT, GOOGLE OAUTH, LEARNING MANAGEMENT SYSTEM.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП.....	9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ВИМОГ ДО СИСТЕМИ....	11
1.1. Постановка задачі та аналіз предметної області.....	11
1.2. Аналіз існуючих аналогів програмного продукту.....	13
1.3. Вибір архітектури системи	19
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	22
Висновки до розділу 1	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ	27
2.1. Визначення варіантів використання та рольова модель системи	27
2.2. Розробка бази даних системи	30
2.3. Проєктування алгоритмів роботи системи.....	40
2.4. Реалізація програмного комплексу та компонентна будова додатку	47
Висновки до розділу 2	55
РОЗДІЛ 3. ОПИС ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ СИСТЕМИ	56
3.1. Порядок встановлення та налаштування системи	56
3.2. Структура інтерфейсу вебплатформи	60
3.3. Тестування роботи програмного комплексу	73
Висновки до розділу 3	77
ВИСНОВКИ	78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
ДОДАТКИ	83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ПЗ – програмне забезпечення

СКБД – система керування базами даних

API – прикладний програмний інтерфейс (Application Programming Interface)

REST – архітектурний стиль побудови розподілених систем для взаємодії клієнта і сервера (Representational State Transfer)

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol)

JWT – формат токена для автентифікації користувачів (JSON Web Token)

OAuth – відкритий протокол авторизації (Open Authorization)

ORM – технологія об'єктно-реляційного відображення даних (Object-Relational Mapping)

SPA – односторінковий вебзастосунок (Single Page Application)

UML – уніфікована мова моделювання (Unified Modeling Language)

SaaS – програмне забезпечення як послуга (Software as a Service)

URL – уніфікований локатор ресурсу (Uniform Resource Locator)

FK – зовнішній ключ (Foreign Key)

PK – первинний ключ (Primary Key)

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Стрімкий розвиток інформаційних технологій та масштабна цифровізація суспільства суттєво змінили підходи до організації освітнього процесу. Дистанційне навчання, яке ще кілька років тому вважалося додатковим інструментом, сьогодні перетворилося на повноцінну альтернативу традиційній очній формі здобуття освіти. Онлайн-школи, репетиторські центри та приватні освітні платформи активно впроваджують цифрові рішення для управління навчальним процесом, однак більшість доступних на ринку систем орієнтована на великі університети або масові відкриті курси й не враховує специфіку невеликих освітніх організацій із персоналізованим підходом до навчання.

Актуальність теми дослідження зумовлена зростаючою потребою в ефективних інструментах автоматизації навчального процесу для онлайн-шкіл, де ключову роль відіграє індивідуальна взаємодія між учнем, викладачем та куратором. Існуючі рішення — такі як Google Classroom, Moodle або Coursera — є або надмірно складними у налаштуванні, або не забезпечують необхідного рівня контролю за успішністю учнів у форматі приватної школи. Це зумовлює потребу в розробці спеціалізованої вебплатформи, яка враховує особливості організаційної структури онлайн-школи з чіткою рольовою моделлю та персоналізованим управлінням навчанням.

Метою випускної роботи є проектування та розробка вебплатформи організації навчального процесу онлайн-школи, що забезпечує автоматизацію управління розкладом занять, ведення електронного журналу оцінок, відстеження прогресу навчання учнів, внутрішню комунікацію між учасниками освітнього процесу та адміністрування користувачів з розподілом прав доступу за ролями.

Для досягнення поставленої мети необхідно вирішити такі завдання: провести аналіз предметної галузі та існуючих аналогів програмних продуктів для організації дистанційного навчання; визначити функціональні

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

та нефункціональні вимоги до розроблюваної системи; обґрунтувати вибір архітектурних рішень та інструментальних засобів реалізації; спроектувати структуру бази даних та модульну архітектуру системи; реалізувати програмний комплекс із повним набором запланованих функціональних можливостей; провести тестування розробленої системи та описати порядок її розгортання.

Об'єктом дослідження є процес організації навчального процесу в онлайн-школі із використанням вебтехнологій.

Предметом дослідження є архітектура, функціональні компоненти та методи реалізації вебплатформи для управління освітнім процесом онлайн-школи.

В роботі використано методи об'єктно-орієнтованого аналізу та проектування, методи моделювання програмних систем засобами UML 2.0, методи клієнт-серверної взаємодії на основі REST API та WebSocket, а також методи компонентного проектування інтерфейсу користувача.

Практичне значення отриманих результатів полягає в тому, що розроблена вебплатформа є повністю функціональним програмним продуктом, розгорнутим у хмарному середовищі, який може бути впроваджений у діяльність приватних онлайн-шкіл та репетиторських центрів для автоматизації ключових процесів управління навчанням.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ВИМОГ ДО СИСТЕМИ

1.1. Постановка задачі та аналіз предметної області

Сучасний стан розвитку інформаційних технологій та цифрова трансформація суспільства створюють передумови для широкого впровадження автоматизованих систем управління в усіх сферах людської діяльності. Особливої актуальності це питання набуває у сфері освіти, де дистанційне навчання стає не просто альтернативою традиційному, а його невід'ємною складовою. Онлайн-школи, репетиторські центри та приватні освітні платформи щоденно стикаються з проблемами ефективної організації навчального процесу в цифровому середовищі.

Аналіз діяльності типових онлайн-шкіл дозволив виявити ключові проблеми, які виникають при використанні традиційних методів управління навчанням. Ведення обліку занять та оцінок за допомогою паперових журналів або електронних таблиць є вкрай неефективним — такий підхід призводить до помилок при внесенні даних, дублювання інформації та відсутності централізованого доступу для всіх учасників процесу. Комунікація між учасниками навчального процесу через загальні месенджери не дозволяє структурувати повідомлення в контексті конкретних занять та призводить до втрати важливої інформації. Управління розкладом в ручному режимі створює ризики подвійного бронювання часу викладачів та конфліктів у розкладі. Відстеження прогресу навчання кожного учня вимагає значних витрат часу від викладачів та кураторів, а отримані дані часто є фрагментарними і не дозволяють своєчасно виявляти проблеми.

З урахуванням виявлених проблем сформульовано задачу розробки вебплатформи для організації навчального процесу онлайн-школи. Система має забезпечувати управління розкладом занять у вигляді інтерактивного календаря з можливістю створення як одноразових, так і повторюваних щотижневих занять. Платформа повинна підтримувати ведення електронного журналу оцінок з автоматичним розрахунком успішності та динаміки навчання учня. Необхідною є реалізація системи внутрішньої комунікації в

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

реальному часі між усіма учасниками освітнього процесу, а також автоматичне сповіщення про нові заняття, зміни в розкладі та виставлені оцінки. Система повинна підтримувати чітку рольову модель з розподілом прав доступу між учнями, викладачами, кураторами та адміністраторами, а також забезпечувати адміністрування користувачів із можливістю проведення вступної співбесіди перед наданням доступу до платформи.

Для досягнення поставленої мети визначено функціональні та нефункціональні вимоги до системи. Серед функціональних вимог виокремлено такі: реєстрація та автентифікація користувачів з підтримкою традиційного входу за електронною поштою і паролем, а також входу через обліковий запис Google за протоколом OAuth 2.0; подача заявки на вступ новим користувачем із зазначенням бажаної ролі та мотивації і проходження процедури співбесіди перед отриманням доступу до системи; управління розкладом занять у вигляді тижневого та списочного календаря з можливістю створення одноразових і повторюваних щотижневих занять, їх редагування та скасування; виставлення оцінок двох типів — поточних і тематичних — безпосередньо в картці заняття із автоматичним збереженням та розрахунком прогресу навчання; система домашніх завдань з можливістю прикріплення тексту та посилань, подання виконаної роботи учнем і перевірки викладачем; внутрішній чат у реальному часі між учасниками навчального процесу на базі протоколу WebSocket; система автоматичних сповіщень про нові заняття, зміни в розкладі, виставлені оцінки та суттєве зниження успішності учня; управління профілями користувачів з можливістю завантаження аватара у хмарне сховище; адміністрування облікових записів з розподілом ролей, блокуванням та призначенням кураторів до учнів.

Нефункціональні вимоги до системи включають: забезпечення безпеки шляхом використання хешування паролів бібліотекою bcrypt та підписаних токенів JWT для автентифікації; рольовий контроль доступу до ресурсів API на рівні серверного middleware; коректна обробка одночасних WebSocket-з'єднань без втрати повідомлень; відсутність конфліктів у

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

розкладі занять завдяки серверній перевірці перетину часових інтервалів; мінімальний час відгуку системи при роботі з базою даних завдяки оптимізованим запитам через ORM Prisma; адаптивний інтерфейс, придатний для роботи на десктопних та мобільних пристроях.

Розроблювана платформа орієнтована на онлайн-школи з персоналізованим підходом до навчання, де кожен учень має індивідуальний розклад занять зі своїми викладачами та персонального куратора, який відповідає за моніторинг успішності та взаємодію з педагогічним складом. Така модель суттєво відрізняється від масових платформ групового навчання і потребує спеціалізованих архітектурних рішень, зорієнтованих на зв'язки між конкретними парами учень-викладач, а не на загальні курси з потоками слухачів.

Загальна структура системи передбачає три рівні: рівень представлення (клієнтський застосунок React), рівень бізнес-логіки (серверний застосунок Node.js/Express) та рівень даних (база даних PostgreSQL). Взаємодія між рівнями здійснюється через чітко визначені інтерфейси — REST API для CRUD-операцій та WebSocket для реальних подій, що забезпечує слабе зв'язування компонентів та можливість незалежного розвитку кожного рівня системи.

1.2. Аналіз існуючих аналогів програмного продукту

Для визначення функціональних можливостей розроблюваної системи та виявлення її конкурентних переваг проведено аналіз існуючих програмних продуктів для організації дистанційного навчання. Розглянуто три найбільш поширені платформи: Google Classroom, Moodle та Coursera.

Google Classroom є безкоштовною вебплатформою від компанії Google, що інтегрується з іншими сервісами екосистеми Google — Drive, Docs, Calendar та Gmail (рис.1.1). Платформа призначена для організації навчального процесу в школах та університетах і надає можливості створення та управління навчальними курсами, розміщення навчальних матеріалів різних форматів, створення та перевірки завдань з виставленням

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижморя О.Г.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

оцінок, а також організації комунікації через стрічку новин та коментарі (рис.1.2) [1]. Перевагами платформи є повна безкоштовність, простий інтерфейс та відсутність необхідності налаштування серверної інфраструктури. Водночас Google Classroom має суттєві обмеження для використання в приватній онлайн-школі: відсутність персонального розкладу занять у форматі тижневого календаря для пари учень-викладач, відсутність ролі куратора як окремого учасника процесу, відсутність вбудованої системи миттєвих повідомлень та неможливість налаштування власного процесу прийому учнів [1].

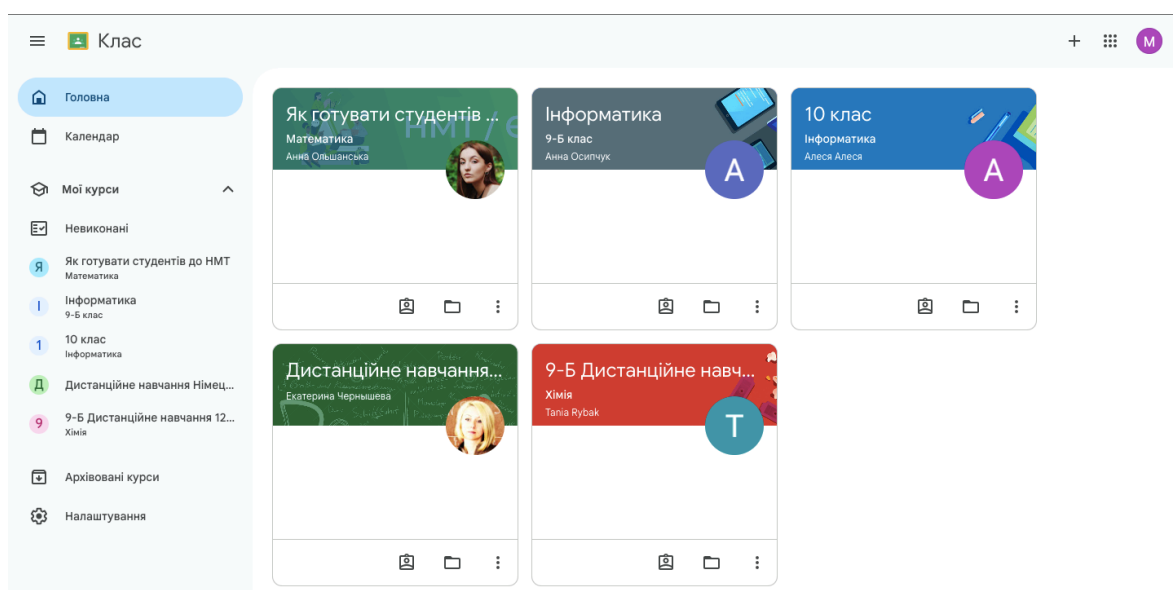


Рис. 1.1. Головна сторінка Google Classroom.

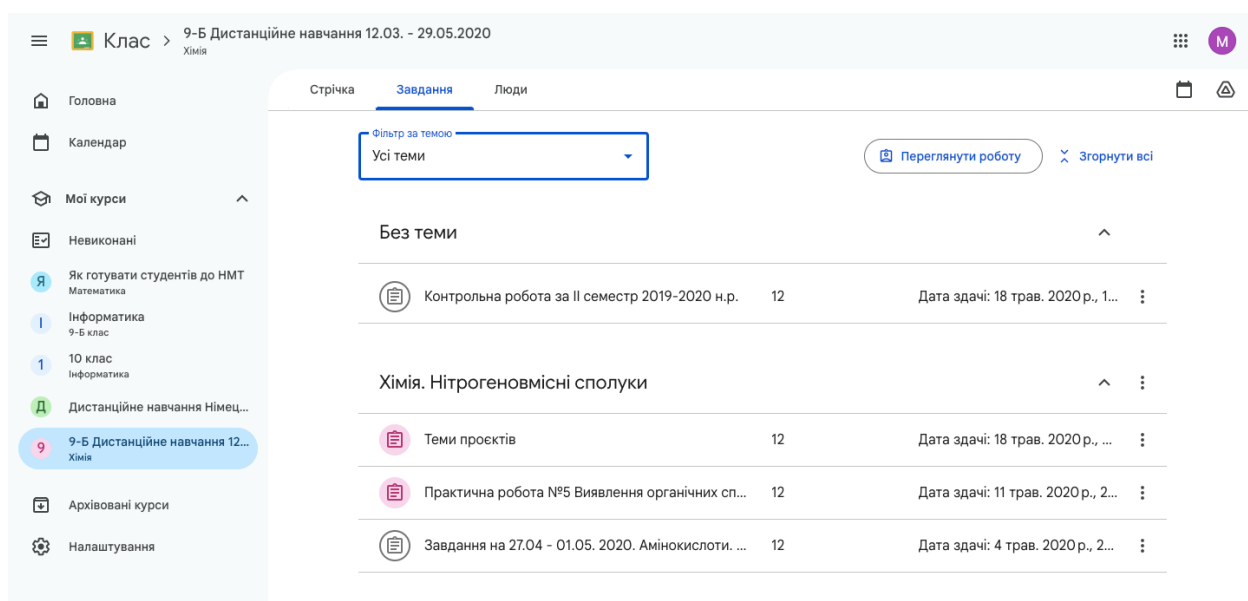


Рис. 1.2. Вкладка завдань що були задані на курсі.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижморя О.Г.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Moodle є відкритою системою управління навчанням, що широко використовується у вищих навчальних закладах по всьому світу. Система надає широкий набір інструментів: управління курсами та модулями, форуми, тести, завдання, оцінювання, журнали успішності та розширену систему звітності (рис.1.3). Moodle є надзвичайно гнучким рішенням завдяки модульній архітектурі та відкритому вихідному коду. Проте для розгортання та підтримки системи необхідні значні технічні ресурси та адміністративні знання, а сам інтерфейс є складним для кінцевих користувачів без технічної підготовки. Крім того, Moodle орієнтована на групове навчання в рамках курсів, тоді як онлайн-школи потребують персоналізованого підходу з індивідуальним розкладом для кожної пари учень-викладач [2].

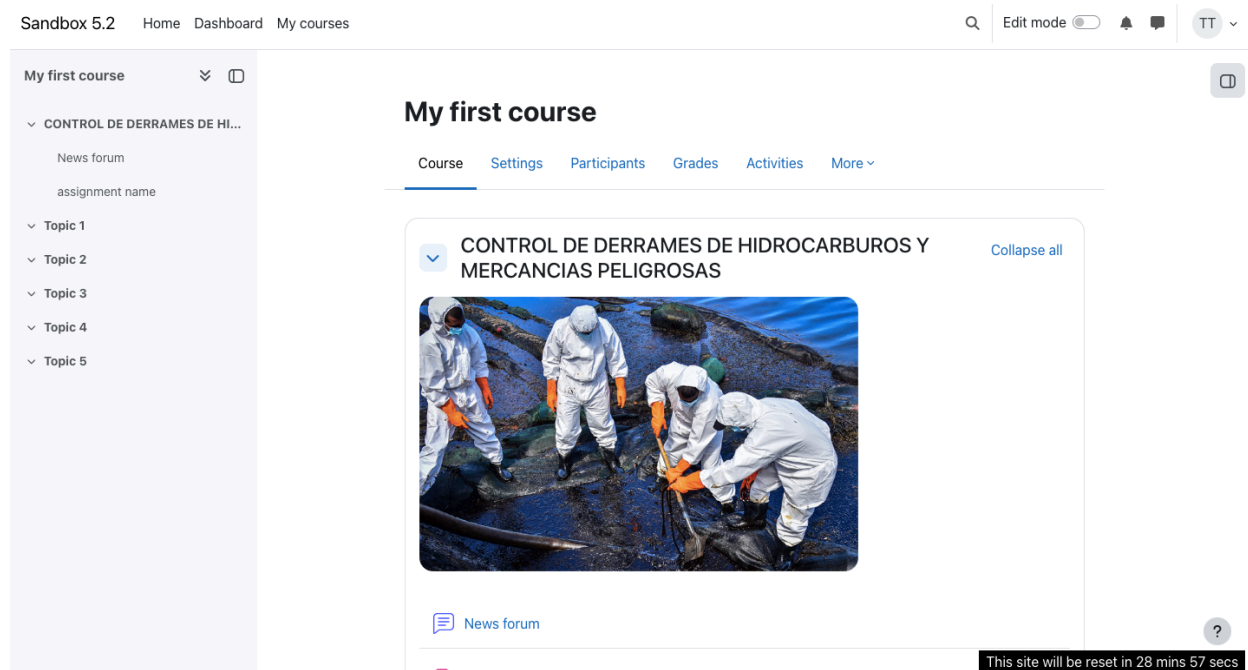


Рис. 1.3. Інтерфейс сторінки курсу в системі Moodle.

Coursera є однією з найбільших платформ масових відкритих онлайн-курсів у світі. Платформа пропонує тисячі курсів від провідних університетів та компаній, підтримує відеолекції, тести, проектні роботи та видачу сертифікатів. Однак Coursera є закритою комерційною платформою, непридатною для розгортання власної онлайн-школи — вона не підтримує індивідуальне планування занять між конкретним учнем та викладачем, не

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

має механізму призначення куратора до учня та не забезпечує внутрішню комунікацію в реальному часі [3] (рис.1.4).

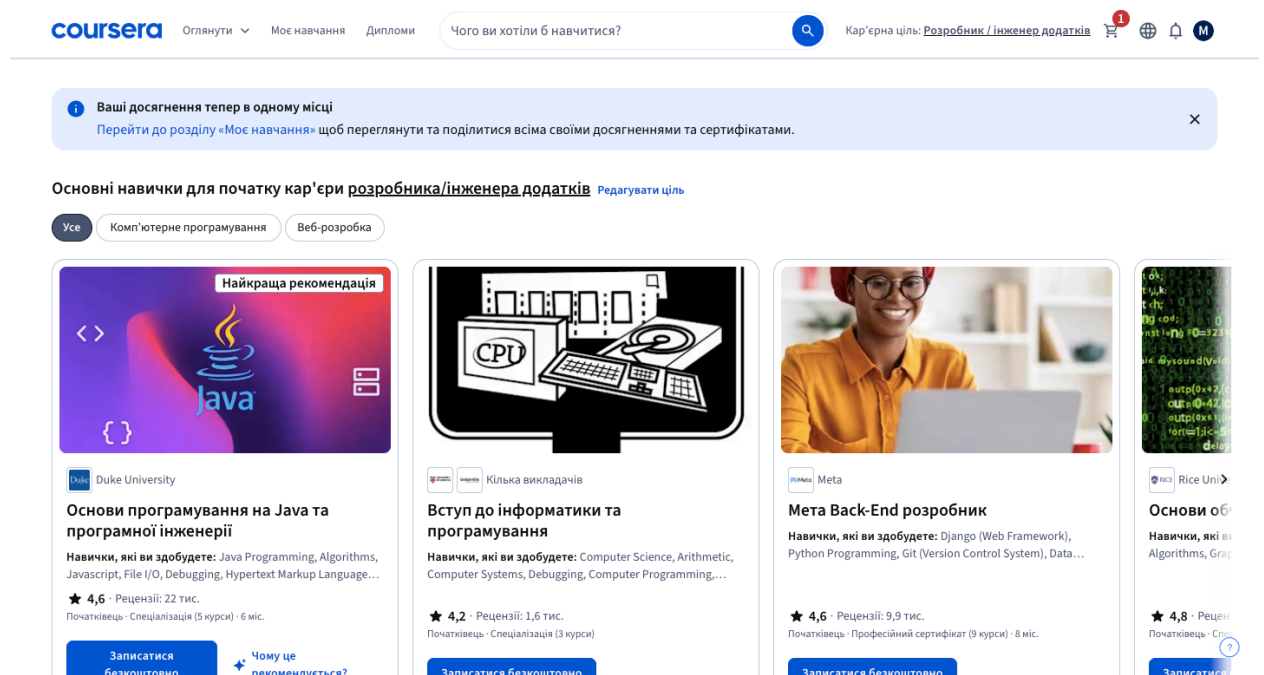


Рис. 1.4. Головний екран на Coursera.

Для об'єктивного аналізу кожної платформи розглянуто детальніше ключові аспекти їх функціонування, що безпосередньо стосуються потреб приватної онлайн-школи.

Google Classroom реалізує модель навчання на основі «класів» — постійних груп учнів, об'єднаних навколо одного курсу або предмету. Викладач створює клас та запрошує до нього учнів, після чого усі завдання, матеріали та оголошення публікуються в потоці класу і є доступними всім його учасникам одночасно. Така модель є зручною для шкільних класів або навчальних груп, але не відповідає потребам онлайн-школи з індивідуальними заняттями, де розклад і прогрес кожного учня є унікальними. Платформа не містить вбудованого інструменту для планування занять у форматі тижневого календаря та не підтримує автоматичне відстеження успішності учня в динаміці з виявленням тенденцій. Інтеграція з Google Calendar дозволяє створювати події, але не забезпечує перевірку конфліктів у розкладі на рівні пари учень-викладач. Функціонал сповіщень обмежено стрічкою новин класу та електронними

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

листами Google, без можливості отримувати повідомлення у реальному часі всередині самого застосунку без оновлення сторінки.

Moodle як система управління навчанням (LMS) орієнтована на організацію структурованих навчальних курсів із модульною побудовою — курс складається з тем, кожна тема містить ресурси та активності. Адміністрування системи вимагає технічної компетентності: Moodle потребує налаштування власного або орендованого сервера, встановлення СКБД, конфігурації файлового сховища та регулярного оновлення. Великий функціональний набір платформи (понад 600 офіційних плагінів) одночасно є її слабкою стороною — складність інтерфейсу перевищує потреби невеликої приватної школи, а необхідність налаштування численних модулів вимагає значних витрат часу на початковому етапі впровадження. Система оцінювання в Moodle реалізує журнал оцінок у розрізі курсів, але не підтримує концепцію занять у форматі окремих зустрічей між учнем та викладачем із прив'язаними до них оцінками та домашніми завданнями. Вбудований чат Moodle не підтримує WebSocket і реалізований через технологію довгого опитування, що обмежує його продуктивність і зручність у реальному використанні.

Coursera представляє принципово іншу модель — МООС-платформу (масові відкриті онлайн-курси), де навчальний контент є заздалегідь записаним і структурованим, а взаємодія між учасниками здійснюється переважно через форуми обговорень. Платформа не передбачає живих занять між конкретним учнем та викладачем — весь прогрес будується на проходженні відеоматеріалів, автоматичних тестів та проектних завдань з peer review. Це робить Coursera непридатною для моделі онлайн-школи з персоналізованим репетиторством або викладанням. Coursera є закритою платформою, на якій навчальні організації можуть лише розміщувати свій контент за умовами угоди з оператором, але не можуть розгортати власні екземпляри системи для внутрішнього використання. Відсутність реалізації

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

функцій куратора, індивідуального розкладу та внутрішнього чату підтверджує обмеженість цієї платформи для поставлених задач.

Крім розглянутих трьох аналогів, на ринку існують й інші рішення. Microsoft Teams for Education розширює можливості комунікаційної платформи до освітнього використання, але залишається в першу чергу інструментом групової відеокommунікації, а не повноцінною LMS. Canvas LMS широко використовується у вищих навчальних закладах США та підтримує широкий API для інтеграцій, однак є комерційним продуктом з орієнтацією на великі університети без необхідної для онлайн-школи рольової моделі з куратором. ClassDojo орієнтована на початкову школу та взаємодію з батьками, що не відповідає профілю цільової аудиторії розроблюваної системи.

Порівняльний аналіз розглянутих платформ демонструє, що жодна з них не задовольняє повною мірою потреби приватної онлайн-школи з персоналізованим навчанням. Дана інформація проілюстрована у таблиці 1.1. Власна розробка відрізняється від аналогів наявністю чотирирівневої рольової моделі (учень, викладач, куратор, адміністратор), персональним тижневим розкладом для кожної пари учень-викладач із підтримкою повторюваних занять, системою моніторингу прогресу учня в розрізі кожного викладача, вбудованим чатом у реальному часі на базі WebSocket, а також автоматизованим процесом прийому нових учнів через систему заявок та вступної співбесіди.

Таблиця 1.1

Таблиця порівняння аналогів

Функція	Google Classroom	Moodle	Coursera	Власна розробка
Персональний розклад учня	-	+/-	-	+
Чат в реальному часі	+/-	+/-	-	+ (WebSocket)

Рольова модель з куратором	-	-	-	+ (4 полі)
Прогрес по викладачам	-	+/-	-	+ (Детально)
Домашні завдання	+	+	-	+ (Розширено)
Автоматичне сповіщення	+/-	+/-	-	+ (WebSocket)
Безкоштовність	+	+	-	+
Простота налаштування	+	-	-	+ (SaaS)
Підтримка OAuth 2.0	+	+	+	+
Вбудоване сховище файлів	+/-	+	-	+ (Supabase)

1.3. Вибір архітектури системи

Архітектура розроблюваної вебплатформи базується на клієнт-серверній моделі з чітким розподілом відповідальності між фронтендом та бекендом [5]. Такий підхід забезпечує незалежне масштабування окремих компонентів системи, спрощує підтримку та тестування, а також дозволяє в майбутньому підключати додаткові клієнти — наприклад, мобільні застосунки — без зміни серверної частини [4].

Для взаємодії між клієнтом та сервером обрано архітектурний стиль REST API. Кожен ресурс системи — користувачі, заняття, оцінки, домашні завдання, повідомлення — представлено окремим набором ендпоінтів, що забезпечує передбачувану структуру запитів та відповідей. Для реалізації функціоналу сповіщень та чату в реальному часі застосовано протокол WebSocket, який дозволяє підтримувати постійне двостороннє з'єднання між сервером та клієнтом без необхідності постійного опитування сервера.

Серверна частина реалізована у вигляді монолітного застосунку на базі Node.js з фреймворком Express.js. Монолітна архітектура обрана свідомо

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

— для проекту такого масштабу вона є оптимальним рішенням, що забезпечує простоту розгортання, відлагодження та підтримки порівняно з мікросервісним підходом [25]. Для роботи з базою даних використовується ORM Prisma, яка забезпечує типобезпечний доступ до даних та спрощує управління міграціями схеми бази даних [11].

Клієнтська частина реалізована у вигляді односторінкового застосунку (SPA) на базі бібліотеки React з використанням збирача Vite. Підхід SPA забезпечує плавну навігацію між сторінками без перезавантаження браузера, що є критично важливим для застосунків з реальноточасовим оновленням даних, таких як чат та сповіщення [33].

Рішення щодо розподілу відповідальності між клієнтом та сервером є фундаментальним архітектурним вибором [6]. Підхід, за якого сервер генерує HTML-сторінки (server-side rendering у традиційному розумінні), є прийнятним для статичного контенту, але не є оптимальним для застосунків з активним оновленням даних у реальному часі — кожна зміна стану вимагатиме повного або часткового перезавантаження сторінки. Натомість архітектура SPA (Single Page Application), де весь інтерфейс формується на стороні браузера за рахунок JavaScript-коду, а сервер надає лише дані через API, забезпечує плавну навігацію, можливість інкрементального оновлення окремих елементів інтерфейсу та природну інтеграцію з WebSocket-з'єднаннями.

REST API побудовано відповідно до принципів архітектурного стилю REST, описаних у дисертації Роя Філдінга. Ключовими з них є: клієнт-серверне розділення відповідальності, відсутність стану (statelessness), при якому кожен запит містить всю необхідну для його обробки інформацію без покладання на серверну сесію, використання єдиного інтерфейсу для взаємодії через стандартні HTTP-методи (GET, POST, PUT, PATCH, DELETE) та ідентифікація ресурсів через URL-адреси. Наприклад, ресурс «заняття» доступний за адресою /api/lessons, конкретне заняття — за /api/lessons/:id, оцінки конкретного заняття — за /api/lessons/:id/grades. Такий

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижморя О.Г.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

підхід забезпечує передбачувану структуру API та спрощує його документування і тестування [7].

Для реалізації реальночасової функціональності — чату та сповіщень — застосовано протокол WebSocket (RFC 6455) [8]. На відміну від HTTP, WebSocket підтримує повнодуплексний зв'язок: після встановлення з'єднання сервер може самостійно ініціювати відправку даних клієнту без попереднього запиту. Це принципово важливо для системи сповіщень, де подія (наприклад, виставлення оцінки) відбувається не в клієнтському браузері, а на сервері у відповідь на запит іншого користувача, і повинна бути миттєво доставлена зацікавленому учаснику. Для реалізації WebSocket-сервера та клієнта використано бібліотеку Socket.IO [22], яка реалізує власний протокол поверх WebSocket з підтримкою автоматичного відновлення з'єднання при розриві та резервним переходом на HTTP Long Polling за відсутності підтримки WebSocket у браузері.

Серверна частина реалізована за патерном MVC (Model-View-Controller) у спрощеному варіанті: Model відповідає схемам Prisma та функціям доступу до даних, Controller — функціям-обробникам маршрутів (route handlers), View відсутній у традиційному розумінні, оскільки сервер повертає не HTML, а JSON-дані. Middleware-компоненти (authMiddleware, roleMiddleware) реалізують наскрізні обов'язки — автентифікацію та авторизацію — шляхом перехоплення запитів до їх передачі контролеру. Такий підхід забезпечує принцип єдиної відповідальності: контролер займається виключно бізнес-логікою, не перевіряючи токени або ролі вручну [31].

Клієнтська частина організована за компонентною архітектурою React з односпрямованим потоком даних [15]. Спільний стан, доступний у всьому застосунку — зокрема, дані авторизованого користувача — зберігається в контексті (AuthContext) і не передається через props через кілька рівнів компонентів. Стан окремих сторінок і їхніх дочірніх компонентів є локальним і управляється хуками useState та useEffect. Запити до API

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

інкапсульовані в окремих модулях з централізованим перехоплювачем Axios, який автоматично додає JWT-токен до заголовку Authorization кожного запиту — це усуває необхідність повторення цього коду в кожному компоненті [9].

Архітектура розгортання (deployment architecture) передбачає три незалежні хмарні компоненти: база даних PostgreSQL на Supabase, серверний застосунок на Render та клієнтський застосунок на Netlify. Така розподілена конфігурація має ряд переваг: незалежне масштабування кожного компонента, ізоляція збоїв (відмова одного компонента не призводить до недоступності інших), а також можливість оновлення кожного компонента незалежно від решти. Комунікація між компонентами здійснюється через публічні HTTPS-ендпоінти з налаштуванням CORS для обмеження дозволених джерел запитів.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментальних засобів реалізації платформи здійснювався з урахуванням вимог до продуктивності, типобезпечності, зрілості екосистеми та доступності безкоштовних хмарних рішень для розгортання.

Основною мовою програмування обрано TypeScript — статично типізований надмножина JavaScript, що дозволяє виявляти помилки на етапі компіляції, покращує читабельність коду та забезпечує надійнішу розробку великих застосунків [13]. TypeScript використовується як на серверній, так і на клієнтській стороні, що забезпечує єдиний технологічний стек та можливість повторного використання типів [17].

Node.js обрано як середовище виконання серверного коду завдяки його асинхронній однопоточній моделі, яка є ефективною для обробки великої кількості одночасних з'єднань — що є важливим для реалізації WebSocket-сервера [14]. Фреймворк Express.js надає мінімалістичний, але потужний інструментарій для побудови REST API з підтримкою middleware для автентифікації, обробки помилок та валідації запитів.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

Для роботи з базою даних обрано PostgreSQL — надійну реляційну СКБД з відкритим вихідним кодом, що підтримує складні запити, транзакції та зовнішні ключі [10][12]. Доступ до бази даних здійснюється через ORM Prisma, яка генерує типобезпечний клієнт на основі декларативної схеми та забезпечує зручне управління міграціями. База даних розгорнута на хмарній платформі Supabase, що надає безкоштовний хостинг PostgreSQL з вбудованим сховищем файлів для зберігання аватарів користувачів [21].

На клієнтській стороні використовується бібліотека React версії 18 — найпоширеніше рішення для побудови компонентних інтерфейсів користувача [16]. Збирач Vite забезпечує швидку збірку проекту та гарячу заміну модулів під час розробки. Для маршрутизації використовується React Router, для HTTP-запитів — бібліотека Axios, для реалізації WebSocket-клієнта — бібліотека Socket.IO [22].

Автентифікація користувачів реалізована на основі JSON Web Tokens з підтримкою входу через обліковий запис Google за протоколом OAuth 2.0, що забезпечує безпечний та зручний механізм ідентифікації. Для відправки електронних листів використовується сервіс Resend, який надає HTTP API для доставки листів без обмежень хостингового середовища [23].

Розгортання системи здійснено в хмарному середовищі: серверна частина розгорнута на платформі Render, клієнтська — на платформі Netlify. Обидві платформи надають безкоштовні тарифні плани, достатні для демонстрації та тестування системи.

Вибір TypeScript як основної мови програмування обумовлений не лише статичною типізацією, а й конкретними практичними перевагами при розробці даної системи. Типізація дозволила визначити загальні типи для об'єктів предметної галузі — наприклад, тип User із переліком полів та їх типів, тип Lesson із обов'язковими та необов'язковими полями — і використовувати ці визначення як на сервері (при формуванні відповідей API), так і на клієнті (при відображенні даних). Завдяки TypeScript компілятор автоматично виявляє невідповідності між очікуваною

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

структурою відповіді API та фактичною структурою об'єкта, що значно скорочує час відлагодження. Prisma Client, згенерований на основі схеми бази даних, автоматично типізує всі запити та їх результати — це виключає можливість звернення до неіснуючого поля або передачі некоректного типу значення.

Node.js як середовище виконання серверного коду демонструє ефективність саме при обробці великої кількості конкурентних з'єднань з переважаючою часткою операцій введення-виведення над обчислювальними операціями. Серверна частина платформи виконує значну кількість асинхронних операцій: запити до бази даних, відправка електронних листів через API Resend, завантаження файлів до Supabase Storage, WebSocket-повідомлення. Асинхронна модель Node.js з циклом подій (event loop) дозволяє обробляти всі ці операції конкурентно без блокування потоку виконання в очікуванні відповіді від зовнішніх сервісів [18].

Express.js надає мінімалістичний, але потужний інструментарій для маршрутизації HTTP-запитів і побудови middleware-ланцюжків. Порівняно з альтернативами — Fastify або Koa — Express.js має найширшу екосистему готових middleware-пакетів і є найбільш документованим Node.js-фреймворком, що прискорює пошук рішень при виникненні нетипових задач. Функціональність CORS, парсингу JSON-тіла запитів, валідації та обробки помилок реалізована через middleware і підключається декларативно, не захаращуючи логіку контролерів.

Prisma ORM обрана замість прямих SQL-запитів або альтернативних ORM (TypeORM, Sequelize) з ряду причин [20]. По-перше, декларативна схема Prisma є єдиним джерелом правди для структури бази даних — з неї автоматично генерується типізований клієнт для TypeScript, документація схеми та SQL-міграції. По-друге, Prisma підтримує сучасний синтаксис запитів із вкладеними відносинами через include та select, що дозволяє отримати об'єкт заняття разом із даними викладача, учня, оцінок та домашніх завдань за один запит до бази даних. По-третє, Prisma Migrate забезпечує

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

версіонований контроль змін схеми з автоматичним генеруванням та застосуванням SQL-міграцій.

Supabase Storage як сховище для аватарів користувачів обраний через нативну інтеграцію з хмарною PostgreSQL-базою даних на тій же платформі [21]. Supabase надає JavaScript SDK для завантаження файлів, генерації публічних URL та управління правами доступу через Policies. Це дозволило реалізувати завантаження аватара безпосередньо з браузера на сховище без проксіювання через серверну частину, що зменшує навантаження на сервер та прискорює процес завантаження файлу.

Сервіс Resend обраний для відправки електронних листів замість налаштування власного SMTP-сервера або використання бібліотеки nodemailer. SMTP-конфігурація є ненадійним рішенням у хмарному середовищі — більшість провайдерів хостингу блокують вихідні з'єднання на порт 25, а конфігурація зовнішнього SMTP нерідко призводить до проблем з таймаутами при недоступності сервера. Resend надає HTTP REST API для відправки листів із гарантованою доставкою, детальними логами та зручною вебпанеллю для моніторингу, що є значно надійнішим рішенням для продакшн-середовища [23].

Vite обраний як збирач клієнтської частини завдяки значно вищій швидкості запуску сервера розробки порівняно з Webpack або Create React App — на відміну від традиційних збирачів, Vite не виконує попередню збірку всього коду при старті, а використовує рідні ES-модулі браузера для обслуговування файлів під час розробки, перетворюючи лише запитувані модулі в режимі реального часу. Виробнича збірка виконується через Rollup, що генерує оптимізовані та мінімізовані бандли.

Мінімальні вимоги до апаратного забезпечення для кінцевого користувача: комп'ютер або мобільний пристрій з сучасним браузером (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+), підключення до мережі Інтернет зі швидкістю не менше 1 Мбіт/с. Серверна частина розгорнута в хмарі та не потребує локального апаратного забезпечення від користувача.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 1

У першому розділі проведено аналіз предметної галузі та сформульовано задачу розробки вебплатформи для організації навчального процесу онлайн-школи. Виявлено ключові проблеми традиційних методів управління навчанням та визначено функціональні вимоги до системи. Проведено порівняльний аналіз існуючих аналогів — Google Classroom, Moodle та Coursera — та встановлено, що жодна з розглянутих платформ не задовольняє повною мірою потреби приватної онлайн-школи з персоналізованим навчанням. Обґрунтовано вибір клієнт-серверної архітектури на базі REST API та WebSocket, визначено інструментальні засоби реалізації: TypeScript, Node.js, Express.js, PostgreSQL, Prisma, React та Vite.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПЛАТФОРМИ

2.1. Визначення варіантів використання та рольова модель системи

Проектування системи розпочалося з визначення акторів та варіантів використання [27]. Система передбачає чотири типи користувачів: адміністратор, куратор, викладач та учень. Кожен актор має чітко визначений набір доступних функцій, що реалізується через механізм рольового контролю доступу на рівні серверної частини.

Адміністратор є головним керуючим суб'єктом системи. Він здійснює управління обліковими записами всіх користувачів — створення, редагування, блокування та видалення. Адміністратор розглядає вхідні заявки від нових користувачів, призначає дату та час вступної співбесіди, надсилає запрошення на співбесіду електронною поштою, а також підтверджує або відхиляє заявку за результатами співбесіди. Крім того, адміністратор призначає кураторів до учнів та має доступ до загальної статистики платформи.

Куратор відповідає за супровід закріплених за ним учнів. Він переглядає профілі та прогрес навчання своїх учнів, шукає викладачів за предметом та призначає їх до учнів, а також отримує автоматичні сповіщення у разі різкого зниження успішності підопічного. Куратор має доступ до вбудованого чату для комунікації з учнями, викладачами та адміністрацією.

Викладач управляє власним розкладом занять — створює одноразові та повторювані щотижневі заняття, редагує та скасовує їх. Він виставляє оцінки учням безпосередньо в картці заняття, додає домашні завдання та перевіряє їх виконання. Викладач має доступ до профілю кожного свого учня та бачить прогрес навчання в розрізі власних занять. Також викладач може налаштувати постійне посилання на відеоконференцію, яке автоматично відображається в деталях кожного заняття.

Учень переглядає власний розклад занять у форматі тижневого або списочного вигляду, бачить виставлені оцінки та домашні завдання, надсилає

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижморя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

виконані завдання на перевірку. Учень має доступ до власного профілю з відображенням загального прогресу навчання та може спілкуватись з викладачами і куратором через вбудований чат.

Діаграма варіантів використання системи відображає взаємодію всіх чотирьох акторів із функціональними можливостями платформи та наводиться у додатках до роботи (Додаток А) [29].

Для детальнішого розуміння функціональних можливостей системи розглянемо варіанти використання кожного актора в контексті типових сценаріїв взаємодії з платформою.

Типовий сценарій роботи адміністратора починається з перегляду списку вхідних заявок на вступ на вкладці «Заявки» сторінки адміністрування. Для кожної заявки адміністратор бачить ім'я заявника, вказану електронну пошту, бажану роль та мотивацію. Якщо заявник претендує на роль учня або викладача, адміністратор призначає дату та час співбесіди і надсилає автоматичне запрошення електронною поштою через сервіс Resend з посиланням на відеоконференцію. Посилання на відеоконференцію беруть із поля meetingLink профілю адміністратора або зазначаються безпосередньо при надсиланні запрошення. Після проведення співбесіди адміністратор підтверджує заявку — у цей момент обліковий запис активується (поле isActive стає true) і новий користувач отримує листа з підтвердженням та посиланням для входу. За потреби адміністратор може відхилити заявку — тоді заявнику надсилається лист із відповідним повідомленням, а обліковий запис не активується.

Після підтвердження нового учня адміністратор переходить до вкладки «Куратори» для призначення відповідального куратора. Вкладка відображає список усіх кураторів із кількістю закріплених за кожним учнів, що дозволяє розподіляти навантаження рівномірно. Призначення здійснюється шляхом зміни поля curatorId у профілі учня. Адміністратор також керує обліковими записами на вкладці «Користувачі» — може знайти

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

будь-якого користувача за ім'ям або відфільтрувати за роллю, заблокувати обліковий запис при порушенні правил або видалити його повністю.

Типовий сценарій роботи куратора включає щоденний моніторинг прогресу підопічних учнів. Куратор переглядає список закріплених учнів з індикаторами їхнього поточного прогресу — відсотком успішності, середнім тематичним балом та тенденцією. При виявленні учня з негативною тенденцією куратор може переглянути детальний профіль та звернутись до відповідного викладача через вбудований чат. При потребі призначити нового викладача куратор використовує функцію пошуку за предметом — система відображає список викладачів із зазначенням предмету, досвіду та напрямів підготовки. Після вибору відповідного викладача куратор призначає його до учня, і між ними автоматично створюється зв'язок у таблиці StudentTeacher. При різкому падінні успішності — якщо шість останніх оцінок учня від одного викладача є нижчими за 50 балів — куратор отримує автоматичне сповіщення в режимі реального часу через WebSocket.

Типовий сценарій роботи викладача в рамках одного тижня охоплює кілька регулярних активностей. На початку тижня викладач переглядає розклад у тижневому вигляді та переконується у відсутності конфліктів. Перед кожним заняттям він відкриває картку заняття для ознайомлення з темою та перегляду попередніх оцінок учня. Після проведення заняття викладач виставляє поточну або тематичну оцінку в картці заняття, а також може додати домашнє завдання з описом і дедлайном. Наприкінці тижня викладач перевіряє надіслані учнями виконані домашні завдання та виставляє оцінки за них. При необхідності скорегувати розклад наступного тижня викладач може перемістити заняття перетягуванням на нову клітинку тижневого календаря або відредагувати час у формі редагування — в обох випадках система автоматично перевіряє відсутність конфліктів.

Типовий сценарій роботи учня передбачає щоденний перегляд розкладу і домашніх завдань. Учень відкриває сторінку розкладу, щоб дізнатись час наступного заняття та посилання на відеоконференцію

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

викладача. Після отримання домашнього завдання учень переходить на сторінку HomeworkPage, знаходить завдання та надсилає виконану роботу у вигляді текстового опису або посилання на файл. Учень може переглянути власний профіль із показниками загального прогресу навчання — відсотком успішності та середнім тематичним балом, розрахованими на основі всіх отриманих оцінок. Для спілкування з викладачем або куратором учень відкриває сторінку чату та вибирає відповідний контакт зі списку доступних [32].

Механізм рольового контролю доступу реалізований на рівні серверного middleware. Після перевірки JWT-токена в authMiddleware наступним у ланцюжку йде roleMiddleware, який порівнює роль, збережену в токені, із переліком дозволених ролей для конкретного маршруту. Якщо роль користувача відсутня в переліку дозволених, сервер повертає відповідь зі статусом 403 Forbidden без виконання бізнес-логіки контролера. Така реалізація є надійнішою за перевірку ролей безпосередньо в контролерах, оскільки захист застосовується декларативно на рівні визначення маршрутів і не може бути випадково пропущений.

2.2. Розробка бази даних системи

Структура бази даних розроблялася відповідно до функціональних вимог системи та принципів нормалізації реляційних баз даних. База даних реалізована в СКБД PostgreSQL та описана через декларативну схему ORM Prisma (Додаток Д).

Центральною сутністю системи є таблиця User, яка зберігає дані про всіх користувачів незалежно від їхньої ролі. Таблиця містить поля ідентифікатора, електронної пошти, хешованого паролю, повного імені, ролі користувача (ADMIN, CURATOR, TEACHER, STUDENT), прапорця активності облікового запису, URL-адреси аватара, біографії, мотивації, контактних даних (телефон, Telegram, Viber, Signal), а також специфічних для викладача полів — предмету викладання, напрямів підготовки, освітніх програм, методів викладання, досвіду та посилання на відеоконференцію.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Така денормалізація таблиці користувачів є свідомим архітектурним рішенням, що спрощує запити та зменшує кількість об'єднань таблиць для найбільш частих операцій.

Зв'язок між куратором та учнем реалізовано через зовнішній ключ `curatorId` у таблиці `User`, що відповідає відношенню один-до-багатьох. Зв'язок між учнем та викладачами реалізовано через проміжну таблицю `StudentTeacher`, що забезпечує відношення багато-до-багатьох — один учень може навчатись у кількох викладачів одночасно.

Таблиця `Lesson` зберігає інформацію про заняття та містить поля заголовку, теми, дати та часу початку, тривалості в хвилинах, статусу (скасоване або активне), а також зовнішні ключі на викладача та учня. Для підтримки повторюваних щотижневих занять передбачено поле `groupId`, яке об'єднує пов'язані заняття в групу — це дозволяє редагувати або видаляти одне заняття, всі майбутні або всі заняття групи одночасно.

Таблиця `Grade` зберігає оцінки учнів та пов'язана з таблицями `Lesson` і `User` через зовнішні ключі на заняття, учня та викладача. Тип оцінки визначається полем `type` зі значеннями `CURRENT` (поточна) та `THEMATIC` (тематична), що дозволяє розраховувати окремі показники успішності для різних типів контролю.

Таблиця `Homework` описує домашні завдання, прив'язані до конкретного заняття. Таблиця `HomeworkSubmission` зберігає відповіді учнів на завдання з можливістю прикріплення текстового коментаря та посилання на файл. Таблиця `Progress` зберігає агрегований прогрес навчання учня — відсоток успішності, середній тематичний бал та тенденцію (зростає, знижується, стабільна) — і оновлюється автоматично при кожній зміні оцінок.

Таблиця `Message` зберігає повідомлення чату та містить посилання на відправника, отримувача та тип чату (учень-викладач, учень-куратор, викладач-куратор тощо). Таблиця `Notification` зберігає сповіщення для користувачів з типом події, текстом та посиланням для переходу. Таблиця

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

ApplicationRequest зберігає заявки нових користувачів на вступ із зазначенням ролі та мотивації.

Нижче подано детальний опис кожної сутності бази даних у форматі таблиць атрибутів.

Структуру таблиці користувачів наведено в табл. 2.1. Вона є центральною сутністю системи та забезпечує зберігання персональних, контактних та рольових даних усіх користувачів вебплатформи.

Таблиця 2.1

Опис таблиці — User (Користувач)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор
email	String	UNIQUE, NOT NULL	Електронна пошта
passwordHash	String	NOT NULL	Хеш паролю (bcrypt)
fullName	String	NOT NULL	Повне ім'я користувача
role	String	NOT NULL, default "STUDENT"	Роль: ADMIN, CURATOR, TEACHER, STUDENT
isActive	Boolean	NOT NULL, default false	Чи активований обліковий запис
createdAt	DateTime	NOT NULL, default now()	Дата реєстрації
updatedAt	DateTime	NOT NULL, auto	Дата останнього оновлення

avatarUrl	String?	NULL	URL аватара в Supabase Storage
bio	String?	NULL	Біографія користувача
motivation	String?	NULL	Мотивація (заповнюється при заявці)
phone	String?	NULL	Номер телефону
telegramUsername	String?	NULL	Нікнейм у Telegram
viberUsername	String?	NULL	Нікнейм у Viber
signalUsername	String?	NULL	Нікнейм у Signal
subject	String?	NULL	Предмет викладання (лише для TEACHER)
teachingDirections	String?	NULL	Напрями підготовки (лише для TEACHER)
educationalPrograms	String?	NULL	Освітні програми (лише для TEACHER)
teachingMethods	String?	NULL	Методи викладання (лише для TEACHER)
experience	Int?	NULL	Досвід у роках (лише для TEACHER)
meetingLink	String?	NULL	Посилання на відеоконференцію
googleId	String?	UNIQUE, NULL	Ідентифікатор Google OAuth

interviewDate	String?	NULL	Дата призначеної співбесіди
interviewTime	String?	NULL	Час призначеної співбесіди
interviewLink	String?	NULL	Посилання на відеоконференцію співбесіди
curatorId	Int?	FK → User.id, NULL	Ідентифікатор куратора (лише для STUDENT)

Структуру проміжної таблиці StudentTeacher наведено в табл. 2.2. Дана таблиця реалізує зв'язок багато-до-багатьох між учнями та викладачами, забезпечуючи можливість навчання одного учня у декількох викладачів. При видаленні користувача записи видаляються каскадно.

Таблиця 2.2

Опис таблиці — StudentTeacher (Зв'язок учень–викладач)

Атрибут	Тип	Обмеження	Опис
studentId	Int	PK (частина), FK → User.id, CASCADE	Ідентифікатор учня
teacherId	Int	PK (частина), FK → User.id, CASCADE	Ідентифікатор викладача

Структуру таблиці Lesson наведено в табл. 2.3. Таблиця призначена для зберігання інформації про навчальні заняття, їхній розклад, тривалість та зв'язки з учнями і викладачами.

Таблиця 2.3

Опис таблиці — Lesson (Заняття)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор заняття

продовж. табл. 2.3

title	String	NOT NULL	Назва заняття
topic	String	NOT NULL	Тема заняття
date	DateTime	NOT NULL	Дата і час початку
duration	Int	NOT NULL	Тривалість у хвилинах
status	String?	NULL	Статус: CANCELLED або NULL (активне)
teacherId	Int	NOT NULL, FK → User.id	Ідентифікатор викладача
studentId	Int	NOT NULL, FK → User.id	Ідентифікатор учня
createdAt	DateTime	NOT NULL, default now()	Дата створення запису
updatedAt	DateTime	NOT NULL, auto	Дата оновлення запису
groupId	String?	NULL	Ідентифікатор групи повторюваних занять (UUID)

Структуру таблиці Grade наведено в табл. 2.4. Таблиця забезпечує зберігання результатів оцінювання учнів та підтримує різні типи контролю знань.

Таблиця 2.4

Опис таблиці — Grade (Оцінка)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор оцінки

value	Float	NOT NULL	Значення оцінки (числове)
type	String	NOT NULL, default "CURRENT"	Тип: CURRENT або THEMATIC
comment	String?	NULL	Коментар викладача до оцінки
date	DateTime	NOT NULL, default now()	Дата виставлення оцінки
studentId	Int	NOT NULL, FK → User.id	Ідентифікатор учня
lessonId	Int	NOT NULL, FK → Lesson.id, CASCADE	Ідентифікатор заняття
teacherId	Int	NOT NULL, FK → User.id	Ідентифікатор викладача
homeworkId	Int?	UNIQUE, NULL, FK → Homework.id	Ідентифікатор ДЗ (якщо оцінка за ДЗ)

Структуру таблиці LessonMaterial наведено в табл. 2.5. Таблиця використовується для зберігання навчальних матеріалів, які прикріплюються до конкретних занять.

Таблиця 2.5

Опис таблиці — LessonMaterial (Матеріал до заняття)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор матеріалу
title	String	NOT NULL	Назва матеріалу

type	String	NOT NULL	Тип матеріалу (документ, посилання тощо)
fileUrl	String	NOT NULL	URL файлу або посилання
description	String?	NULL	Опис матеріалу
uploadDate	DateTime	NOT NULL, default now()	Дата завантаження
lessonId	Int	NOT NULL, FK → Lesson.id, CASCADE	Ідентифікатор заняття

Структуру таблиці Homework наведено в табл. 2.6. Таблиця призначена для зберігання домашніх завдань, що видаються учням у межах проведених занять.

Таблиця 2.6

Опис таблиці — Homework (Домашнє завдання)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор
taskDescription	String	NOT NULL	Опис завдання
deadline	DateTime	NOT NULL	Термін здачі
attachments	String	NOT NULL, default ""	Прикріплення (рядок з посиланнями)
status	String	NOT NULL, default "NOT_STARTED"	Статус: NOT_STARTED, SUBMITTED тощо
lessonId	Int	NOT NULL, FK → Lesson.id, CASCADE	Ідентифікатор заняття

Структуру таблиці HomeworkSubmission наведено в табл. 2.7. Таблиця забезпечує зберігання відповідей учнів на домашні завдання та інформації про їхній статус виконання.

Таблиця 2.7

Опис таблиці — HomeworkSubmission (Відповідь учня на ДЗ)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор
homeworkId	Int	NOT NULL, FK → Homework.id, CASCADE	Ідентифікатор домашнього завдання
studentId	Int	NOT NULL, FK → User.id	Ідентифікатор учня
text	String?	NULL	Текстова відповідь учня
fileUrl	String?	NULL	Посилання на файл з відповіддю
status	String	NOT NULL, default "DONE_NO_QUESTIONS"	Статус відповіді
submittedAt	DateTime	NOT NULL, default now()	Дата і час подачі

Структуру таблиці Progress наведено в табл. 2.8. Таблиця використовується для зберігання агрегованих показників успішності та відстеження динаміки навчальних досягнень учня.

Таблиця 2.8

Опис таблиці — Progress (Прогрес навчання учня)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор

successPercentage	Float	NOT NULL	Загальний відсоток успішності
averageThematicGrade	Float	NOT NULL	Середній тематичний бал
trend	String	NOT NULL, default "STABLE"	Тенденція: GROWING, DECLINING, STABLE
lastUpdated	DateTime	NOT NULL, default now()	Дата останнього перерахунку
studentId	Int	UNIQUE, NOT NULL, FK → User.id	Ідентифікатор учня (1:1 з User)

Структуру таблиці Message наведено в табл. 2.9. Таблиця забезпечує зберігання повідомлень між користувачами та реалізує функціональність внутрішнього чату системи.

Таблиця 2.9

Опис таблиці — Message (Повідомлення чату)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор
text	String	NOT NULL	Текст повідомлення
sentDate	DateTime	NOT NULL, default now()	Дата і час відправлення
isRead	Boolean	NOT NULL, default false	Чи прочитане повідомлення
chatType	String	NOT NULL	Тип чату (student-teacher, student-curator тощо)

senderId	Int	NOT NULL, FK → User.id	Ідентифікатор відправника
receiverId	Int	NOT NULL, FK → User.id	Ідентифікатор отримувача

Структуру таблиці Notification наведено в табл. 2.10. Таблиця призначена для зберігання сповіщень користувачів про події, що відбуваються в системі.

Таблиця 2.10

Опис таблиці — Notification (Сповіщення)

Атрибут	Тип	Обмеження	Опис
id	Int	PK, autoincrement	Унікальний ідентифікатор
type	String	NOT NULL	Тип події сповіщення
text	String	NOT NULL	Текст сповіщення
date	DateTime	NOT NULL, default now()	Дата і час створення
isViewed	Boolean	NOT NULL, default false	Чи переглянуте сповіщення
targetUrl	String?	NULL	Посилання для переходу за сповіщенням
userId	Int	NOT NULL, FK → User.id	Ідентифікатор отримувача

Фізична схема БД наводиться у додатках до роботи (Додаток Б).

2.3. Проектування алгоритмів роботи системи

Ключовим алгоритмом системи є процес автентифікації та авторизації користувачів. При вході через електронну пошту та пароль сервер перевіряє наявність користувача в базі даних, порівнює хеш введеного паролю з збереженим за допомогою бібліотеки bcrypt, перевіряє статус активності облікового запису та генерує JSON Web Token з вбудованим ідентифікатором

та роллю користувача. Токен передається клієнту та зберігається в localStorage для використання в наступних запитах. При вході через Google OAuth 2.0 сервер отримує профіль користувача від Google, перевіряє наявність облікового запису з відповідною електронною поштою та або автентифікує існуючого користувача, або створює новий обліковий запис зі статусом очікування підтвердження. Кожен захищений запит до API проходить через middleware автентифікації, який верифікує токен та додає дані користувача до об'єкта запиту [9] (Додаток В).

```
export const authMiddleware = (req: AuthRequest, res: Response, next:
NextFunction) => {
  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res.status(401).json({ error: 'No token provided' });
  }

  const token = authHeader.split(' ')[1];

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET!) as JwtPayload;
    req.user = decoded;
    next();
  } catch (error) {
    return res.status(401).json({ error: 'Invalid token' });
  }
};
```

Алгоритм управління розкладом занять реалізує перевірку конфліктів часу при створенні та редагуванні занять. Перед збереженням нового заняття система отримує всі активні заняття викладача та перевіряє перетин часових інтервалів — якщо новий інтервал перетинається з будь-яким існуючим, повертається помилка з відповідним повідомленням. При створенні повторюваного заняття перевірка виконується для кожного тижня окремо. Статус заняття розраховується динамічно на основі поточного часу та збереженого статусу — заняття може мати статус запланованого, активного (якщо поточний час знаходиться в межах тривалості заняття), завершеного або скасованого (Додаток В).

Алгоритм розрахунку прогресу учня запускається автоматично при кожній зміні оцінок. Система отримує всі оцінки учня, впорядковані за

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

датою, та розраховує загальний відсоток успішності як середнє значення всіх оцінок, середній тематичний бал на основі оцінок типу THEMATIC, а також тенденцію навчання шляхом порівняння середнього балу першої та другої половини всіх оцінок. Для моніторингу різкого зниження успішності система перевіряє чи не містять останні шість оцінок учня від одного викладача значень нижче п'ятдесяти балів — у такому разі куратору надсилається автоматичне сповіщення із зазначенням імені викладача та предмету.

```
export async function updateStudentProgress(studentId: number) {
  try {
    const grades = await prisma.grade.findMany({
      where: { studentId },
      orderBy: { date: 'asc' },
      select: { value: true, type: true, teacherId: true, date: true },
    });

    const result = calcProgress(grades);
    if (!result) return;

    await prisma.progress.upsert({
      where: { studentId },
      update: { ...result, lastUpdated: new Date() },
      create: { studentId, ...result },
    });
  }
}
```

Після збереження агрегованих показників прогресу функція переходить до перевірки тенденції зниження успішності в розрізі кожного викладача. Оцінки групуються за ідентифікатором викладача за допомогою структури Map, що дозволяє ізолювати оцінки кожного педагога в окремий масив. Відповідний фрагмент наведено нижче:

```
const gradesByTeacher = new Map<number, typeof grades>();
for (const g of grades) {
  if (!g.teacherId) continue;
  if (!gradesByTeacher.has(g.teacherId)) gradesByTeacher.set(g.teacherId,
    []);
  gradesByTeacher.get(g.teacherId)!.push(g);
}
```

Після формування груп для кожної з них аналізуються останні шість оцінок. Якщо всі вони є нижчими за 50 балів, виконуються паралельні запити до бази даних для отримання імені учня та даних викладача через Promise.all, що мінімізує час очікування. Якщо куратор учня існує — перевіряється

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

наявність аналогічного сповіщення за останні 24 години. Відповідний фрагмент наведено нижче:

```
for (const [teacherId, teacherGrades] of gradesByTeacher) {
  const lastSix = teacherGrades.slice(-6);
  if (lastSix.length < 6 || !lastSix.every(g => g.value < 50)) continue;

  const [student, teacher] = await Promise.all([
    prisma.user.findUnique({
      where: { id: studentId },
      select: { fullName: true, curatorId: true },
    }),
    prisma.user.findUnique({
      where: { id: teacherId },
      select: { fullName: true, subject: true },
    }),
  ]);

  if (!student?.curatorId) continue;
```

Якщо аналогічне сповіщення за останні 24 години відсутнє, алгоритм формує текст повідомлення, що містить ім'я учня, ім'я викладача та, за наявності відповідних даних, назву предмету викладання — це дозволяє куратору одразу зрозуміти суть проблеми без необхідності переходити до додаткових екранів системи. Сформоване сповіщення зберігається в базі даних для подальшого перегляду в історії сповіщень, а також одночасно доставляється куратору в реальному часі через WebSocket-з'єднання, завдяки чому він отримує сповіщення без потреби оновлювати сторінку чи очікувати на періодичну перевірку. Усі операції алгоритму обгорнуті в блок обробки винятків, що дозволяє системі коректно завершувати виконання навіть у разі непередбачуваної помилки бази даних, не перериваючи роботу інших залежних процесів. Завершальний фрагмент функції наведено нижче:

```
const recentNotif = await prisma.notification.findFirst({
  where: {
    userId: student.curatorId,
    type: 'DECLINING_TREND',
    text: { contains: teacher?.fullName || '' },
    date: { gte: new Date(Date.now() - 1000 * 60 * 60 * 24) },
  },
});

if (!recentNotif) {
  const teacherInfo = teacher
```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    ? `${teacher.fullName}${teacher.subject ? ` (${teacher.subject})` :
    : 'невідомий викладач';

    await createNotification(
      student.curatorId,
      'DECLINING_TREND',
      `⚠️ Учень ${student.fullName} отримав 6 оцінок підряд нижче 50 балів
у викладача ${teacherInfo}`,
      '/my-students',
    );
  }
} catch (e) {
  console.error('updateStudentProgress error:', e);
}
}

```

Система реального часу реалізована на базі Socket.IO. При підключенні клієнта сервер автентифікує WebSocket-з'єднання за JWT-токеном та реєструє користувача в персональній кімнаті за його ідентифікатором. Повідомлення чату зберігаються в базі даних та одночасно транслюються отримувачу через WebSocket. Сповіщення про нові заняття, оцінки та зміни в розкладі також доставляються через WebSocket у реальному часі [22].

Функція buildChatGroups реалізує різну логіку формування груп чату залежно від ролі поточного користувача. Для викладача формується список груп, де кожна група відповідає одному учню та додатково містить куратора цього учня за наявності. Для учня визначаються контакти куратора та всіх призначених викладачів як окремі групи. Початковий фрагмент функції наведено нижче:

```

export const buildChatGroups = async (
  contacts: Contact[],
  userRole: string,
): Promise<ChatGroup[]> => {
  if (userRole === 'TEACHER') {
    const groups: ChatGroup[] = [];
    const admins = contacts.filter(c => c.role === 'ADMIN');
    if (admins.length > 0) groups.push({ label: 'Адміністратор', contacts:
admins });

    const studentsRes = await axiosClient.get('/users/teacher/all-
students').catch(() => ({ data: [] }));
    const students: any[] = studentsRes.data;

```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    for (const student of students) {
      const groupContacts: Contact[] = [student as Contact];
      if (student.curatorTeacher) groupContacts.push(student.curatorTeacher as
Contact);
      groups.push({ label: `Група за учнем ${student.fullName}`, studentId:
student.id, contacts: groupContacts });
    }
    return groups;
  }
}

```

Для ролі STUDENT функція звертається до окремого ендпоінту API, який повертає інформацію про призначеного куратора та повний список викладачів, що ведуть заняття з цим учнем. На відміну від гілки для викладача, де групи формуються динамічно за кожним учнем, для учня контакти агрегуються за категоріями — куратор виокремлюється в одну групу, а всі призначені викладачі об'єднуються в іншу групу під загальною назвою «Мої викладачі», оскільки учневі зазвичай не потрібно розрізняти викладачів за окремими групами так само детально, як це потрібно викладачу стосовно учнів. Якщо учневі взагалі не призначено жодного контакту (наприклад, на етапі, коли розклад ще не сформовано), функція повертає порожній масив груп, що дозволяє інтерфейсу коректно відобразити відсутність доступних співрозмовників. Відповідний фрагмент наведено нижче:

```

if (userRole === 'STUDENT') {
  const assignedRes = await axiosClient
    .get('/users/student/assigned-contacts')
    .catch(() => ({ data: { curator: null, teachers: [] } }));
  const { curator, teachers: assignedTeachers } = assignedRes.data as {
curator: Contact | null; teachers: Contact[] };
  const groups: ChatGroup[] = [];
  if (curator) {
    groups.push({ label: 'Куратор', contacts: [curator] });
  }
  if (assignedTeachers.length > 0) {
    groups.push({ label: 'Мої викладачі', contacts: assignedTeachers });
  }
  if (groups.length === 0) return [];
  return groups;
}

```

Для ролі CURATOR логіка є помітно складнішою, оскільки куратор одночасно взаємодіє з кількома підопічними учнями та з усіма викладачами кожного з них. Для кожного підопічного учня формується окрема група, до

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

якої включається сам учень та всі його викладачі, що дозволяє куратору вести окреме листування в контексті кожного учня без змішування повідомлень про різних дітей в одному потоці. З метою оптимізації кількості звернень до API, викладачі спочатку шукаються в локальній структурі Map, що вже містить раніше завантажені контакти куратора, і лише за відсутності потрібного викладача в цій структурі виконується додатковий запит до ендпоінту профілю користувача — такий підхід дозволяє уникнути зайвих мережових запитів для викладачів, дані про яких уже наявні на клієнті. Відповідний фрагмент наведено нижче:

```
if (userRole === 'CURATOR') {
  const studentsRes = await
axiosClient.get('/users/curator/students').catch(() => ({ data: [] }));
  const students: any[] = studentsRes.data;
  const allContactsMap = new Map<number, Contact>(contacts.map(c => [c.id,
c]));
  const groups: ChatGroup[] = [];
  const admins = contacts.filter(c => c.role === 'ADMIN');
  if (admins.length > 0) groups.push({ label: 'Адміністратор', contacts:
admins });
}
```

Перед тим як формувати групи за учнями, усі вже наявні контакти куратора заздалегідь складаються у спеціальну структуру (Map), що дозволяє швидко знаходити потрібного викладача за його ідентифікатором — без потреби кожного разу переглядати весь список контактів заново. Завдяки цьому пошук залишається швидким навіть тоді, коли в куратора багато підопічних учнів і викладачів. Окремо, ще до основного циклу, формується група «Адміністратор» — оскільки адміністратори не прив'язані до конкретного учня, їх виносять у самостійну групу одразу, а не обробляють у логіці, що стосується учнів. Лише після цього алгоритм переходить до побудови окремої групи для кожного підопічного учня куратора:

```
for (const student of students) {
  const groupContacts: Contact[] = [student as Contact];
  for (const st of (student.studentTeachers || [])) {
    const teacher = allContactsMap.get(st.teacher.id);
    if (teacher) {
      groupContacts.push(teacher);
    } else {
      const tRes = await
axiosClient.get(`/profile/${st.teacher.id}`).catch(() => null);
    }
  }
}
```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (tRes?.data) groupContacts.push(tRes.data as Contact);
    }
}
groups.push({ label: `Група за учнем ${student.fullName}`, studentId:
student.id, contacts: groupContacts });
}
return groups;
}

```

Завершальна гілка функції обробляє роль ADMIN. Адміністратор не має прив'язки до конкретних учнів, тому його список контактів формується простою фільтрацією за роллю — окремо виділяються всі куратори та всі викладачі. Якщо роль не відповідає жодному з визначених варіантів, функція повертає єдину групу з усіма контактами як резервний варіант. Завершальний фрагмент функції наведено нижче:

```

if (userRole === 'ADMIN') {
    const teachers = contacts.filter(c => c.role === 'TEACHER');
    const curators = contacts.filter(c => c.role === 'CURATOR');
    const result: ChatGroup[] = [];
    if (curators.length > 0) result.push({ label: 'Куратори', contacts:
curators });
    if (teachers.length > 0) result.push({ label: 'Викладачі', contacts:
teachers });
    return result;
}

return [{ label: 'Всі контакти', contacts }];
};

```

2.4. Реалізація програмного комплексу та компонентна будова додатку

Серверна частина організована за принципом розподілу відповідальності між окремими модулями. Точка входу застосунку — файл app.ts — ініціалізує Express-застосунок, налаштовує CORS із підтримкою переліку дозволених джерел через змінну середовища CORS_ORIGIN, підключає middleware для парсингу JSON-тіла запитів та реєструє маршрути для кожного функціонального модуля:

```

import express from 'express';
import cors from 'cors';
import dotenv from 'dotenv';
import authRoutes from './routes/authRoutes';
import lessonRoutes from './routes/lessonRoutes';
import gradeRoutes from './routes/gradeRoutes';
import userRoutes from './routes/userRoutes';

```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import messageRoutes    from './routes/messageRoutes';
import profileRoutes    from './routes/profileRoutes';
import notificationRoutes from './routes/notificationRoutes';
import homeworkRoutes   from './routes/homeworkRoutes';

app.use('/api/auth',      authRoutes);
app.use('/api/lessons',   lessonRoutes);
app.use('/api/grades',    gradeRoutes);
app.use('/api/users',     userRoutes);
app.use('/api/messages',  messageRoutes);
app.use('/api/profile',   profileRoutes);
app.use('/api/notifications', notificationRoutes);
app.use('/api/homework',  homeworkRoutes);
```

Маршрутизація REST API організована за ресурсним принципом — кожна сутність системи має власний набір ендпоінтів. Усі маршрути захищені middleware автентифікації `authMiddleware`, який перевіряє наявність та валідність JWT-токена у заголовку `Authorization` кожного вхідного запиту. Структура директорій серверної частини організована таким чином: `controllers/` містить функції-обробники запитів, `routes/` — визначення маршрутів із прив'язкою до контролерів, `middlewares/` — функції проміжної обробки, `utils/` — допоміжні утиліти для оновлення прогресу, відправки листів та розрахунку статусу заняття, `prisma/` — схему бази даних та файли міграцій.

Реалізація middleware автентифікації передбачає зчитування токена з заголовку `Authorization`, його верифікацію за допомогою бібліотеки `jsonwebtoken` та додавання декодованих даних користувача до об'єкта запиту для подальшого використання в контролерах:

```
export const authMiddleware = (req: AuthRequest, res: Response, next:
NextFunction) => {
  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res.status(401).json({ error: 'No token provided' });
  }

  const token = authHeader.split(' ')[1];

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET!) as JwtPayload;
    req.user = decoded;
    next();
  } catch (error) {
```

```

        return res.status(401).json({ error: 'Invalid token' });
    }
};

```

Контролер `authController` реалізує автентифікацію через два механізми. При вході за email та паролем функція `login` знаходить користувача у базі даних, порівнює хеш паролю через `bcrypt.compare()`, перевіряє статус `isActive` та генерує JWT-токен з вбудованим ідентифікатором та роллю з терміном дії сім діб. При вході через Google OAuth 2.0 функція `googleAuth` верифікує ідентифікаційний токен через Google API, знаходить або створює відповідний обліковий запис та генерує JWT-токен аналогічним чином. Контролер `lessonController` є найбільшим за обсягом та реалізує повний набір CRUD-операцій для управління заняттями. Перед збереженням нового або оновленого заняття викликається функція `hasOverlap`, яка перевіряє відсутність часових конфліктів у розкладі викладача:

```

async function hasOverlap(teacherId: number, date: Date, duration: number,
excludeLessonId?: number) {
    const start = date;
    const end = new Date(date.getTime() + duration * 60000);

    const candidates = await prisma.lesson.findMany({
        where: {
            teacherId: Number(teacherId),
            id: excludeLessonId ? { not: Number(excludeLessonId) } : undefined,
            OR: [
                { status: null },
                { status: { not: 'CANCELLED' } },
            ],
        },
        select: { id: true, date: true, duration: true },
    });

    return candidates.some(l => {
        const lStart = new Date(l.date);
        const lEnd = new Date(lStart.getTime() + l.duration * 60000);
        return start < lEnd && end > lStart;
    });
}

```

Функція отримує всі активні заняття викладача та для кожного перевіряє перетин часових інтервалів за формулою: початок нового заняття є меншим за кінець існуючого і кінець нового заняття є більшим за початок

існуючого. При редагуванні до умови WHERE додається виключення самого заняття що редагується через параметр `excludeLessonId`. При створенні повторюваних занять функція `hasOverlap` викликається окремо для кожного тижня у вказаному діапазоні.

Статус заняття не зберігається у базі даних як фіксоване значення, а розраховується динамічно при кожному отриманні даних через допоміжну функцію `computeLessonStatus` з файлу `lessonStatus.ts`. Функція порівнює поточний час із датою початку та тривалістю заняття і повертає один з чотирьох статусів: `UPCOMING` для майбутніх занять, `ACTIVE` якщо поточний час знаходиться всередині інтервалу заняття, `COMPLETED` для минулих занять та `CANCELLED` для скасованих. Такий підхід забезпечує актуальний статус без необхідності виконання планових задач для його оновлення.

Ключовим алгоритмом системи є розрахунок прогресу навчання учня. Функція `updateStudentProgress` викликається автоматично при кожній зміні оцінок та реалізує дворівневий аналіз: на першому рівні розраховуються агреговані показники успішності, на другому — перевіряється тенденція до зниження в розрізі кожного викладача. Допоміжна функція `calcProgress` обчислює три показники на основі масиву оцінок, впорядкованого за датою:

```
function calcProgress(grades: { value: number; type: string }[]) {
  if (grades.length === 0) return null;

  const avg = grades.reduce((s, g) => s + g.value, 0) / grades.length;
  const thematic = grades.filter(g => g.type === 'THEMATIC');
  const avgThematic = thematic.length
    ? thematic.reduce((s, g) => s + g.value, 0) / thematic.length
    : avg;

  let trend = 'STABLE';
  if (grades.length >= 4) {
    const half = Math.floor(grades.length / 2);
    const oldAvg = grades.slice(0, half).reduce((s, g) => s + g.value, 0) /
half;

    const newAvg = grades.slice(-half).reduce((s, g) => s + g.value, 0) /
half;

    if (newAvg > oldAvg + 3) trend = 'GROWING';
    else if (newAvg < oldAvg - 3) trend = 'DECLINING';
  }
}
```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return {
  successPercentage: Math.min(100, avg),
  averageThematicGrade: avgThematic,
  trend,
};
}

```

Відсоток успішності розраховується як середнє арифметичне всіх оцінок з обмеженням до 100. Середній тематичний бал розраховується лише на основі оцінок типу ТЕМАТИС, а якщо тематичних оцінок немає — використовується загальне середнє. Тенденція визначається шляхом порівняння середнього балу першої та другої половини всього масиву оцінок з порогом у 3 бали для визнання зміни суттєвою.

Після розрахунку показників функція `updateStudentProgress` отримує всі оцінки учня з бази даних, передає їх у `calcProgress` та зберігає результат через операцію `upsert`, яка оновлює вже наявний запис прогресу або створює новий, якщо учень ще не мав розрахованого прогресу раніше:

```

export async function updateStudentProgress(studentId: number) {
  try {
    const grades = await prisma.grade.findMany({
      where: { studentId },
      orderBy: { date: 'asc' },
      select: { value: true, type: true, teacherId: true, date: true },
    });

    const result = calcProgress(grades);
    if (!result) return;

    await prisma.progress.upsert({
      where: { studentId },
      update: { ...result, lastUpdated: new Date() },
      create: { studentId, ...result },
    });
  }
}

```

Після збереження агрегованих показників функція переходить до другого рівня аналізу — перевірки тенденції зниження успішності в розрізі кожного окремого викладача. Для цього оцінки групуються за ідентифікатором викладача за допомогою структури `Map`, що дозволяє розподілити їх по окремих масивах для кожного педагога:

```

const gradesByTeacher = new Map<number, typeof grades>();
for (const g of grades) {
  if (!g.teacherId) continue;
}

```

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!gradesByTeacher.has(g.teacherId)) gradesByTeacher.set(g.teacherId,
[]);
        gradesByTeacher.get(g.teacherId)!.push(g);
    }

    for (const [teacherId, teacherGrades] of gradesByTeacher) {
        const lastSix = teacherGrades.slice(-6);
        if (lastSix.length < 6 || !lastSix.every(g => g.value < 50)) continue;

```

Оцінки групуються за ідентифікатором викладача, після чого для кожної групи аналізуються останні шість значень. Якщо всі шість є нижчими за 50 балів, куратор учня отримує автоматичне сповіщення. Для запобігання дублюванню сповіщень перед відправкою перевіряється відсутність аналогічного сповіщення за останні 24 години. Сповіщення містить ім'я учня, ім'я викладача та предмет викладання, що дозволяє куратору миттєво зрозуміти суть проблеми без необхідності переходу на інші сторінки.

Система реального часу реалізована на базі бібліотеки Socket.IO. При підключенні клієнта WebSocket-з'єднання автентифікується за JWT-токеном, переданим у параметрах підключення, та користувач реєструється у персональній кімнаті з назвою `user:${userId}`. Така модель дозволяє точково надсилати події конкретному користувачу. Повідомлення чату зберігаються у базі даних та одночасно доставляються отримувачу через WebSocket-подію `new_message`, а сповіщення про оцінки, нові заняття та зміни в розкладі — через подію `new_notification`.

Клієнтська частина реалізована у вигляді SPA на базі React 18 з TypeScript та збирачем Vite. Точка входу `main.tsx` монтує кореневий компонент `App`, який обгортає весь застосунок провайдерами чотирьох контекстів: `AuthProvider` зберігає дані авторизованого користувача, `SocketProvider` ініціалізує та надає WebSocket-з'єднання, `ThemeProvider` управляє темою оформлення, `UnreadProvider` зберігає лічильник непрочитаних сповіщень.

Маршрутизація реалізована через React Router з розділенням на захищені та публічні маршрути. Компонент `ProtectedRoute` перевіряє наявність авторизованого користувача та при невдалій перевірці

перенаправляє на сторінку входу. Додатково реалізований компонент `ActiveRoute`, який перевіряє статус активації облікового запису — користувачі з неактивованим акаунтом перенаправляються на інформаційну сторінку очікування незалежно від наявності токена.

Усі HTTP-запити до серверного API виконуються через централізований екземпляр `Axios`, налаштований у файлі `axiosClient.ts`:

```
const axiosClient = axios.create({
  baseURL: import.meta.env.VITE_API_URL || 'http://localhost:5000/api',
  headers: {
    'Content-Type': 'application/json',
  },
});

axiosClient.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

Перехоплювач запитів автоматично зчитує JWT-токен із `localStorage` та додає його до заголовку `Authorization` кожного запиту. Це усуває необхідність повторення цього коду в кожному компоненті та забезпечує єдину точку управління автентифікацією на клієнтській стороні. Запити до різних ресурсів інкапсульовані в окремих модулях директорії `api/` — `lessonsApi.ts`, `gradesApi.ts`, `homeworkApi.ts` тощо — що дозволяє компонентам не залежати безпосередньо від деталей реалізації HTTP-запитів.

Компонентна архітектура організована за принципом ієрархічної декомпозиції. Компонент `Sidebar` динамічно формує набір пунктів меню залежно від ролі поточного користувача: адміністратор бачить «Дашборд» та «Адміністрування», куратор — «Дашборд», «Мої учні» та «Повідомлення», викладач — «Дашборд», «Розклад», «Домашні завдання» та «Повідомлення», учень — повний набір з шести пунктів включаючи «Мій прогрес» та «Мої викладачі». Компонент `TopBar` відображає верхню панель з лічильником непрочитаних сповіщень, що оновлюється у реальному часі через `WebSocket` без перезавантаження сторінки.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка розкладу SchedulePage є найбільш складним компонентом системи. Вона підтримує два режими відображення — тижневий WeekView та списочний ListView — між якими користувач перемикається через відповідні кнопки. Тижневий вигляд відображає заняття у вигляді карток у сітці з колонками для кожного дня та підтримує переміщення занять перетягуванням через нативний HTML5 Drag and Drop API. При завершенні перетягування обчислюється нова дата та час заняття на основі цільової клітинки сітки і надсилається запит на оновлення до API з повторною серверною перевіркою конфліктів. Кожна картка заняття відображає іконки наявності домашнього завдання та виставленої оцінки, дозволяючи викладачу швидко оцінити стан заняття без відкриття модального вікна.

Сторінка адміністрування AdminPage реалізована з вкладковою навігацією між трьома розділами — PendingTab для заявок, UsersTab для управління користувачами та CuratorsTab для призначень. Кожна вкладка є окремим компонентом з власною логікою отримання даних та відповідним інтерфейсом взаємодії.

Компонент ChatPage ініціалізує WebSocket-з'єднання при монтуванні та підписується на події нових повідомлень. Ліва панель формується функцією buildChatGroups, яка залежно від ролі поточного користувача збирає різні набори контактів та груп — для куратора це групи за кожним підопічним учнем разом із його викладачами, для викладача — групи за кожним учнем разом із куратором цього учня, для учня — окремі контакти куратора та призначених викладачів. При виборі контакту завантажуються історія повідомлень та нові повідомлення відображаються миттєво через WebSocket без перезавантаження сторінки.

Сторінка профілю ProfilePage підтримує перегляд та редагування профілю. Завантаження аватара реалізоване через пряме завантаження файлу до Supabase Storage з використанням клієнтського SDK — файл передається безпосередньо з браузера до сховища без проксіювання через серверну

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

частину, що зменшує навантаження на сервер. Отриманий публічний URL зберігається у базі даних через окремий запит до API профілю.

Висновки до розділу 2

У другому розділі визначено рольову модель системи з чотирма типами акторів та їхніми функціональними можливостями. Розроблено структуру реляційної бази даних з дев'ятьма таблицями, що охоплюють усі функціональні сутності платформи. Описано ключові алгоритми роботи системи — автентифікацію з підтримкою JWT та Google OAuth 2.0, управління розкладом із перевіркою конфліктів, розрахунок прогресу навчання та реалізацію реального часу через WebSocket. Наведено структуру серверної та клієнтської частин застосунку з описом компонентної будови фронтенду та модульної організації бекенду.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОПИС ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Порядок встановлення та налаштування системи

Розроблена вебплатформа розгорнута в хмарному середовищі та доступна користувачам через браузер без необхідності локального встановлення. Система складається з трьох незалежних компонентів: бази даних PostgreSQL на платформі Supabase, серверної частини на платформі Render та клієнтської частини на платформі Netlify. Така трикомпонентна архітектура дозволяє масштабувати кожен складову незалежно, а відмова одного компонента не призводить до повної недоступності системи.

База даних розгорнута на платформі Supabase, яка надає хмарний PostgreSQL з вебінтерфейсом управління, вбудованою автентифікацією, файловим сховищем та зручним дашбордом для адміністрування. Для початку роботи необхідно зареєструватись на supabase.com, створити новий проект та обрати регіон розгортання, оптимальний за географічним розташуванням відносно цільової аудиторії. Після ініціалізації проекту Supabase автоматично надає рядок підключення до бази даних у форматі PostgreSQL-з'єднання, який зазначається у змінній середовища DATABASE_URL серверної частини. Схема бази даних містить дев'ять основних таблиць, описаних у Prisma-схемі. Міграції схеми бази даних виконуються автоматично командою `prisma migrate deploy` при кожному запуску серверного застосунку на платформі Render, що гарантує відповідність структури бази даних актуальній версії схеми без ручного втручання.

Для зберігання аватарів користувачів у Supabase Storage необхідно перейти до розділу Storage у вебінтерфейсі Supabase, створити новий bucket з назвою `avatars` та встановити для нього режим `Public`. Після створення bucket налаштовуються Storage Policies — правила доступу, що дозволяють автентифікованим користувачам завантажувати файли та всім користувачам читати публічні URL-адреси. Для bucket `avatars` рекомендується налаштувати `INSERT-policy` для автентифікованих користувачів, `SELECT-policy` для

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

публічного читання та DELETE/UPDATE-policy лише для власника файлу. Максимальний розмір файлу обмежується на рівні 5 МБ, дозволені MIME-типи — image/jpeg, image/png, image/webp та image/gif. Ці обмеження забезпечують безпечне зберігання аватарів та запобігають завантаженню шкідливих файлів. Окрім основного рядка підключення до бази даних, для роботи клієнтського SDK Supabase Storage необхідно отримати два додаткових значення: SUPABASE_URL (публічна URL-адреса проекту) та SUPABASE_ANON_KEY (анонімний публічний ключ). Обидва значення доступні у розділі Settings → API вебінтерфейсу Supabase та використовуються у змінних середовища клієнтської частини.

Для розгортання серверної частини на платформі Render необхідно зареєструватись на render.com та підключити репозиторій GitLab або GitHub з вихідним кодом проекту. Після підключення репозиторію обирається тип сервісу Web Service та вказується директорія серверної частини як Root Directory. Платформа автоматично виявляє пакетний менеджер npm, встановлює залежності командою npm install та виконує команду збірки npm run build. Стартова команда налаштовується вручну: npx prisma migrate deploy && npm start — така послідовність гарантує, що міграції бази даних застосовуються перед запуском сервера. Для коректної роботи серверної частини необхідно налаштувати змінні середовища в розділі Environment платформи Render. DATABASE_URL містить повний рядок підключення до бази даних PostgreSQL на Supabase з обов'язковим параметром sslmode=require. JWT_SECRET — секретний ключ для підпису та верифікації JSON Web Tokens; рекомендується використовувати криптографічно стійкий випадковий рядок довжиною щонайменше 64 символи, згенерований командою:

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))".
```

GOOGLE_CLIENT_ID — ідентифікатор OAuth 2.0 застосунку, отриманий у Google Cloud Console при налаштуванні облікових даних для вебзастосунку; при цьому необхідно вказати дозволені URI переспрямування, включаючи

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

URL серверного API. RESEND_API_KEY — ключ API сервісу відправки електронних листів Resend, який отримується після реєстрації на resend.com та створення ключа в панелі управління. CORS_ORIGIN містить URL клієнтської частини на Netlify; допускається передача кількох значень через кому для підтримки одночасно хмарного та локального середовища розробки. NODE_ENV встановлюється у значення production для оптимізації продуктивності Node.js та вимкнення налагоджувального виводу.

Платформа Render підтримує автоматичне перерозгортання при кожному push-і до основної гілки репозиторію, що забезпечує безперервне постачання: після комміту змін нова версія серверного застосунку автоматично збирається та розгортається без ручного втручання. Безкоштовний тарифний план Render має обмеження: сервер переходить у режим очікування після 15 хвилин неактивності та потребує до 30 секунд для відновлення роботи при першому запиті. Для виробничого використання рекомендується перейти на платний план, який забезпечує постійну доступність. Журнали роботи серверного застосунку доступні в режимі реального часу через вкладку Logs платформи Render. Для критичних виробничих середовищ рекомендується налаштувати зовнішню систему моніторингу, наприклад UptimeRobot, яка надсилає HTTP-запити на ендпоінт /api/health кожні 5 хвилин — це одночасно запобігає переходу сервера в режим очікування на безкоштовному плані та сигналізує про збої в роботі.

Клієнтська частина розгорнута на платформі Netlify, яка спеціалізується на хостингу статичних сайтів та Single Page Applications з підтримкою автоматичних збірок, CDN-розповсюдження та HTTPS-сертифікатів. Перед збіркою необхідно налаштувати змінні середовища у файлі .env. VITE_API_URL містить повну URL-адресу серверної частини з суфіксом /api. VITE_SOCKET_URL — базову URL-адресу серверної частини без суфікса для WebSocket-з'єднання через Socket.IO. VITE_SUPABASE_URL та VITE_SUPABASE_ANON_KEY — відповідно публічна URL-адреса та анонімний ключ проекту Supabase для авторизації

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

клієнтського SDK Storage. Виробнича збірка виконується командою `npm run build` у директорії `frontend`, яка запускає Vite у режимі `production` та генерує оптимізовані та мінімізовані файли у папці `dist`. Для розгортання на Netlify папка `dist` завантажується вручну через вебінтерфейс або через Netlify CLI командою `netlify deploy --prod --dir=dist`.

Оскільки платформа є SPA з маршрутизацією на стороні клієнта, необхідно додатково налаштувати правила перенаправлення, щоб усі запити до шляхів, які фізично не існують у папці `dist`, перенаправлялись на `index.html`. Для цього у директорії `public` клієнтської частини створюється файл `_redirects` з таким вмістом: `/* /index.html 200`. Без цього налаштування при прямому переході за URL виду `/schedule` або `/profile` Netlify повертає помилку 404, оскільки вся маршрутизація обробляється JavaScript-кодом React Router вже після завантаження головного HTML-файлу.

Для локального запуску системи розробнику необхідно послідовно клонувати репозиторій, встановити залежності серверної частини командою `npm install` у директорії `backend`, створити файл `.env` та заповнити всі обов'язкові змінні середовища, виконати міграції бази даних командою `prisma migrate dev`, яка застосовує всі файли міграцій та генерує оновлений Prisma Client, і запустити серверний застосунок у режимі розробки командою `npm run dev`, що використовує `nodemon` для автоматичного перезапуску при змінах у вихідному коді. Серверна частина за замовчуванням доступна на порті 5000. Клієнтська частина запускається аналогічно: після встановлення залежностей та налаштування файлу `.env` з локальними URL-адресами запускається Vite dev-сервер командою `npm run dev` на порті 5173 з підтримкою HMR — автоматичного оновлення модулів у браузері без повного перезавантаження сторінки. Початкове наповнення бази даних адміністраторським обліковим записом виконується скриптом `node create-admin.js`, розташованим у директорії `backend`. Скрипт створює користувача з роллю `ADMIN` та статусом `isActive: true`, що дозволяє відразу увійти в

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

систему та розпочати адміністрування. Решта облікових записів створюються через штатний процес подачі заявок та їх підтвердження адміністратором.

3.2. Структура інтерфейсу вебплатформи

Інтерфейс платформи розроблений відповідно до принципів адаптивного дизайну та забезпечує зручну роботу як на десктопних, так і на мобільних пристроях. Навігація здійснюється через бічну панель Sidebar (рис.3.1), яка містить посилання на основні розділи залежно від ролі користувача, та верхню панель TopBar з кнопкою сповіщень, посиланням на профіль та кнопкою виходу (рис.3.2).

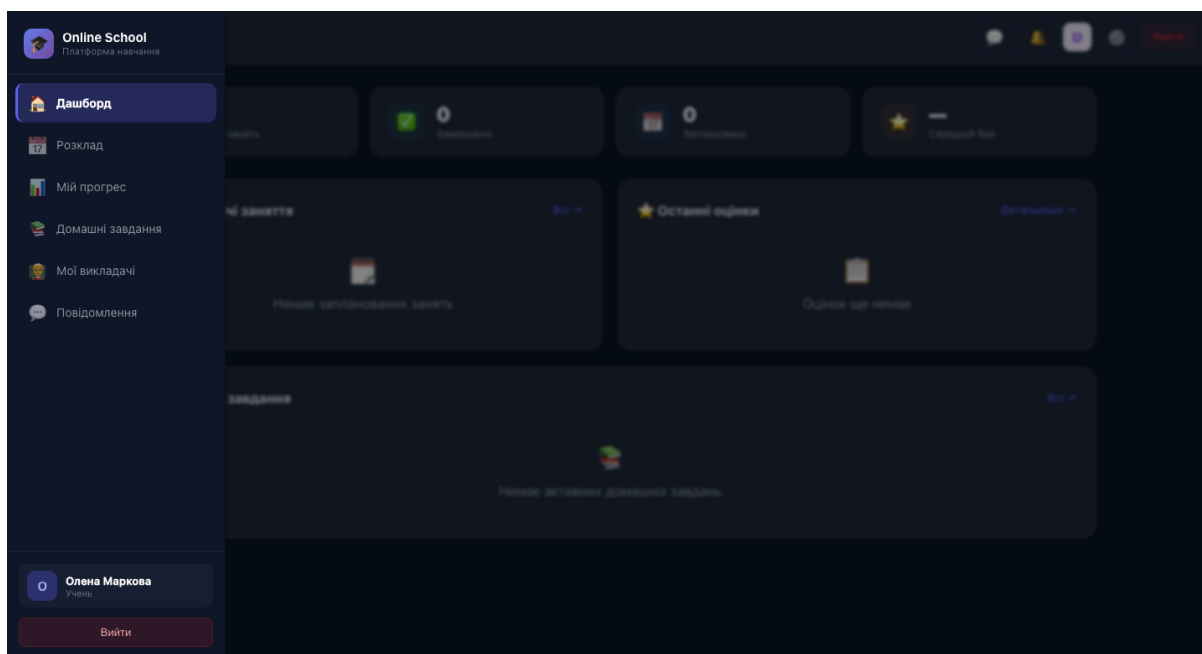


Рис. 3.1. Навігація через бічну панель Sidebar.



Рис. 3.2. Панель TopBar з кнопкою сповіщень, посиланням на профіль та кнопкою виходу.

Сторінка автентифікації надає користувачу два способи входу — через електронну пошту та пароль або через обліковий запис Google (рис. 3.3). Форма виконана в темній кольоровій гамі з логотипом сервісу у верхній частині. Кнопка основного входу виділена градієнтним фоном, що відрізняє її від кнопки входу через Google, розташованої нижче.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

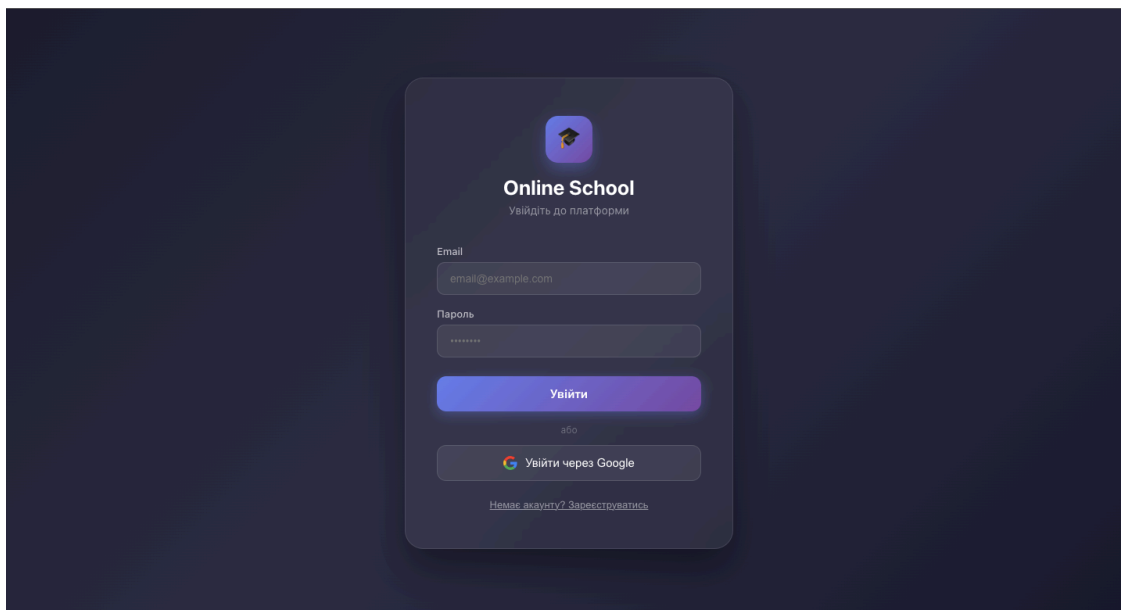


Рис. 3.3. Сторінка автентифікації.

Нові користувачі можуть подати заявку на вступ, вказавши ім'я, електронну пошту, бажану роль та пароль (рис. 3.4). Роль обирається з випадного списку, що містить доступні варіанти реєстрації. Після заповнення форми заявка надсилається на розгляд через кнопку «Надіслати заявку».

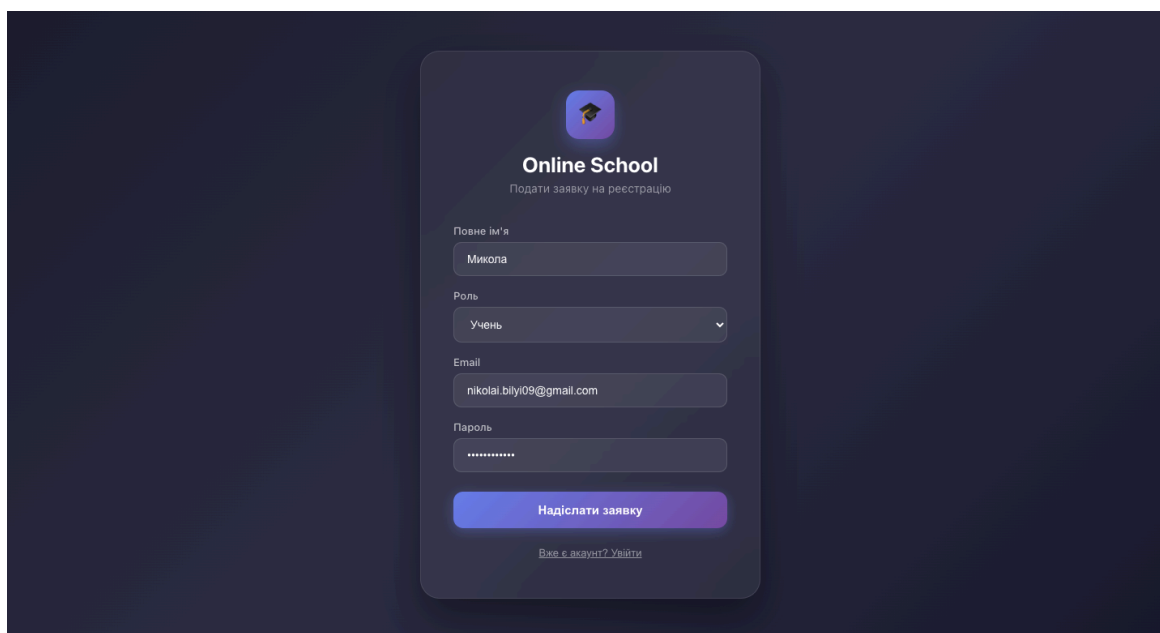


Рис. 3.4. Сторінка подачі заявки на вступ.

Після подачі заявки користувач отримує сповіщення про необхідність очікування підтвердження від адміністратора (рис. 3.5). Сторінка містить покроковий перелік подальших дій — від розгляду заявки адміністратором до надсилання листа на електронну пошту з результатом. Унизу

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

відображається електронна адреса, вказана при реєстрації, та кнопка повернення до сторінки входу.

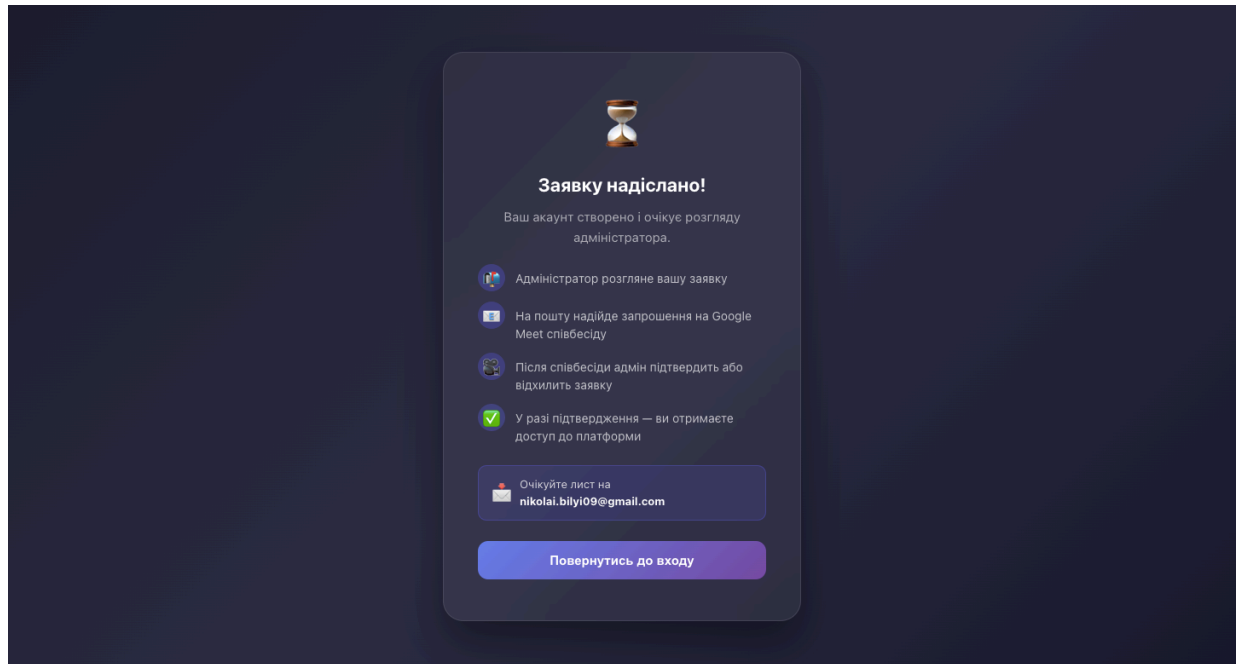


Рис. 3.5. Інформаційна сторінка очікування підтвердження від адміністратора.

Головна сторінка Dashboard відображає персоналізований привітальний блок та набір статистичних карток залежно від ролі користувача. Для адміністратора відображається кількість нових заявок, загальна кількість користувачів, учнів, викладачів та кураторів, а також список заявок що очікують розгляду (рис.3.6).

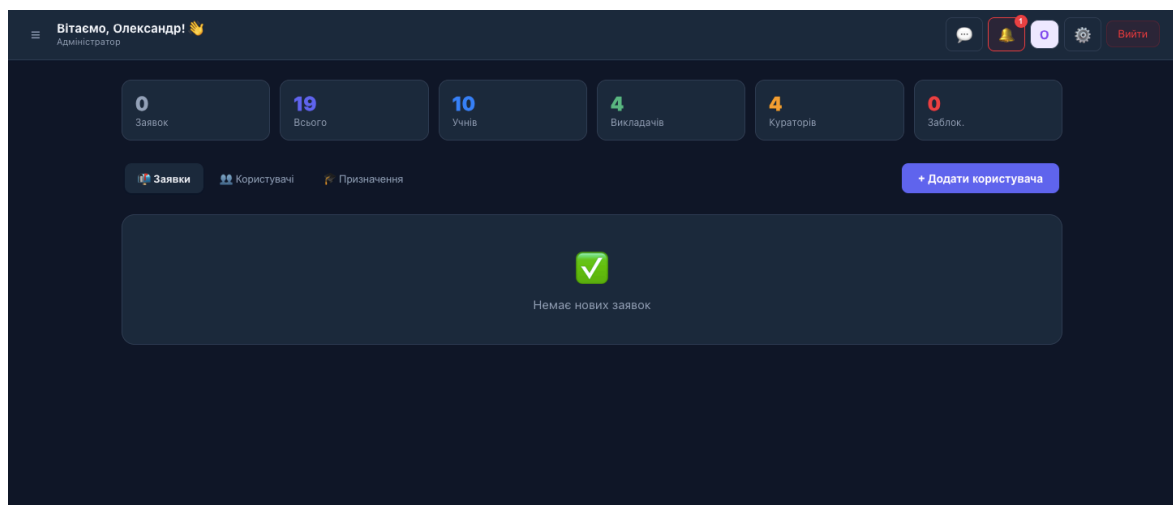


Рис. 3.6. Dashboard адміністратора.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Для викладача відображається кількість занять на поточний тиждень, список найближчих занять, а також список домашніх завдань (рис. 3.7). У верхній частині сторінки розміщено картки з ключовими показниками — кількістю занять, завершених занять, учнів та домашніх завдань. Список найближчих занять показує дату, час та тривалість кожного заняття з можливістю переходу до нього, а блок домашніх завдань відображає статус виконання та термін подачі.

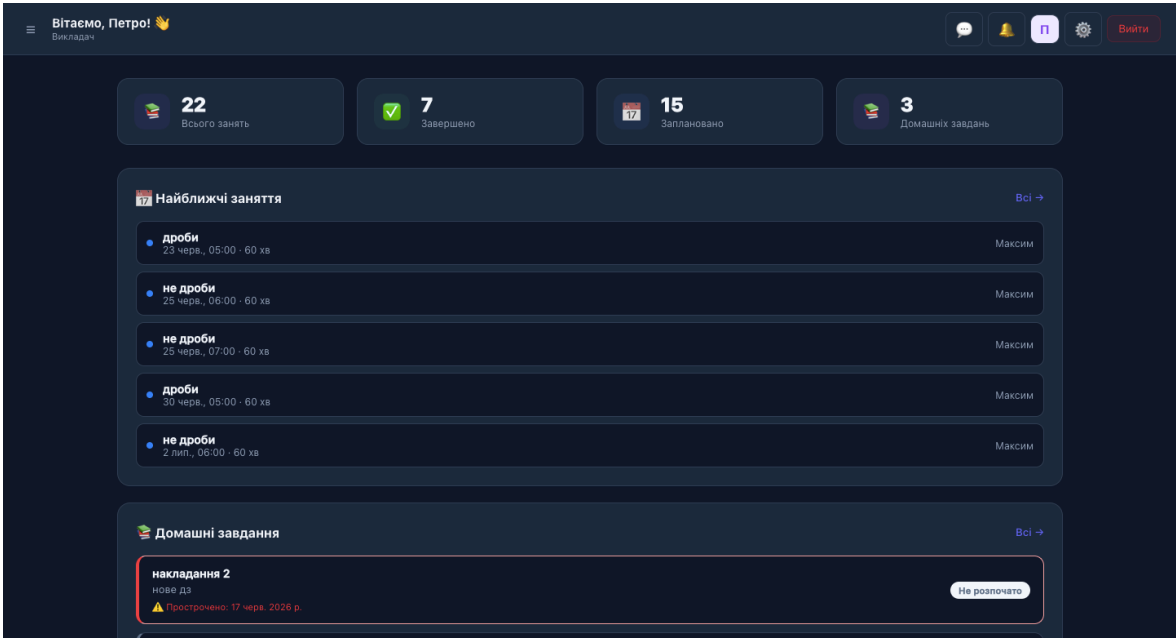


Рис. 3.7. Dashboard викладача.

Для учня показується прогрес навчання, найближче заняття та домашнє завдання (рис. 3.8). Картки у верхній частині відображають кількість занять, завершених занять, учнів-однолітків (за наявності) та загальний відсоток успішності. Окремий блок «Останні оцінки» показує нещодавно отримані оцінки з кольоровим кодуванням за рівнем балу, а блок домашніх завдань вказує статус та дедлайн виконання.

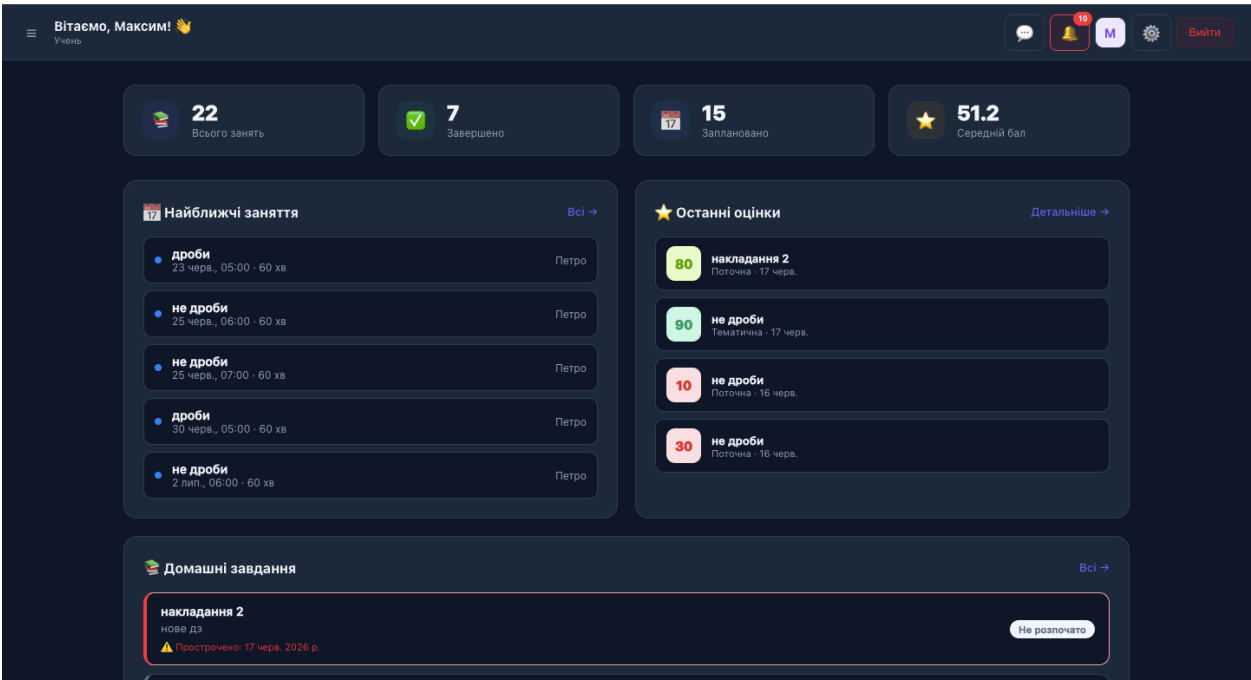


Рис. 3.8. Dashboard учня.

Для куратора відображається зручний пошук викладачів, а також вкладка з призначеними йому учнями (рис. 3.9). Пошук дозволяє фільтрувати викладачів за напрямком підготовки та освітньою програмою. Кожна картка викладача в результатах пошуку містить ім'я, контактні дані, мову викладання, досвід роботи та середній рейтинг.

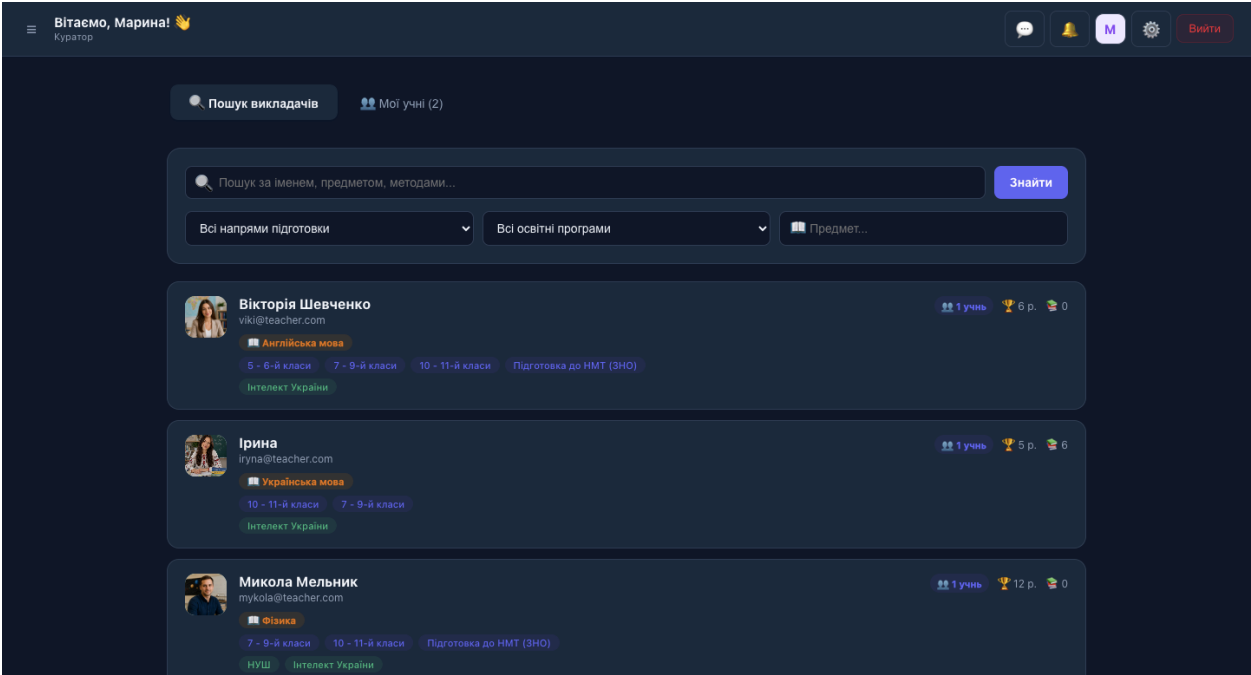


Рис. 3.9. Dashboard куратора.

Сторінка розкладу SchedulePage є центральним функціональним елементом платформи. Вона підтримує два режими відображення — тижневий та списочний — між якими користувач може перемикається. У тижневому режимі заняття відображаються у вигляді карток у сітці з колонками для кожного дня тижня. Картка заняття містить час початку, назву, ім'я учня або викладача (залежно від ролі), а також іконки що сигналізують про наявність домашнього завдання та виставленої оцінки (рис.3.10).

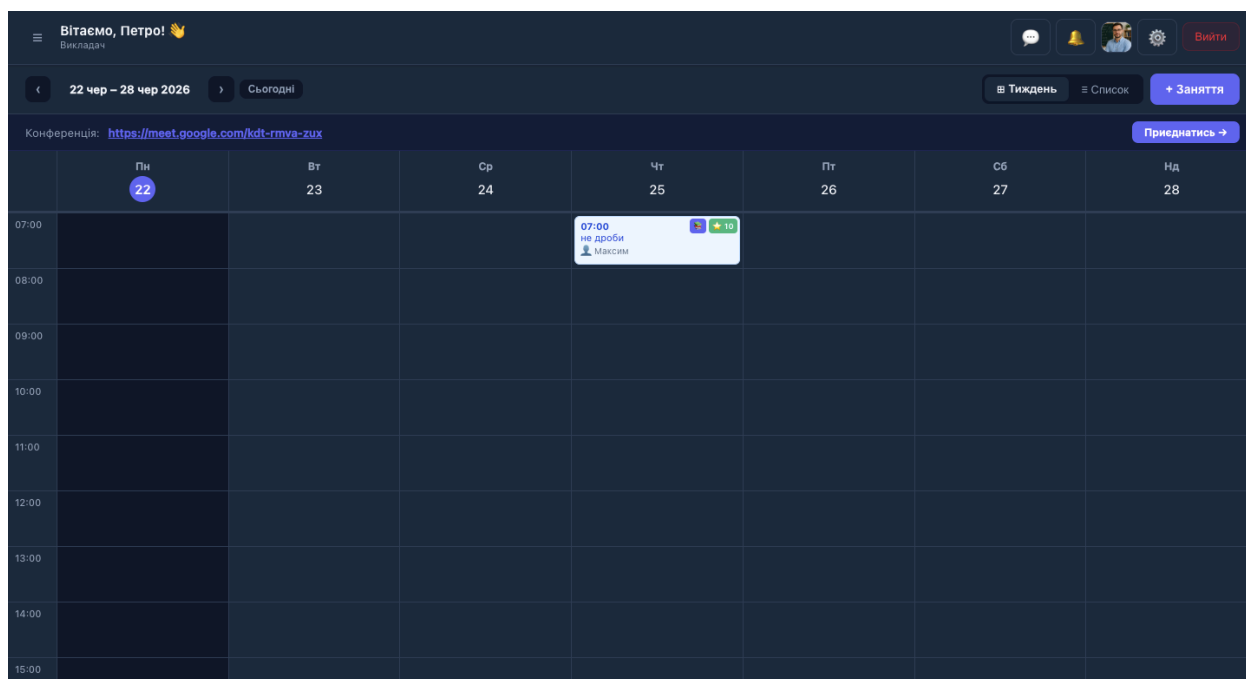


Рис. 3.10. Сторінка розкладу (SchedulePage).

При натисканні на картку відкривається модальне вікно з детальною інформацією про заняття — темою, статусом, посиланням на відеоконференцію, оцінкою та домашнім завданням. Для викладача в модальному вікні доступні кнопки виставлення оцінки, додавання домашнього завдання та редагування заняття, а також прикріплених до нього домашнього завдання та оцінки (рис.3.11).

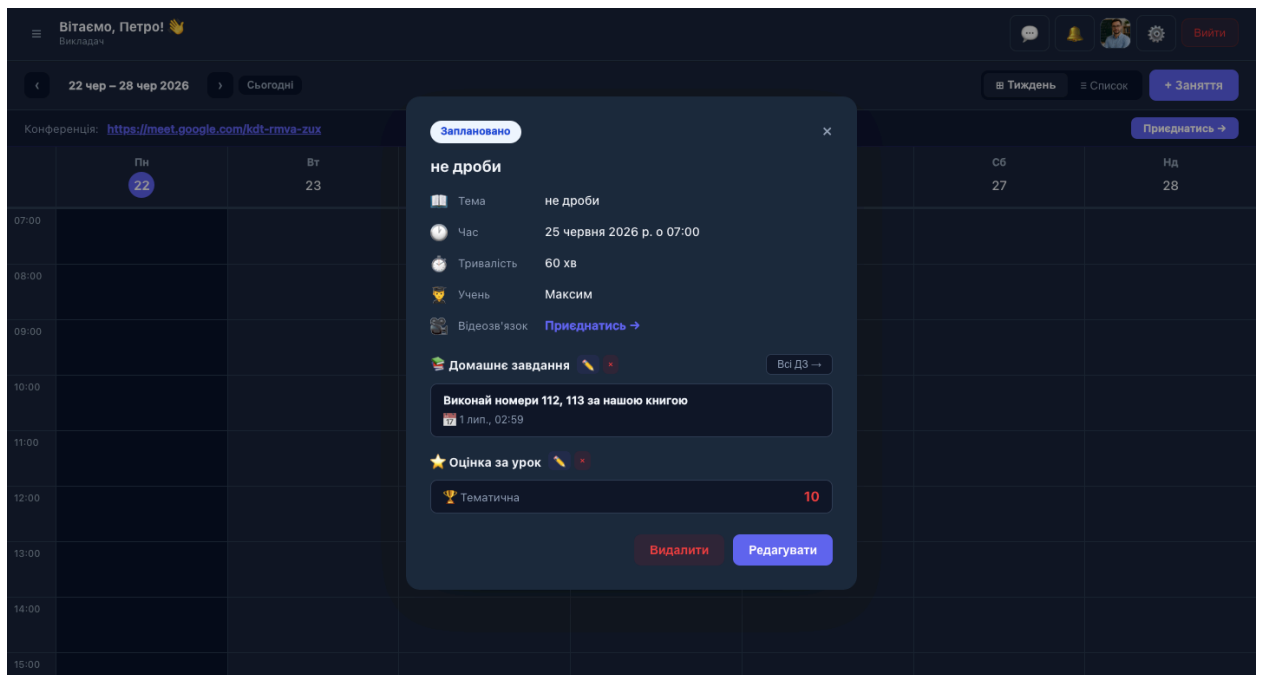


Рис. 3.11. Модальне вікно з детальною інформацією про заняття.

Викладач може переміщати картку заняття перетягуванням для швидкої зміни часу. При редагуванні повторюваного заняття система пропонує вибір області змін — лише поточне заняття, поточне та всі майбутні або всі заняття групи (рис.3.12).

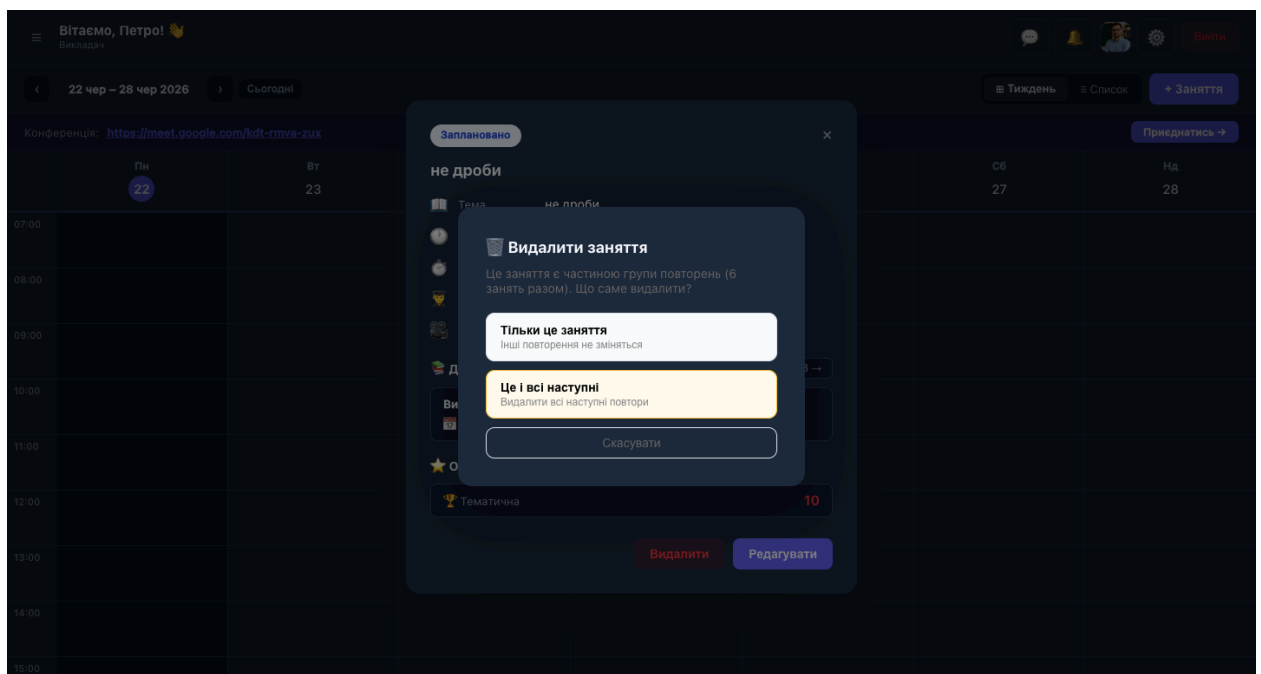


Рис. 3.12. Модальне вікно при редагуванні повторюваного заняття.

Сторінка домашніх завдань HomeworkPage відображає список усіх домашніх завдань учня з фільтрацією за статусом виконання. Кожне завдання

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

показує назву пов'язаного заняття, текст завдання, дедлайн та поточний статус. Учень може відкрити завдання та надіслати виконану роботу у вигляді текстового коментаря. Для викладача сторінка відображає завдання що очікують перевірки з можливістю переглянути відповідь учня та виставити оцінку (рис.3.13).

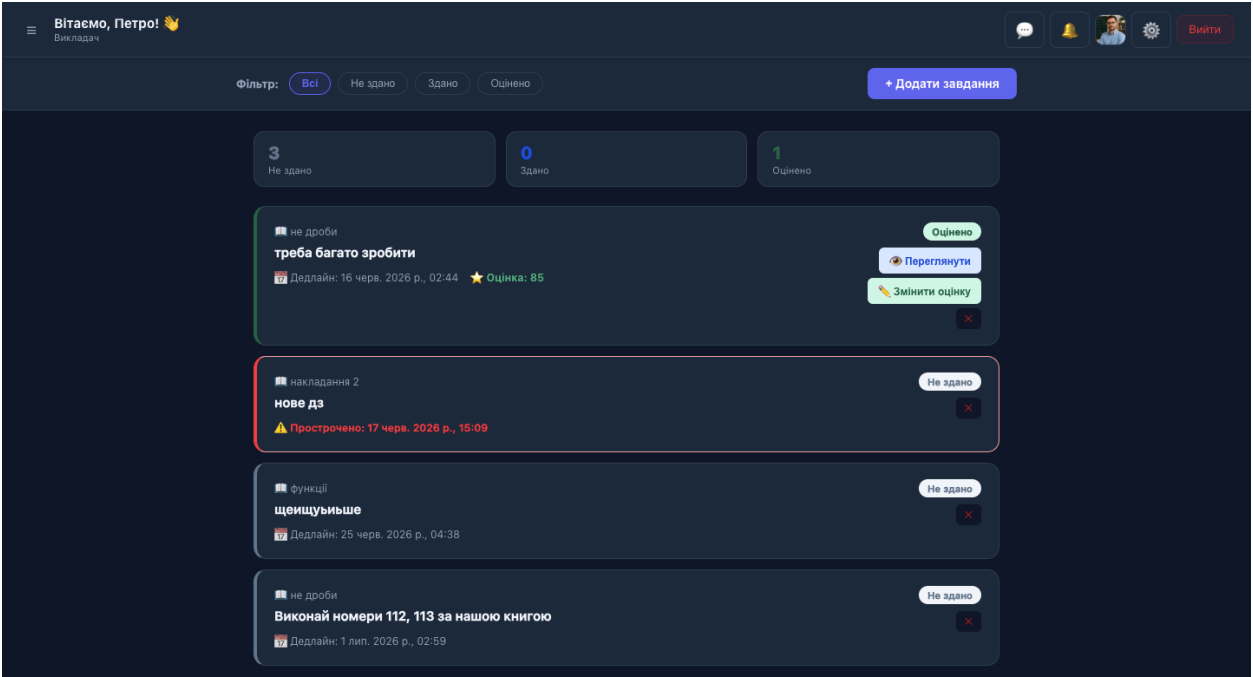


Рис. 3.13. Сторінка домашніх завдань (HomeworkPage).

Сторінка профілю ProfilePage відображає повну інформацію про користувача — аватар, ім'я, роль, дату реєстрації на платформі, біографію, мотивацію та контактні дані. Для викладача додатково відображаються предмет викладання, напрями підготовки, освітні програми, досвід та методи викладання. Власник профілю може редагувати інформацію та завантажувати фото профілю безпосередньо з пристрою — файл зберігається у хмарному сховищі Supabase Storage та публічне посилання зберігається в базі даних (рис.3.14 та рис.3.15).

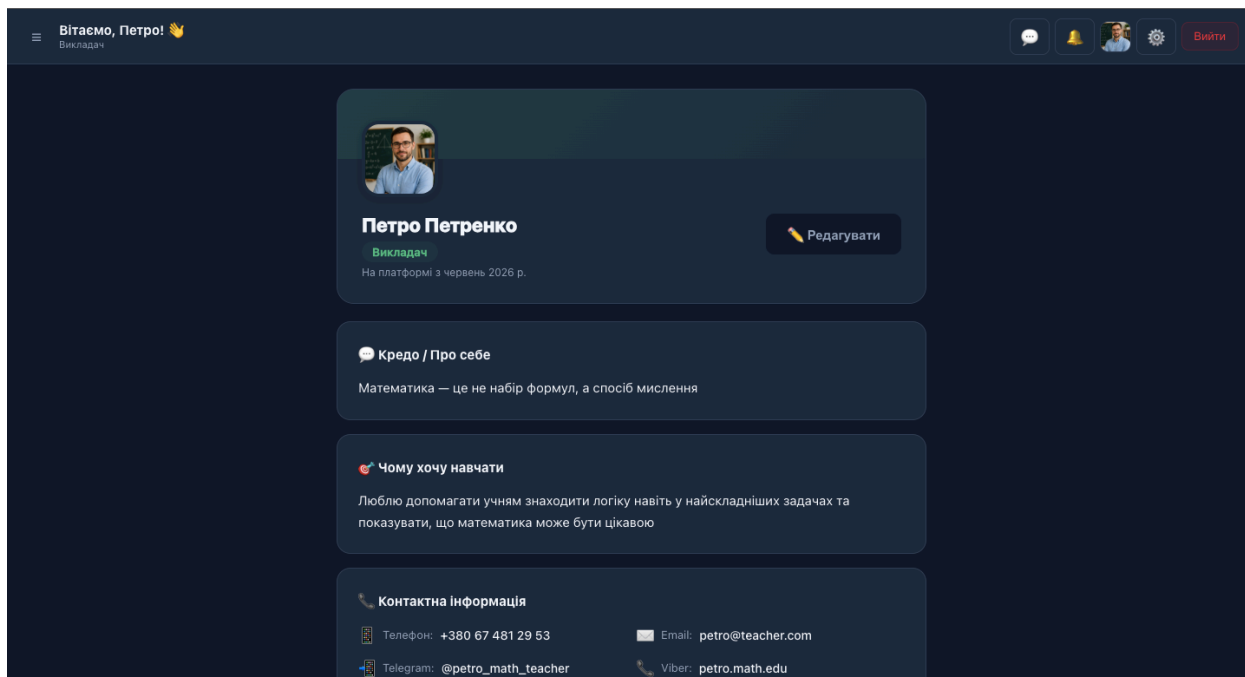


Рис. 3.14. Сторінка профілю (ProfilePage).

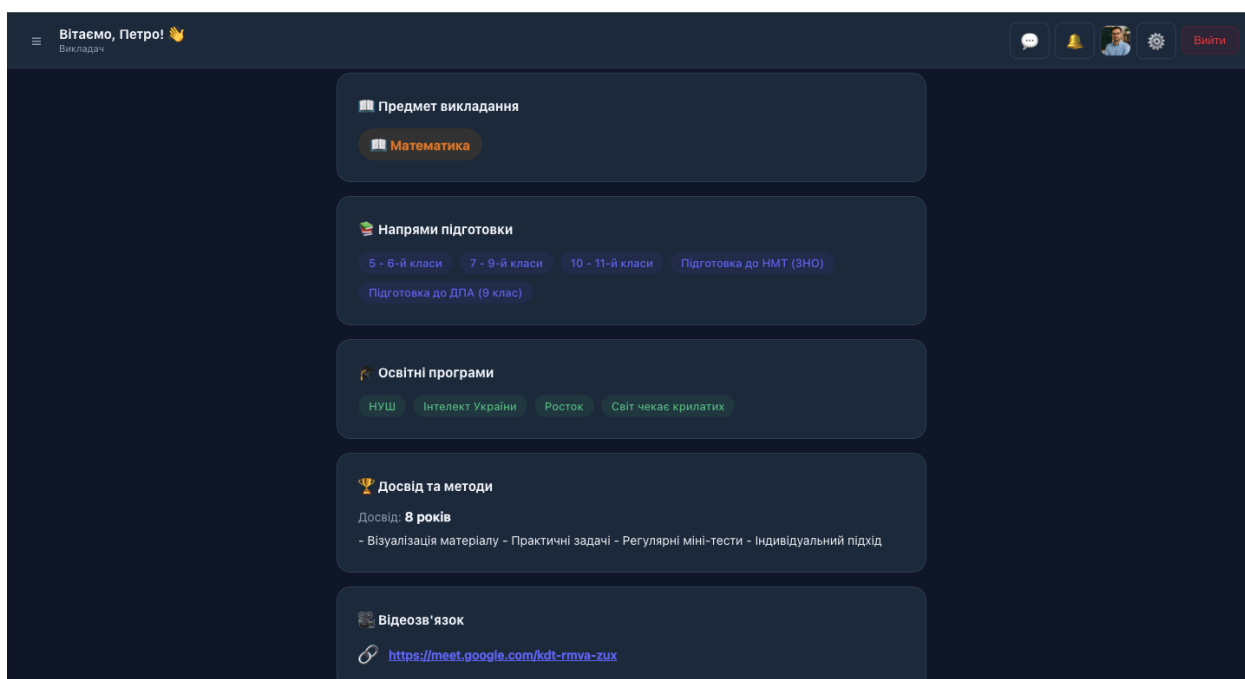


Рис. 3.15. Продовження сторінки профілю (ProfilePage).

Сторінка чату ChatPage реалізує систему внутрішніх повідомлень. Ліва панель відображає список доступних контактів та груп залежно від ролі користувача — учень бачить свого куратора та всіх призначених викладачів, викладач бачить групи за кожним своїм учнем, куратор бачить групи за кожним підопічним учнем разом з його викладачами (рис.3.16).

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

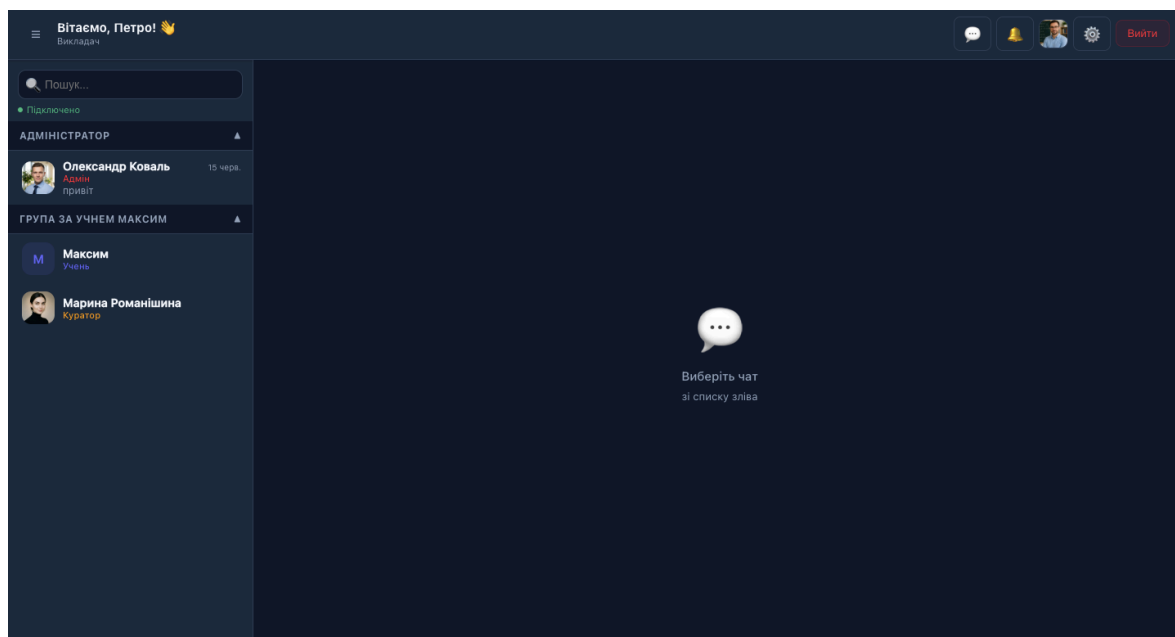


Рис. 3.16. Сторінка чату (ChatPage).

При виборі контакту в правій панелі відображається історія листування та форма надсилання повідомлення. Нові повідомлення доставляються миттєво через WebSocket без перезавантаження сторінки (рис. 3.17). У лівій панелі контакти згруповані за категоріями — наприклад, «Адміністратор» та групи за конкретним учнем, що відповідає логіці функції buildChatGroups, описаній раніше. Поле введення повідомлення розташоване в нижній частині екрана з кнопкою відправки.

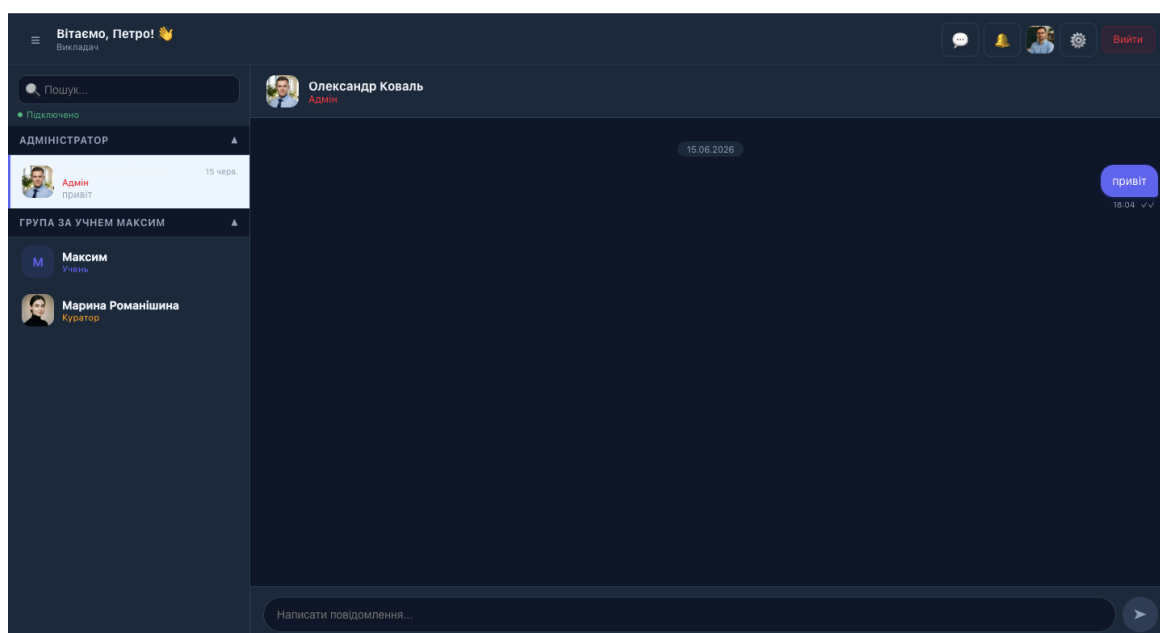


Рис. 3.17. Відображення історії листування та форма надсилання повідомлення.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка адміністрування AdminPage доступна лише користувачу з роллю ADMIN та містить три вкладки. Вкладка заявок відображає список нових заявок на вступ з можливістю призначити дату та час співбесіди і надіслати запрошення електронною поштою, або відхилити заявку (рис. 3.18). У верхній частині сторінки наведено зведену статистику — кількість заявок, користувачів, викладачів, кураторів та заблокованих облікових записів. Кожна заявка в списку містить ім'я кандидата, електронну пошту, бажану роль та дату подачі.

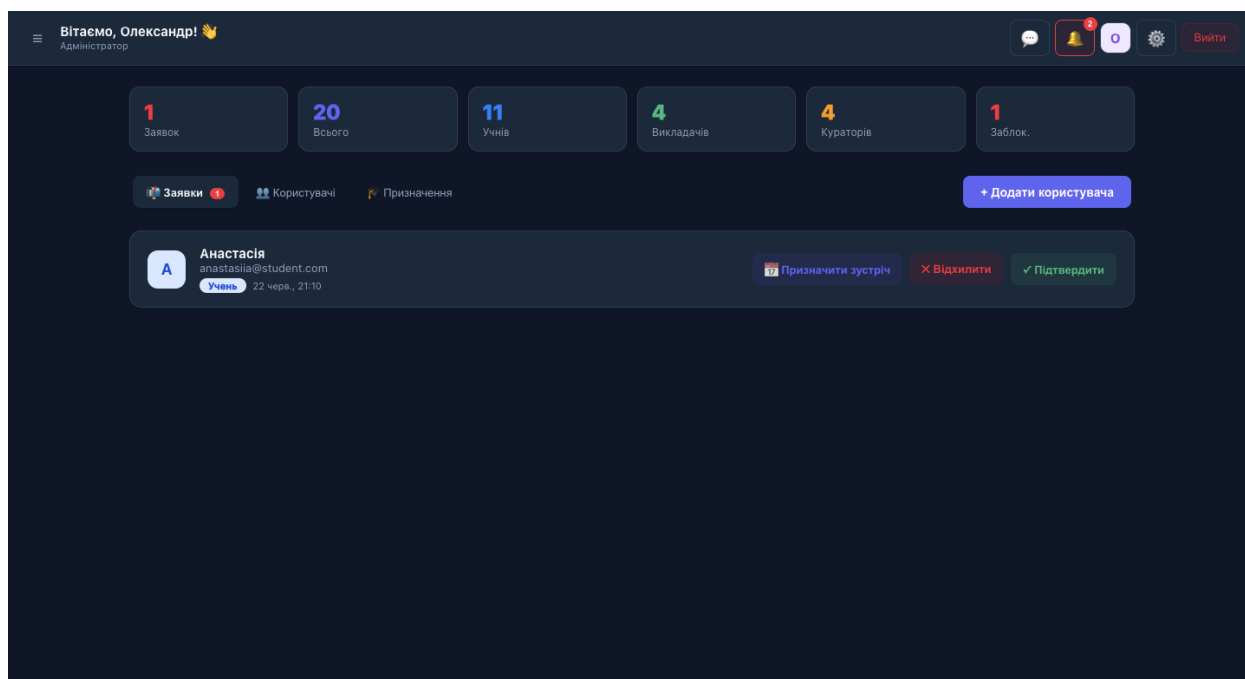


Рис. 3.18. Сторінка адміністрування (AdminPage).

Вкладка користувачів відображає повний список зареєстрованих користувачів з можливістю фільтрації за роллю, пошуку за ім'ям та блокування або видалення облікових записів (рис. 3.19). Таблиця містить ім'я, електронну пошту, роль, статус активності та дату реєстрації кожного користувача, а для кожного запису доступні дії редагування та блокування.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

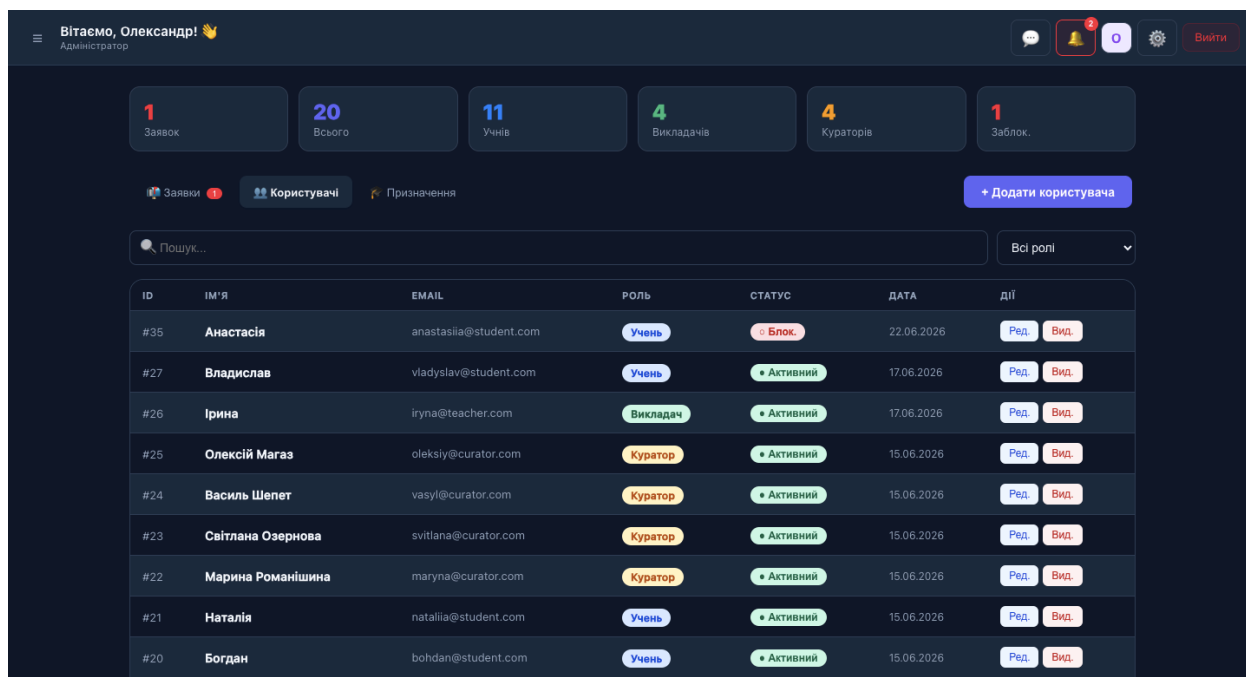


Рис. 3.19. Вкладка користувачів на сторінці адміністрування.

Вкладка кураторів відображає навантаження кожного куратора — список закріплених учнів та статус призначення викладача до кожного з них (рис. 3.20). Над списком розташована форма призначення куратора обраному учню через два випадні списки. Кожна картка куратора показує кількість підопічних учнів і вказує, чи всім із них вже призначено викладача.

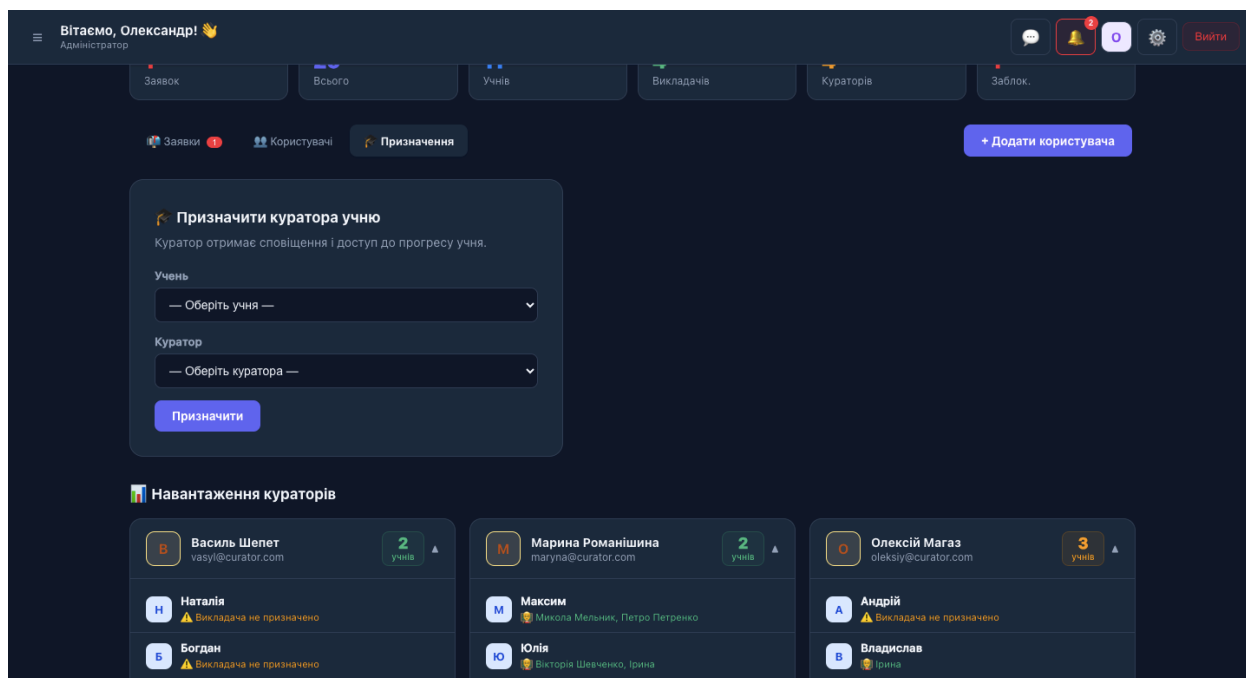


Рис. 3.20. Вкладка кураторів на сторінці адміністрування.

Сторінка куратора CuratorPage надає куратору інструменти для управління підопічними учнями. Вкладка учнів відображає список закріплених учнів з їхніми профілями та прогресом навчання (рис. 3.21). Кожна картка учня містить контактні дані, список призначених викладачів та індикатор прогресу у відсотках, а також кнопки для перегляду статистики чи переходу до листування.

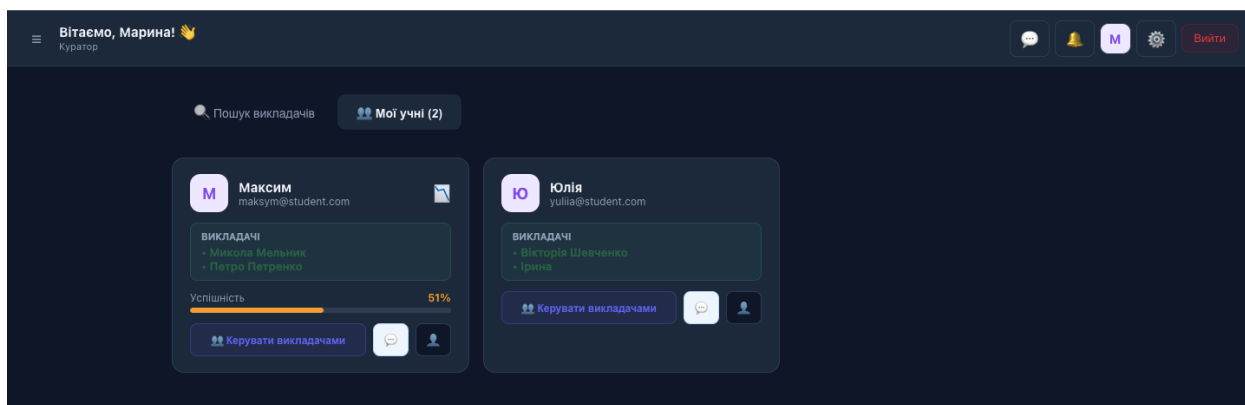


Рис. 3.21. Вкладка учнів на сторінці куратора.

Вкладка пошуку викладачів дозволяє знаходити викладачів за предметом та призначати їх до обраного учня (рис. 3.22). Форма пошуку містить поле для введення імені чи предмету, а також фільтри за напрямком підготовки та освітньою програмою. Кожен результат пошуку відображає ім'я викладача, мову викладання, досвід та рейтинг.

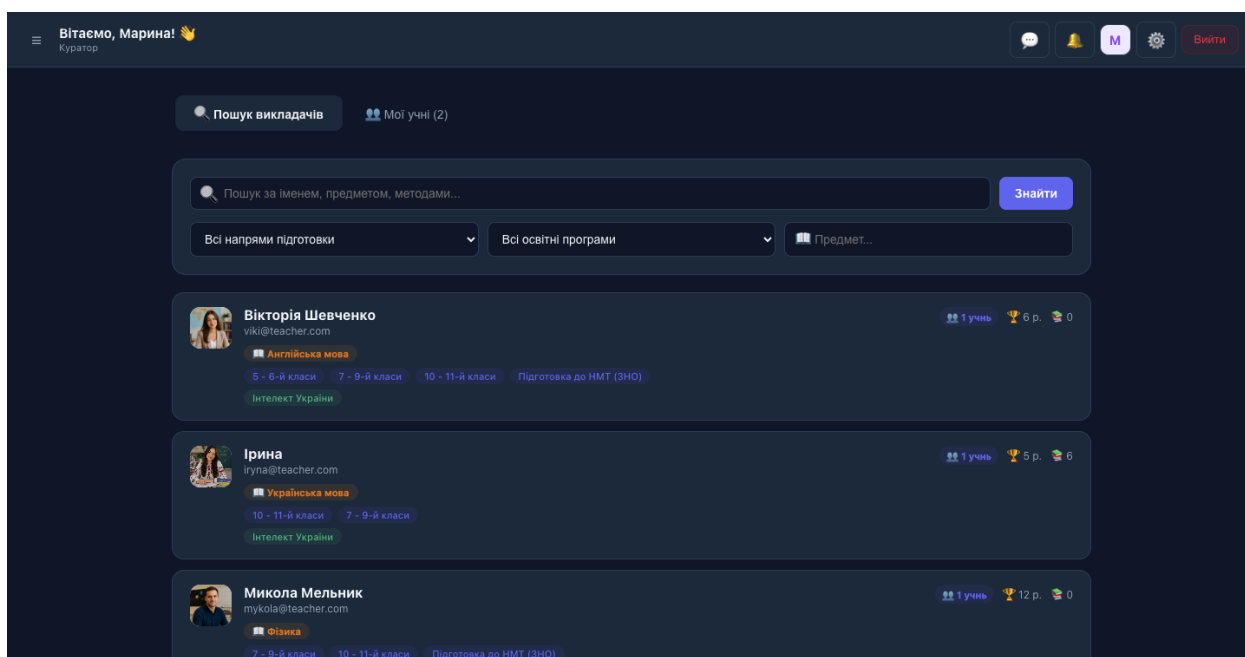


Рис. 3.22. Вкладка пошуку викладачів на сторінці куратора.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк. 72
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		

При виборі учня система відображає вже призначених до нього викладачів та дозволяє додавати нових або видаляти існуючих (рис. 3.23). У модальному вікні кожен призначений викладач позначається статусом (наприклад, «Призначено»), а поруч розташована кнопка видалення; для викладачів, що ще не призначені учню, доступна кнопка додавання.

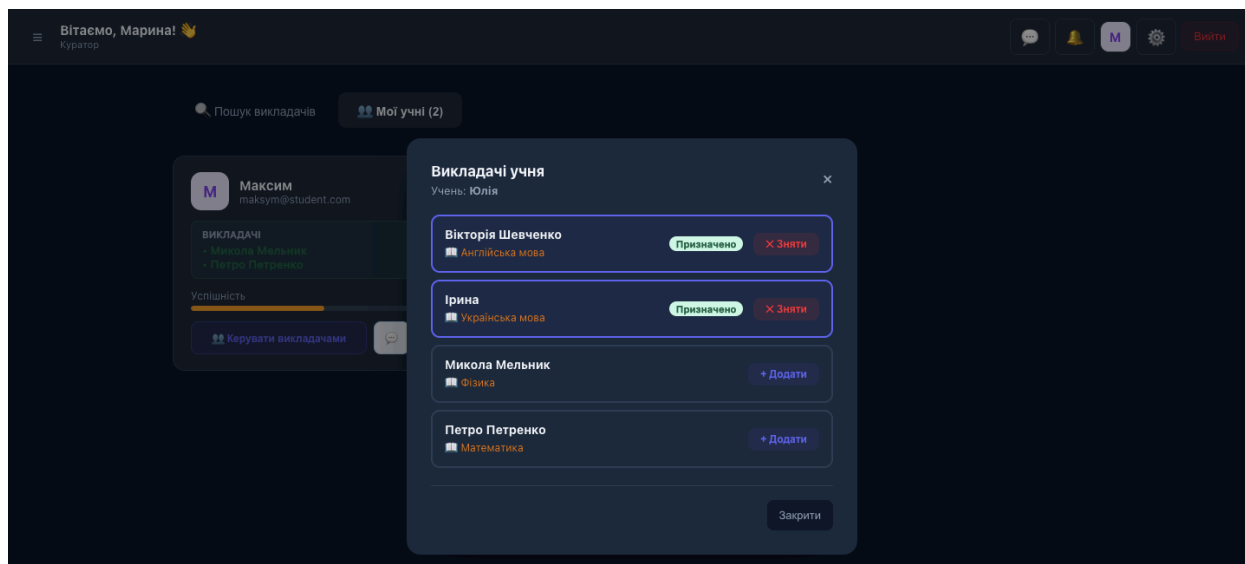


Рис. 3.23. Модальне вікно призначення викладачів.

3.3. Тестування роботи програмного комплексу

Тестування системи проводилось методом функціонального тестування — перевірки відповідності реалізованого функціоналу визначеним вимогам. Для кожного ключового сценарію роботи визначено вхідні дані, очікуваний результат та фактичний результат виконання. Тестування охоплювало п'ять функціональних модулів: автентифікація та управління обліковими записами, управління розкладом занять, система оцінювання та розрахунок прогресу, домашні завдання та реальноточасова комунікація, адміністрування платформи.

Тестування процесу реєстрації та автентифікації охоплювало сім сценаріїв. При поданні заявки з коректними даними — унікальна електронна пошта, повне ім'я, обрана роль та пароль — система зберігає заявку в таблиці ApplicationRequest зі статусом очікування та відображає інформаційну сторінку з інструкцією щодо подальших кроків; тест пройдено успішно. При спробі подати заявку з вже зареєстрованою електронною поштою сервер

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

повертає відповідь зі статусом 409 Conflict та повідомленням про дублювання, форма не очищається, заявка не зберігається — серверна валідація на унікальність електронної пошти спрацювала коректно, тест пройдено успішно. При вході з коректними обліковими даними активованого облікового запису сервер повертає JWT-токен з терміном дії сім діб, токен зберігається у localStorage, користувач перенаправляється на Dashboard відповідно до своєї ролі; тест пройдено успішно. При вході з некоректним паролем сервер повертає відповідь зі статусом 401 Unauthorized, токен не генерується, відображається повідомлення про помилку автентифікації — функція bcrypt.compare() повернула false, обробник помилок відпрацював очікувано, тест пройдено успішно. При вході через Google OAuth сервер верифікує ідентифікаційний токен через google-auth-library, знаходить або створює відповідний обліковий запис та повертає JWT-токен платформи; тест пройдено успішно. При спробі доступу до захищеного ресурсу без заголовку Authorization сервер повертає відповідь зі статусом 401 вже на рівні authMiddleware, не виконуючи бізнес-логіку контролера; тест пройдено успішно. При спробі доступу до адміністраторського ресурсу з токеном ролі STUDENT roleMiddleware повертає відповідь зі статусом 403 Forbidden; тест пройдено успішно.

Тестування управління розкладом занять включало сім сценаріїв. При створенні заняття з часом, що не перетинається з жодним існуючим заняттям викладача, заняття зберігається в таблиці Lesson, учень отримує WebSocket-сповіщення типу new_lesson, заняття відображається в розкладі; тест пройдено успішно. При спробі створити заняття з часом, що перетинається з існуючим — наприклад, нове заняття о 10:00 тривалістю 60 хвилин за наявності заняття з 9:30 до 10:30 — функція hasOverlap() виявляє перетин інтервалів, сервер повертає відповідь зі статусом 409 та повідомленням про конфлікт, заняття не зберігається; тест пройдено успішно. При створенні повторюваного заняття на 4 тижні система генерує унікальний UUID групи та створює 4 пов'язані заняття в таблиці Lesson з однаковим значенням поля

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

groupId, кожне зміщене на 7 днів відносно попереднього; тест пройдено успішно. При редагуванні одного заняття з групи з вибором опції «Лише це заняття» змінюється виключно вибране заняття, решта занять групи залишаються незмінними; тест пройдено успішно. При видаленні з вибором опції «Це і всі наступні» система видаляє вибране заняття та всі майбутні заняття з тим самим groupId, тоді як минулі заняття зберігаються; тест пройдено успішно. При переміщенні картки заняття перетягуванням на нову клітинку тижневого календаря система обчислює новий час, надсилає PATCH-запит до серверного API, сервер повторно перевіряє відсутність конфліктів та зберігає оновлені дату і час; тест пройдено успішно. При скасуванні заняття поле status встановлюється у значення CANCELLED, заняття відображається як скасоване без фізичного видалення, пов'язані оцінки та домашні завдання зберігаються; тест пройдено успішно.

Тестування системи оцінювання та розрахунку прогресу охоплювало чотири сценарії. При виставленні поточної оцінки запис зберігається в таблиці Grade, автоматично викликається функція updateStudentProgress(), прогрес учня перераховується та зберігається в таблиці Progress, учень отримує WebSocket-сповіщення про нову оцінку; тест пройдено успішно. Для перевірки коректності розрахунку тенденції виставлялось 8 оцінок — перші 4 зі значеннями 40, 45, 50, 45 (середнє 45), наступні 4 зі значеннями 70, 75, 80, 85 (середнє 77.5); функція calcProgress() визначила тенденцію як GROWING, оскільки різниця між середніми значеннями другої та першої половини перевищує поріг у 3 бали, тест пройдено успішно. При виставленні 6 оцінок підряд від одного викладача зі значеннями нижче 50 балів функція updateStudentProgress() виявила критичну тенденцію, перевірила відсутність аналогічного сповіщення за останні 24 години, створила запис у таблиці Notification для куратора та доставила його через WebSocket в режимі реального часу; тест пройдено успішно. При виставленні ще однієї оцінки нижче 50 від того ж викладача протягом 24 годин функція виявила наявний запис у таблиці сповіщень і не створила дублікат; тест пройдено успішно.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижомотря О.Г.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестування системи домашніх завдань охоплювало чотири сценарії. При додаванні домашнього завдання до заняття запис зберігається в таблиці Homework, на картці заняття в розкладі з'являється відповідна іконка, учень отримує сповіщення; тест пройдено успішно. При надсиланні виконаної роботи учнем запис зберігається в таблиці HomeworkSubmission, статус завдання в таблиці Homework оновлюється до значення SUBMITTED, відображення статусу на сторінці HomeworkPage змінюється; тест пройдено успішно. При перевірці відповіді та виставленні оцінки за домашнє завдання оцінка зберігається в таблиці Grade з заповненим полем homeworkId, що реалізує унікальний зв'язок «один-до-одного» між оцінкою та домашнім завданням, учень отримує сповіщення; тест пройдено успішно. Завдання з дедлайном, що минув, та статусом «не надіслано» коректно відображається на сторінці HomeworkPage з відповідним маркуванням та датою у виділеному форматі; тест пройдено успішно.

Тестування системи реального часу перевіряло доставку повідомлень та сповіщень у трьох сценаріях. При надсиланні повідомлення в чаті воно зберігається в таблиці Message та через WebSocket-подію new_message доставляється отримувачу та відображається у вікні чату без перезавантаження сторінки; затримка між надсиланням та отриманням не перевищила 1 секунди, тест пройдено успішно. При створенні нового заняття викладачем учень миттєво отримує WebSocket-сповіщення типу new_lesson з деталями заняття, лічильник непрочитаних сповіщень на панелі TopBar збільшується на 1; тест пройдено успішно. При виставленні оцінки учень отримує відповідне сповіщення в режимі реального часу без перезавантаження сторінки; тест пройдено успішно.

Тестування процесу адміністрування охоплювало чотири сценарії. При підтвердженні заявки адміністратором поле isActive облікового запису встановлюється у true, через сервіс Resend надсилається електронний лист з підтвердженням та посиланням для входу — лист доставлено, обліковий запис активовано, тест пройдено успішно. При відхиленні заявки система

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

надсилає лист з відповідним повідомленням, обліковий запис не активується; тест пройдено успішно. При надсиланні запрошення на співбесіду лист містить дату, час та посилання на відеоконференцію, сервіс Resend підтвердив доставку; тест пройдено успішно. При блокуванні облікового запису подальші спроби входу з його обліковими даними повертають відповідь зі статусом 403, доступ до захищених ресурсів неможливий; тест пройдено успішно.

Загальні результати тестування підтверджують, що всі 20 визначених тестових сценаріїв, які охоплюють автентифікацію, управління розкладом, оцінювання, домашні завдання, реальночасову комунікацію та адміністрування, пройдено успішно. Реалізована система відповідає функціональним вимогам, визначеним у першому розділі. Серверна перевірка конфліктів розкладу, алгоритм розрахунку прогресу навчання, механізм WebSocket-сповіщень та рольова модель доступу функціонують відповідно до проектних рішень, описаних у другому розділі, що підтверджує готовність платформи до практичного використання в умовах реальної онлайн-школи.

Висновки до розділу 3

У третьому розділі описано порядок розгортання системи в хмарному середовищі на платформах Supabase, Render та Netlify, а також порядок локального встановлення для розробки. Детально описано структуру інтерфейсу всіх сторінок платформи з урахуванням особливостей відображення для кожної ролі користувача. Проведено функціональне тестування системи за ключовими сценаріями використання — автентифікація, управління розкладом, оцінювання, домашні завдання, реальночасова комунікація та адміністрування. Всі визначені тестові сценарії пройдено успішно, що підтверджує відповідність реалізованої системи функціональним вимогам.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У випускній кваліфікаційній роботі розроблено вебплатформу організації навчального процесу онлайн-школи, що є повністю функціональним програмним продуктом, розгорнутим у хмарному середовищі.

Проведено аналіз предметної галузі та виявлено ключові проблеми організації навчального процесу в онлайн-школах при використанні традиційних інструментів — електронних таблиць, загальних месенджерів та ручного управління розкладом. Визначено функціональні вимоги до розроблюваної системи на основі потреб чотирьох типів учасників освітнього процесу.

Проведено порівняльний аналіз існуючих аналогів — Google Classroom, Moodle та Coursera. Встановлено, що розглянуті платформи орієнтовані на групове навчання у форматі курсів і не забезпечують необхідного рівня персоналізації для приватної онлайн-школи з індивідуальним розкладом для кожної пари учень-викладач. Визначено конкурентні переваги власної розробки.

Обґрунтовано вибір архітектурних рішень та інструментальних засобів реалізації. Система побудована на клієнт-серверній архітектурі з REST API та WebSocket на базі TypeScript, Node.js, Express.js, PostgreSQL, Prisma та React.

Спроектовано структуру реляційної бази даних з дев'ятьма таблицями, що охоплюють усі функціональні сутності платформи. Реалізовано чотирирівневу рольову модель з розподілом прав доступу між адміністратором, куратором, викладачем та учнем.

Реалізовано повний набір запланованих функціональних можливостей: управління розкладом занять з перевіркою конфліктів та підтримкою повторюваних занять, електронний журнал оцінок з автоматичним розрахунком прогресу навчання, система домашніх завдань з можливістю надсилання та перевірки виконаних робіт, внутрішній чат у

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

реальному часі на базі WebSocket, система автоматичних сповіщень, процес адміністрування користувачів із вступною співбесідою, завантаження аватарів у хмарне сховище Supabase Storage та автентифікація через Google OAuth 2.0.

Проведено функціональне тестування системи за двадцятьма тестовими сценаріями, що охоплюють усі ключові функціональні модулі платформи. Всі тестові сценарії пройдено успішно, що підтверджує відповідність реалізованої системи визначеним вимогам.

Розроблена вебплатформа розгорнута в хмарному середовищі та доступна для використання. Практична цінність результатів полягає в тому, що система може бути впроваджена в діяльність приватних онлайн-шкіл та репетиторських центрів для автоматизації управління навчальним процесом, підвищення ефективності роботи викладачів та кураторів і покращення комунікації між усіма учасниками освітнього процесу.

Перспективами подальшого розвитку системи є реалізація мобільного застосунку, інтеграція з платіжними системами для автоматизації оплати занять, розширення аналітичного модулю з побудовою детальних звітів про успішність та додавання підтримки відеоконференцій безпосередньо в межах платформи без використання зовнішніх сервісів.

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Help. About Classroom. URL: <https://support.google.com/edu/classroom/answer/6020279?hl=en>
2. Moodle Documentation. Features of the Moodle app. URL: https://docs.moodle.org/502/en/Moodle_app_features
3. Coursera. About Coursera. URL: <https://www.coursera.org/about>
4. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
5. Sommerville I. Software Engineering. 10th ed. London: Pearson Education, 2016. 816 p.
6. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Addison-Wesley Professional, 2012. 640 p.
7. Richardson L., Ruby S. RESTful Web Services. Sebastopol: O'Reilly Media, 2007. 446 p.
8. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. Internet Engineering Task Force, 2011. URL: <https://www.rfc-editor.org/rfc/rfc6455>
9. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. URL: <https://www.rfc-editor.org/rfc/rfc7519>
10. Codd E. F. A relational model of data for large shared data banks. Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387. DOI: 10.1145/362384.362685
11. Kleppmann M. Designing Data-Intensive Applications. Sebastopol: O'Reilly Media, 2017. 616 p.
12. Obe R., Hsu L. PostgreSQL Up and Running. 3rd ed. Sebastopol: O'Reilly Media, 2017. 346 p.
13. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. Sebastopol: O'Reilly Media, 2019. 322 p.

		Білий М.М.			Житомирська політехніка. 25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. 2nd ed. Shelter Island: Manning Publications, 2017. 392 p.
15. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol: O'Reilly Media, 2020. 280 p.
16. Пономаренко О. О., Ковальчук І. В. Ефективність застосування бібліотеки React // Інформаційні технології та суспільство. 2022. № 2(4). С. 13–19. DOI: 10.32689/maup.it.2022.2.2
17. Бачинський А. В. Розробка веб-календаря для створення нотаток-нагадувань на основі React JS з використанням TypeScript : кваліфікаційна робота освітнього рівня «Бакалавр» / А. В. Бачинський. Тернопіль : ТНТУ, 2023. 53 с. URL: <https://elartu.tntu.edu.ua/handle/lib/41879>
18. Сніпур Ю. В. Використання React та Node.js для реалізації web-додатку : кваліфікаційна робота освітнього рівня «Бакалавр» / Ю. В. Сніпур. Запоріжжя: ЗФЕУ, 2024. 67 с. URL: <http://library.econom.zp.ua:85/handle/123456789/338>
19. Аналіз стратегій покращення продуктивності реляційних систем керування базами даних на прикладі PostgreSQL : кваліфікаційна робота освітнього рівня «Бакалавр» / А. Б. Чекановський. Тернопіль: ТНТУ, 2024. URL: <https://elartu.tntu.edu.ua/handle/lib/47067>
20. Гордійчук В. Емпіричне порівняння ORM-систем Prisma, Sequelize та TypeORM для застосунків на основі Node.js та MySQL // Проблеми комп'ютерних наук, програмного моделювання та безпеки цифрових систем : матеріали міжнар. наук.-практ. конф. 2026. С. 74–75. URL: <https://apcssm.vnu.edu.ua/index.php/conf/article/view/318>
21. Зошак Т. А. Supabase як сучасна альтернатива Firebase для розробки веб-застосунків // Матеріали конф. Державного університету «Житомирська політехніка». Житомир, 2025. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/04/11.pdf>

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

22. Gupta S., Kumar A. Real-time Communication in Web Applications: Socket.IO Implementation Study. International Journal of Computer Applications. 2023. Vol. 185, No. 15. P. 32–41. DOI: 10.5120/ijca2023921547
23. Resend. Resend Email API for Developers. URL: <https://resend.com/docs/introduction>
24. Google Cloud. OAuth 2.0 for Web Server Applications. URL: <https://developers.google.com/identity/protocols/oauth2/web-server>
25. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol: O'Reilly Media, 2021. 616 p.
26. Wieringa R. Design Science Methodology for Information Systems and Software Engineering. Berlin: Springer, 2014. 332 p.
27. Pressman R., Maxim B. Software Engineering: A Practitioner's Approach. 8th ed. New York: McGraw-Hill Education, 2014. 976 p.
28. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.
29. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston: Addison-Wesley, 2005. 496 p.
30. Jacobson I., Booch G., Rumbaugh J. The Unified Software Development Process. Boston: Addison-Wesley, 1999. 512 p.
31. Fowler M. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2002. 560 p.
32. Nielsen J. Usability Engineering. San Francisco: Morgan Kaufmann, 1994. 362 p.
33. Grigorik I. High Performance Browser Networking. Sebastopol: O'Reilly Media, 2013. 396 p. URL: <https://hpbnp.co>

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмоторя О.Г.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Білий М.М.			Житомирська політехніка.25.121.03.000 - ПЗ	Арк.
		Чижмотря О.Г.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграма варіантів використання системи

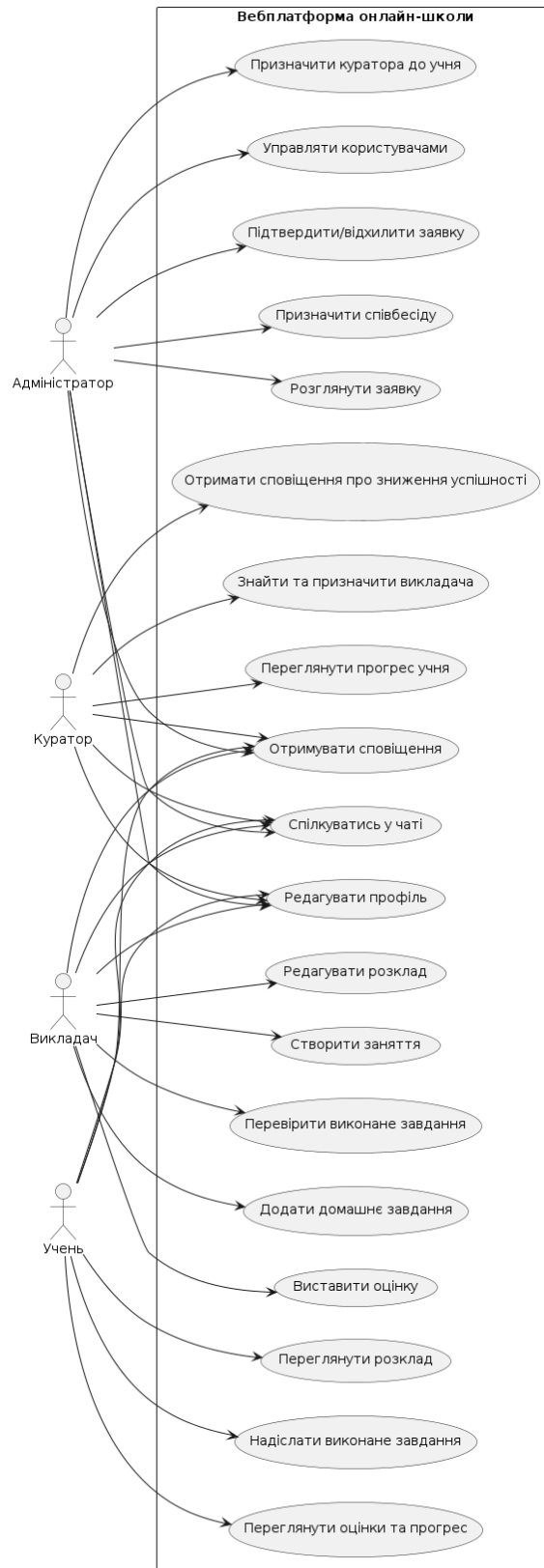


Рис. А.1. Діаграма варіантів використання системи

Схема бази даних додатка

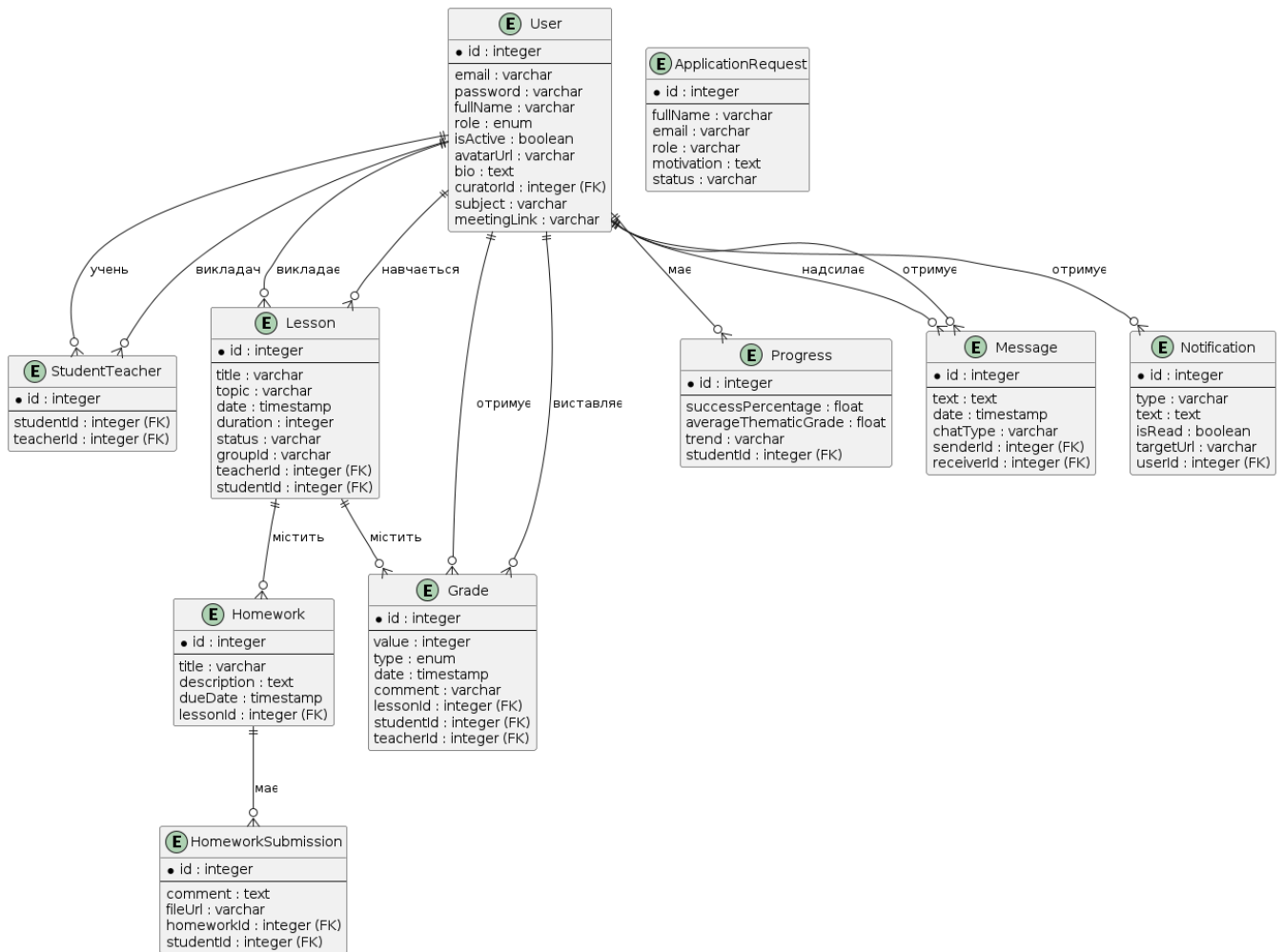


Рис. Б.1. Схема бази даних додатка

Діаграма послідовності створення заняття

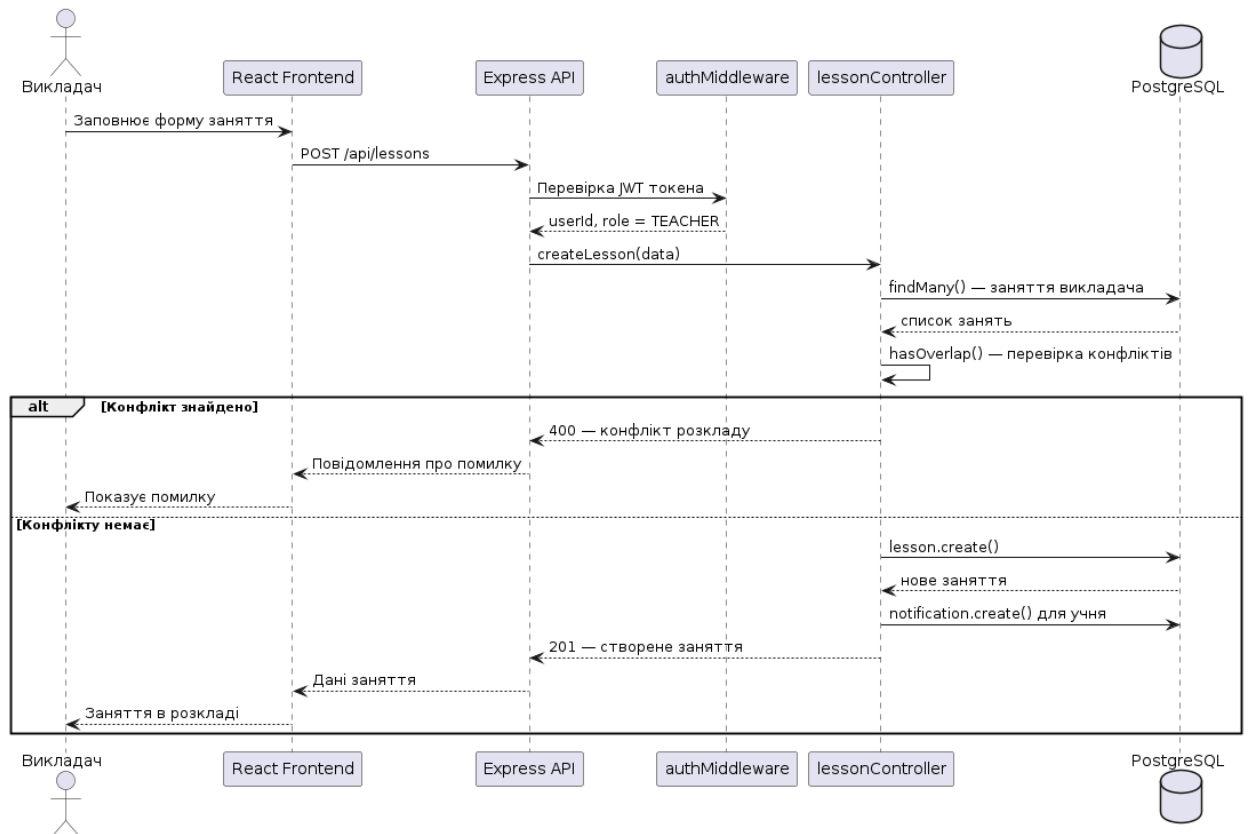


Рис. В.1. Діаграма послідовності створення заняття

Діаграма послідовності автентифікації

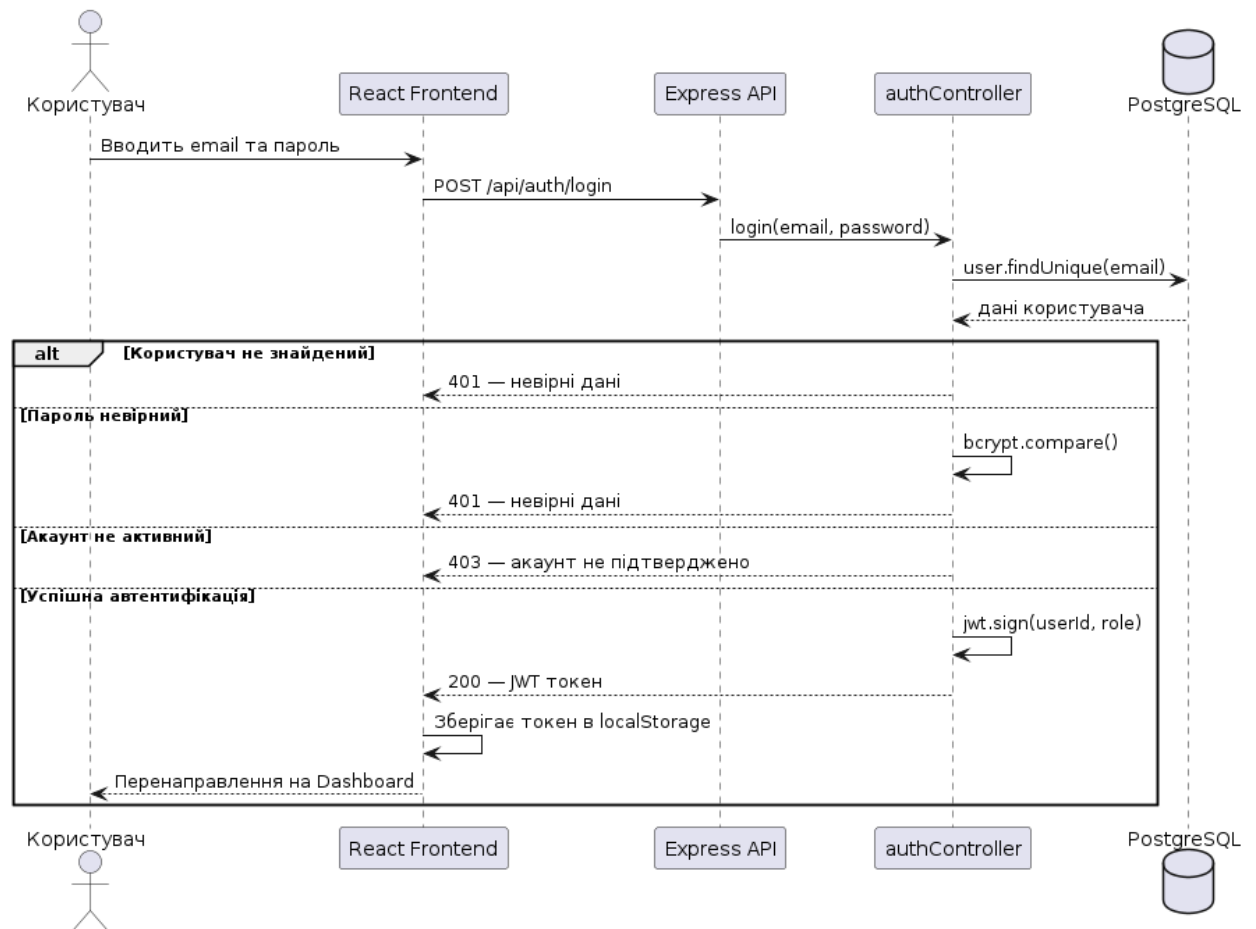


Рис. Г.1. Діаграма послідовності автентифікації

Prisma-схема бази даних

```

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id          Int          @id @default(autoincrement())
  email       String       @unique
  passwordHash String
  fullName    String
  role        String       @default("STUDENT")
  isActive    Boolean      @default(false)
  createdAt   DateTime     @default(now())
  updatedAt   DateTime     @updatedAt
  avatarUrl   String?
  bio         String?
  motivation  String?
  phone       String?
  telegramUsername String?
  viberUsername String?
  signalUsername String?
  subject     String?
  teachingDirections String?
  educationalPrograms String?
  teachingMethods String?
  experience   Int?
  meetingLink  String?
  googleId     String?     @unique
  interviewDate String?
  interviewTime String?
  interviewLink String?
  curatorId    Int?
  gradesReceived Grade[]   @relation("ReceivedGrades")
  gradesGiven   Grade[]   @relation("GivenGrades")
  homeworkSubmissions HomeworkSubmission[]
  lessonsAttended Lesson[] @relation("StudentLessons")
  lessonsTaught Lesson[]  @relation("TeacherLessons")
  receivedMessages Message[] @relation("ReceivedMessages")
  sentMessages Message[]  @relation("SentMessages")
  notifications Notification[]
  progress      Progress?
  curatorTeacher User?      @relation("CuratorStudents", fields:
[curatorId], references: [id])
  curatorStudents User[]    @relation("CuratorStudents")
  studentTeachers StudentTeacher[] @relation("StudentTeachers")
  teacherStudentsMany StudentTeacher[] @relation("TeacherStudentsMany")
}

model StudentTeacher {
  studentId Int
  teacherId Int
  student User @relation("StudentTeachers", fields: [studentId], references: [id],
onDelete: Cascade)
  teacher User @relation("TeacherStudentsMany", fields: [teacherId], references:
[id], onDelete: Cascade)

  @@id([studentId, teacherId])
}

model Lesson {

```

```

    id          Int          @id @default(autoincrement())
    title       String
    topic       String
    date        DateTime
    duration    Int
    status      String?
    teacherId   Int
    studentId   Int
    createdAt   DateTime     @default(now())
    updatedAt   DateTime     @updatedAt
    groupId     String?
    grades      Grade[]
    homeworks   Homework[]
    student     User          @relation("StudentLessons", fields: [studentId],
references: [id])
    teacher     User          @relation("TeacherLessons", fields: [teacherId],
references: [id])
    materials   LessonMaterial[]
}

model Grade {
  id          Int          @id @default(autoincrement())
  value       Float
  type        String       @default("CURRENT")
  comment     String?
  date        DateTime     @default(now())
  studentId   Int
  lessonId    Int
  teacherId   Int
  homeworkId  Int?         @unique
  homework    Homework?    @relation(fields: [homeworkId], references: [id])
  lesson      Lesson       @relation(fields: [lessonId], references: [id], onDelete:
Cascade)
  student     User          @relation("ReceivedGrades", fields: [studentId], references:
[id])
  teacher     User          @relation("GivenGrades", fields: [teacherId], references: [id])
}

model LessonMaterial {
  id          Int          @id @default(autoincrement())
  title       String
  type        String
  fileUrl     String
  description  String?
  uploadDate  DateTime     @default(now())
  lessonId    Int
  lesson      Lesson       @relation(fields: [lessonId], references: [id], onDelete:
Cascade)
}

model Homework {
  id          Int          @id @default(autoincrement())
  taskDescription String
  deadline    DateTime
  attachments String        @default("")
  status      String        @default("NOT_STARTED")
  lessonId    Int
  grade       Grade?
  lesson      Lesson        @relation(fields: [lessonId], references: [id],
onDelete: Cascade)
  submissions HomeworkSubmission[]
}

model HomeworkSubmission {
  id          Int          @id @default(autoincrement())
  homeworkId  Int
  studentId   Int
  text        String?
  fileUrl     String?

```

```

    status      String    @default("DONE_NO_QUESTIONS")
    submittedAt DateTime @default(now())
    homework    Homework  @relation(fields: [homeworkId], references: [id], onDelete:
Cascade)
    student     User      @relation(fields: [studentId], references: [id])
}

model Progress {
  id            Int        @id @default(autoincrement())
  successPercentage Float
  averageThematicGrade Float
  trend         String     @default("STABLE")
  lastUpdated   DateTime @default(now())
  studentId     Int        @unique
  student       User      @relation(fields: [studentId], references: [id])
}

model Message {
  id            Int        @id @default(autoincrement())
  text          String
  sentDate      DateTime @default(now())
  isRead        Boolean   @default(false)
  chatType      String
  senderId      Int
  receiverId    Int
  receiver      User      @relation("ReceivedMessages", fields: [receiverId], references:
[id])
  sender        User      @relation("SentMessages", fields: [senderId], references: [id])
}

model Notification {
  id            Int        @id @default(autoincrement())
  type          String
  text          String
  date          DateTime @default(now())
  isViewed      Boolean   @default(false)
  targetUrl     String?
  userId        Int
  user          User      @relation(fields: [userId], references: [id])
}

```