Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБСАЙТУ «TEACHERS HUB» ДЛЯ ВЧИТЕЛІВ АНГЛІЙСЬКОЇ МОВИ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1

Помирчий Я.М.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Помирчому Ярославу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту «Teachers Hub» для вчителів англійської мови

затверджена наказом по університету від 19 травня 2025 року № 381Ст

2. Термін подання студентом роботи до екзаменаційної комісії 01 червня 2025 р.

3. Вихідні дані до роботи структура та функціональність інформаційних систем для освітніх потреб, перелік сучасних вебтехнологій: теоретичні відомості про роботу з базами даних, інтерфейсами користувача, механізмами автентифікації та принципами побудови клієнт-серверної архітектури без виділеного backend-сервера.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд існуючих інформаційних систем для організації навчального процесу та їхніх функціональних можливостей.

2. Вибір технологічного стеку для розробки вебзастосунку Teachers Hub та обґрунтування його доцільності.

3. Проєктування архітектури системи: моделі даних, взаємодія між компонентами, розмежування прав доступу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) схема архітектури вебзастосунку Teachers Hub, структура бази даних Supabase із вказанням основних таблиць (студенти, уроки), результати роботи застосунку на прикладі тестового користувача (створення уроку, додавання оплати, перегляд домашнього завдання).

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на плагіат	21.05.25-01.06.25	
9	Рецензування	21.05.25-01.06.25	
10	Підготовка презентації та доповіді	21.05.25-01.06.25	
11	Занесення роботи в електронний архів	21.05.25-18.06.25	
12	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тітова О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 54 с., 3 табл., 12 рис., 30 джерел.

TEACHERS HUB, CRM, NEXT.JS, TYPESCRIPT, SUPABASE, CLERK, УПРАВЛІННЯ УРОКАМИ, СТУДЕНТАМИ, СТАТУС ОПЛАТИ, РОЛІ КОРИСТУВАЧІВ.

Об’єктом роботи є процес організації навчального процесу у приватних мовних школах.

Метою роботи є створення вебзастосунку для керування студентами, уроками, оплатами та іншими аспектами навчального процесу на прикладі Teachers Hub.

У рамках роботи проведено аналіз сучасних цифрових рішень у сфері освітніх CRM-систем, зокрема Off2Class, Planboard та MySchool, та визначено їхні переваги й обмеження у контексті приватного викладання англійської мови.

У результаті роботи створено працюючий застосунок Teachers Hub із зручною панеллю для викладача та адміністратора.

TEACHERS HUB, CRM, NEXT.JS, TYPESCRIPT, SUPABASE, CLERK, LESSON MANAGEMENT, STUDENT MANAGEMENT, PAYMENT TRACKING, USER ROLES.

The object of this project is the process of organizing the educational workflow in private language schools.

The aim of the project is to develop a web application for managing students, lessons, payments, and other aspects of the teaching process, using Teachers Hub.

As part of the project, an analysis of modern digital solutions in the field of educational CRM systems was conducted, including platforms such as Off2Class, Planboard, and MySchool. Their advantages and limitations in the context of private English teaching were identified.

As a result, a fully functioning application – Teachers Hub – was developed to support language teachers in their administrative tasks.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 4
Вступ 5
1 Аналіз існуючих рішень для керування студентами та уроками і порівняння сучасних вебзастосунків для вчителів англійської мови 7
1.1 Огляд застосунків для керування студентами, уроками та оплатами .. 7
1.2 Порівняльний аналіз методів керування студентами та уроками 12
1.2.1 Використання стандартних інструментів (Google Calendar, Excel) 13
1.2.2 Використання готових CRM-систем 15
1.3 Порівняльний аналіз існуючих рішень для керування студентами та уроками 20
1.4 Постановка задачі 22
2 Teachers Hub 24
2.1 Аналіз використаних інструментів для реалізації застосунку 24
2.2 Моделі для інтеграції та передачі даних у застосунку Teachers Hub 29
2.2.1 Моделі використані для API Teachers Hub 30
2.2.2 Моделі використані у застосунку Teachers Hub 31
2.2.3 Моделі для організації поширення даних всередині застосунку Teachers Hub 33
3 Програмна реалізація вебзастосунку Teachers Hub 35
3.1 Обґрунтування вибору середовища 35
3.2 Програмна реалізація застосунку Teachers Hub 39
3.3 Публікація та розгортання застосунку Teachers Hub 43
Висновки 46
Перелік джерел посилання 47
Додаток А 50

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CRM – система для обліку клієнтів та керування робочими процесами Supabase – хмарна СУБД на основі PostgreSQL

Clerk – сервіс авторизації з підтримкою ролей Next.js – фреймворк для розробки вебзастосунків UI (User Interface) – інтерфейс користувача

Frontend – візуальна частина застосунку

Backend – логічна частина, відповідальна за дані та обробку запитів Lesson Status – статус уроку (Upcoming, Cancelled, Completed) Payment Status – статус оплати (Paid, Not Paid)

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій онлайн-освіта посідає дедалі важливіше місце у сфері навчання. Особливо помітний стрибок у цьому напрямі стався після пандемії COVID-19, яка суттєво трансформувала формат взаємодії між викладачами та студентами. Онлайн-заняття стали не винятком, а нормою, і дедалі більше приватних викладачів англійської мови почали працювати дистанційно – як самостійно, так і в межах невеликих мовних шкіл або онлайн-платформ.

У результаті такої трансформації виникла нагальна потреба в ефективних цифрових інструментах, які б дозволяли організовувати навчальний процес на якісно новому рівні. Ідеться не лише про зручний розклад занять, а й про ведення обліку студентів, контроль оплат, призначення домашніх завдань, збереження історії занять, а також централізовану систему доступу до інформації як для викладачів, так і для адміністрації школи. Зростання кількості студентів, підвищення очікувань щодо якості навчального сервісу та потреба в автоматизації базових процесів створили запит на більш системне рішення.

Найпростішим підходом до організації освітнього процесу для багатьох викладачів досі залишається використання комбінації стандартних інструментів – Google Calendar для планування занять, Google Sheets або Microsoft Excel для обліку студентів і фінансів, а також месенджерів для комунікації. Проте з часом така структура починає виявляти свої обмеження. Відсутність централізованої системи, розрізненість даних і неможливість масштабування призводять до плутанини, втрати часу та зростання навантаження на викладача.

На ринку існує низка готових рішень, таких як Planboard, MySchool або Off2Class, які пропонують комплексні платформи для управління навчальним процесом. Однак більшість із них мають суттєві недоліки в контексті потреб невеликих мовних шкіл – надмірну складність, високу вартість ліцензій,

обмежену можливість адаптації інтерфейсу та часто відсутність підтримки української мови. Впровадження таких систем потребує додаткового навчання персоналу, а також регулярних фінансових витрат, що не завжди виправдано для маленьких команд.

У зв’язку з цим виникла ідея розробити власне рішення – вебзастосунок Teachers Hub, який відповідав би специфічним вимогам невеликих освітніх колективів, залишаючись водночас простим, надійним і масштабованим. Teachers Hub створено з використанням сучасного вебстеку, що включає Next.js з App Router для інтерфейсної частини [1], мову програмування TypeScript для типобезпечної логіки [2], Supabase як основу для зберігання та обробки даних [3], а також сервіс Clerk для автентифікації та керування доступом [4]. Такий вибір забезпечив швидкість розробки, захищеність даних та зручність для користувачів з різним рівнем цифрової компетентності.

У межах цього дослідження розглядається не лише технічна реалізація застосунку, а й обґрунтовується його необхідність, аналізуються існуючі альтернативи, формулюються функціональні вимоги, а також демонструється, яким чином Teachers Hub оптимізує повсякденну роботу викладачів і адміністрації мовної школи. Проєкт має потенціал для подальшого розвитку, включаючи інтеграцію з платіжними системами, мобільну версію, розширену аналітику та підтримку нових типів навчальних матеріалів.

Розробка Teachers Hub є актуальним прикладом поєднання сучасних технологій із реальними потребами освітньої галузі. Застосунок дозволяє зробити процес викладання більш системним, контрольованим і менш затратним з точки зору організації, що особливо важливо в умовах динамічного цифрового середовища.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ КЕРУВАННЯ СТУДЕНТАМИ ТА УРОКАМИ І ПОРІВНЯННЯ СУЧАСНИХ ВЕБЗАСТОСУНКІВ ДЛЯ ВЧИТЕЛІВ АНГЛІЙСЬКОЇ МОВИ

1.1 Огляд застосунків для керування студентами, уроками та оплатами

У сучасному освітньому середовищі існує велика кількість застосунків та систем, що допомагають організувати навчальний процес, вести облік студентів, планувати розклад занять та контролювати фінансові операції.

Особливо актуальним це стало для приватних шкіл та індивідуальних викладачів, які часто шукають прості, ефективні та доступні за вартістю рішення.

Серед основних платформ, які використовуються в освіті для управління навчальним процесом, можна виділити Planboard, Off2Class та MySchool.

Кожен із цих застосунків має власні переваги та недоліки, залежно від масштабу навчального закладу, кількості студентів, типу занять та інших факторів.

У цьому розділі буде проведено огляд зазначених сервісів та обґрунтування доцільності розробки власного кастомного рішення – Teachers Hub.

1.1.1 Planboard

Planboard – це онлайн-платформа для планування навчальних занять, яка дозволяє викладачам створювати уроки, формувати календарі, додавати освітні стандарти, нотатки, а також прикріплювати матеріали до кожного заняття [5]. Застосунок був розроблений компанією Chalk і став популярним серед шкільних вчителів у Північній Америці, особливо в Канаді та США. Його інтерфейс орієнтований на щоденне або тижневе планування, з

можливістю структурувати навчальний процес по класах, предметах або навчальних темах. Це дозволяє створювати структуровані навчальні плани, які можна використовувати повторно, а також ділитися ними з іншими викладачами в межах закладу або спільноти користувачів Chalk.

Однією з найбільш зручних функцій Planboard є можливість дублювати уроки або переносити їх між класами, що значно зменшує час, витрачений на підготовку розкладу. Завдяки підтримці різних рівнів організації – наприклад, навчальних блоків, тижнів і днів – користувачі можуть адаптувати систему до власного навчального графіка. Планування можна здійснювати не лише у звичному календарному вигляді, а й із додаванням інструкцій до кожного заняття, з посиланнями на зовнішні ресурси, тестові матеріали або вправи. До кожного уроку можна приєднувати файли у форматі PDF, презентації, таблиці та інші типи документів, що значно розширює функціональність підготовки до занять.

Крім цього, Planboard має інтеграцію з Google Classroom, що дозволяє синхронізувати теми уроків і матеріали безпосередньо з акаунтами учнів. Така інтеграція дає змогу автоматично призначати домашні завдання, відстежувати виконання завдань учнями та обмінюватися з ними файлами. Це зручно для вчителів, які активно працюють у гібридному або дистанційному форматі.

Однак, незважаючи на велику кількість зручних інструментів для планування навчального процесу, Planboard має і низку обмежень, які знижують його придатність для використання в приватному викладанні або в невеликих освітніх командах. Наприклад, система не дозволяє здійснювати облік оплат студентів, не має вбудованої фінансової системи або модуля для автоматизованого виставлення рахунків. Також відсутній облік відвідуваності, що є критично важливим у контексті приватних уроків або індивідуальних занять. Викладач не може зафіксувати статус уроку – чи він відбувся, був скасований або перенесений, що створює додаткову потребу у веденні стороннього обліку.

Ще одним обмеженням є відсутність персоналізованих кабінетів для

студентів. Усі дані фіксуються виключно з боку викладача, а студенти не мають можливості самостійно переглядати розклад занять, отримувати нагадування або залишати зворотний зв’язок. Це ускладнює двосторонню комунікацію і є незручним у випадках, коли навчання відбувається індивідуально або в межах невеликих груп.

Крім того, хоча основна функціональність Planboard доступна безкоштовно, деякі розширені можливості (зокрема, аналітика, експорт у PDF, розширені фільтри і додаткові налаштування календаря) надаються тільки користувачам з платною підпискою Chalk Gold. Це може бути додатковим бар’єром для індивідуальних викладачів або невеликих шкіл, особливо якщо вони шукають повністю безкоштовне рішення.

1.1.2 Off2Class

Off2Class – це спеціалізована освітня платформа, розроблена для підтримки викладання англійської мови як іноземної (ESL), що охоплює як організацію навчального процесу, так і надання готових навчальних матеріалів [6]. Основна аудиторія платформи – індивідуальні викладачі та малі освітні центри, які надають персоналізовані заняття англійською мовою.

Однією з головних переваг Off2Class є наявність великої бази готових інтерактивних уроків, структурованих за рівнями складності відповідно до Загальноєвропейських рекомендацій з мовної освіти (CEFR). Це дозволяє викладачам швидко адаптувати навчальні плани під конкретного студента, не витрачаючи багато часу на підготовку до заняття. Крім того, у платформі передбачена система діагностичних тестів, яка допомагає визначити початковий рівень володіння мовою та відстежувати прогрес студента протягом усього періоду навчання.

Кожен студент має персональний кабінет, через який він може переглядати домашні завдання, результати тестів і рекомендовані теми для

наступних занять. Викладач, зі свого боку, отримує доступ до зручної панелі керування, де можна переглядати історію навчання кожного учня, призначати матеріали, оцінювати виконання завдань та формувати індивідуальні траєкторії розвитку.

Разом з тим, Off2Class має і ряд обмежень. Одним з основних недоліків є висока вартість підписки – в середньому від $8 до $10 на одного активного студента щомісячно. Для великих груп або викладачів із численною базою учнів це може стати фінансово невигідним. Крім того, інтерфейс платформи має досить жорстку структуру: викладачі не мають змоги повноцінно кастомізувати зовнішній вигляд інтерфейсу або змінювати логіку роботи системи відповідно до власних потреб.

Ще однією проблемою є значна залежність від готових шаблонів уроків. Хоча це зручно для новачків, досвідчені викладачі, які прагнуть розробляти авторські програми, можуть відчути нестачу гнучкості. Також варто зазначити, що Off2Class орієнтований виключно на викладання англійської мови, а тому не підходить для освітніх закладів, які надають послуги з інших предметів.

Загалом, Off2Class – це якісний інструмент для професійної організації викладання англійської мови, який може значно спростити роботу викладача та забезпечити високу ефективність навчального процесу. Проте обмеження у функціональності, висока вартість та відсутність повної кастомізації зменшують його привабливість для більшості приватних викладачів, які шукають універсальне, доступне та гнучке рішення.

1.1.3 MySchool

MySchool – це комплексна CRM-система, яка призначена для управління освітніми закладами повного циклу, зокрема школами, коледжами та приватними навчальними центрами [7]. Система охоплює всі ключові аспекти адміністративної та академічної діяльності закладу – від обліку студентів до

автоматизації оплати та формування академічної звітності.

Однією з головних переваг MySchool є її функціональна глибина. Платформа дозволяє вести облік як студентів, так і їхніх батьків або опікунів, надаючи інструменти для створення особистих профілів, відображення контактних даних, історії навчання, відвідуваності та оцінок. Окрім цього, система підтримує гнучке керування розкладом занять – з можливістю автоматичного формування, редагування та перегляду у форматі календаря або списку. Завдяки інтеграції з платіжними сервісами, MySchool також надає можливість автоматизувати виставлення рахунків, відстеження оплат та формування фінансової звітності, що особливо важливо для великих навчальних закладів.

Ще однією важливою особливістю є підтримка внутрішньої комунікації – викладачі, студенти та адміністрація можуть обмінюватись повідомленнями, отримувати сповіщення про майбутні події, зміни в розкладі або нові домашні завдання. Це робить систему корисною не лише для адміністративного обліку, а й для покращення взаємодії всіх учасників освітнього процесу.

Разом з тим, попри широкий функціонал, MySchool має низку суттєвих обмежень для невеликих шкіл або індивідуальних викладачів. Насамперед, система потребує складного первинного налаштування. Для повноцінної роботи необхідно визначити шаблони предметів, класи, зв’язки між студентами та викладачами, політики доступу та структуру навчального року. Це значно ускладнює використання платформи без залучення технічних спеціалістів або розробників.

Ще одним бар’єром є вартість. Ліцензія на використання MySchool розрахована здебільшого на повноцінні навчальні заклади й передбачає щомісячну або річну оплату за кількістю студентів або викладачів. Для малих команд або приватних освітніх проєктів це може бути економічно недоцільно. Крім того, платформа орієнтована переважно на роботу з класичними навчальними структурами (класи, семестри, контрольні), тому її адаптація до гнучкого освітнього формату – як-от індивідуальні заняття або короткі курси

– потребує додаткових зусиль.

Важливо також зазначити, що кастомізація інтерфейсу або логіки роботи в MySchool є обмеженою. Користувачі не можуть вільно змінювати вигляд сторінок або налаштовувати поведінку системи без доступу до внутрішнього коду. Це створює додаткові труднощі для команд, які прагнуть налаштувати платформу під специфіку свого освітнього процесу.

У підсумку, MySchool є ефективним інструментом для повноцінних навчальних закладів із великою кількістю користувачів і розгалуженою адміністративною структурою. Однак через складність налаштування, високу вартість та обмежену гнучкість вона не підходить для індивідуальних викладачів, репетиторів або невеликих мовних шкіл, які потребують простих і адаптивних рішень.

1.2 Порівняльний аналіз методів керування студентами та уроками

Для ефективної організації навчального процесу приватним викладачам та малим мовним школам необхідно обирати оптимальні інструменти, які дозволяють не лише планувати уроки, а й зручно вести облік студентів, контролювати оплату занять, зберігати навчальні матеріали та забезпечувати комунікацію з учнями. У сучасних умовах наявність такої системи є критично важливою – адже навіть при невеликій кількості студентів виникає потреба в структурованій системі обліку, особливо коли мова йде про онлайн-навчання, гнучкі графіки та індивідуальні траєкторії учнів.

На практиці існує кілька поширених підходів до організації такого процесу. Перший – це використання загальних офісних інструментів, зокрема Google Calendar, Google Sheets, Excel або Trello. Вони дозволяють вручну фіксувати розклад, статуси уроків і записи про студентів, однак не забезпечують автоматизації процесів, пов’язаних із оплатою або фільтрацією даних. Такий підхід зручний на початковому етапі, але стає проблематичним

при збільшенні навантаження або масштабуванні.

Другий підхід – це використання готових CRM-систем, спеціально розроблених для потреб освітніх закладів. Ці системи, як правило, пропонують набір типових функцій, таких як створення профілів студентів, управління розкладом, формування звітності та контроль оплати. Вони мають певний рівень зручності, особливо коли йдеться про командну роботу або групові заняття. Проте варто враховувати, що більшість з них є платними, обмежено адаптуються до індивідуальних потреб і часто мають складні інтерфейси, що потребують додаткового навчання персоналу.

І, нарешті, третій підхід – це створення власного кастомного рішення, яке враховує специфіку конкретної школи або викладача. Такий підхід дозволяє реалізувати саме той функціонал, який дійсно необхідний у конкретному випадку, і відмовитись від зайвих або незручних компонентів. Прикладом такого рішення є розробка вебзастосунку Teachers Hub – системи, яка була створена з нуля для підтримки приватного викладання англійської мови. У ній враховані потреби як викладача, так і адміністратора, реалізовано підтримку ролей, зручний календар занять, систему обліку оплат, домашні завдання та аналітику.

У цьому розділі буде докладно проаналізовано три підходи до керування навчальним процесом: використання універсальних офісних інструментів, впровадження готових освітніх CRM та розробка власного рішення, зокрема на прикладі Teachers Hub. Кожен з підходів має свої сильні й слабкі сторони, і вибір залежить від масштабу проєкту, бюджету та технічних можливостей команди.

1.2.1 Використання стандартних інструментів (Google Calendar, Excel)

Одним із найпоширеніших підходів до організації навчального процесу серед індивідуальних викладачів і невеликих шкіл є використання

загальнодоступних офісних інструментів. Найчастіше в цьому контексті використовуються Google Calendar для створення та керування розкладом занять, а також Excel або Google Sheets для ведення баз даних студентів, фіксації оплат, відвідуваності та прогресу [8].

Цей підхід має кілька суттєвих переваг, головною з яких є простота впровадження. Такі інструменти не потребують глибоких технічних знань чи спеціального навчання – більшість викладачів уже знайомі з ними або можуть швидко опанувати необхідний функціонал. Google Calendar, наприклад, дозволяє налаштовувати регулярні події (заняття), додавати нагадування, прив'язувати посилання на Zoom або Google Meet, що є особливо корисним для дистанційного навчання. Інтеграція з мобільними додатками забезпечує доступ до розкладу з будь-якого пристрою, а функція спільного доступу дозволяє координувати час між викладачем і студентом.

Google Sheets або Excel часто використовуються як основа для внутрішнього обліку – у вигляді таблиць, де зберігається інформація про студентів (ім'я, контактні дані, рівень), уроки (дати, тривалість, тема), фінанси (суму оплати, дату платежу, статус). Такі таблиці можуть бути кастомізовані під потреби конкретного викладача – із фільтрами, умовним форматуванням, формулами для автоматичних розрахунків тощо.

Однак попри зручність на старті, цей підхід має низку обмежень, які стають критичними при зростанні кількості студентів або ускладненні процесів. Насамперед, мова йде про відсутність централізації та автоматизації**.** Дані, що розкидані по різних файлах або вкладках, не синхронізуються автоматично, що збільшує ризик помилок, дублювання або втрати інформації. Якщо викладач веде окремі таблиці для оплати, розкладу та домашніх завдань – це вимагає ручного переносу та постійного оновлення, що не лише займає час, а й створює навантаження.

Ще однією слабкою стороною є відсутність системи сповіщень і взаємодії з користувачами. Наприклад, студент не може самостійно перевірити інформацію про свою оплату або доступні матеріали – усе це передається

вручну, часто через месенджери. У випадку великої кількості учнів або викладачів це спричиняє плутанину в комунікації, затримки з оплатами або непорозуміння щодо графіку.

Важливо також зазначити, що стандартні інструменти не підтримують ролі користувачів, а отже, не можна організувати обмеження доступу для адміністратора, викладача та студента. Усі дані доступні всім, хто має посилання, або, навпаки, зберігаються локально на комп'ютері викладача, що є ризиковано з точки зору безпеки та збереження інформації.

Така система практично не масштабується. Коли викладач переходить від 5–10 учнів до 20–30, або коли виникає потреба взаємодії кількох викладачів у межах однієї школи – таблиці перестають бути ефективними. Відсутність інтеграції між розкладом, оплатами, профілями студентів і завданнями ускладнює щоденну роботу, створює дублювання записів і часто призводить до втрати контролю над процесами.

Використання стандартних офісних інструментів – це оптимальний варіант для старту, коли кількість студентів незначна, а процеси ще не автоматизовані. Проте в середньостроковій та довгостроковій перспективі цей підхід не є стійким і вимагає переходу до більш структурованого рішення – такого як спеціалізована CRM-система або власний вебзастосунок. Власне, саме такі обмеження і стали підставою для розробки кастомного рішення Teachers Hub.

1.2.2 Використання готових CRM-систем

Спеціалізовані CRM-системи, такі як Off2Class, Planboard або MySchool, надають широкі можливості для структурованої організації освітнього процесу в навчальних закладах різного масштабу [9]. Їх головна особливість полягає у створенні цілісного цифрового середовища, де всі основні функції – від створення профілів студентів і розкладу занять до автоматизованого

управління фінансовими операціями – поєднані в єдину систему. Завдяки цьому викладачі, адміністрація та, в деяких випадках, навіть студенти можуть взаємодіяти з платформою в межах визначених прав доступу.

Такі CRM часто містять вбудовані шаблони для створення домашніх завдань, інструменти для виставлення оцінок, ведення журналу відвідуваності, інтеграцію з месенджерами або email для нагадувань, а також модулі для роботи з батьками (у загальноосвітньому контексті). Частина з них дозволяє прикріплювати матеріали до кожного уроку, що дає змогу формувати базу навчального контенту. У деяких системах реалізовано базову аналітику – статистику прогресу, візуальні графіки відвідуваності, динаміку успішності студентів. Це створює умови для більш інформованого прийняття рішень щодо подальшого навчального маршруту.

До беззаперечних переваг належить і те, що впровадження таких систем не потребує від викладачів чи адміністраторів глибоких знань у сфері програмування або баз даних – платформи вже містять усі необхідні інструменти, оформлені у вигляді готового до використання інтерфейсу. Більшість з них постачається з технічною підтримкою, а також з оновленнями, які автоматично інтегруються в роботу без потреби ручного втручання.

Однак навіть при значному наборі функцій ці системи мають обмеження. Насамперед – це вартість, яка може виявитися критичною для невеликих приватних шкіл або індивідуальних викладачів. Місячна підписка на одного учня часто перевищує $8–10, що при кількості студентів понад 20 вже створює суттєве навантаження на бюджет. Додатково виникають труднощі із адаптацією системи до індивідуальних потреб школи – не всі CRM підтримують кастомізацію полів, структури інтерфейсу або бізнес-логіки. Це може стати проблемою, якщо навчальний заклад має специфічну модель організації занять, яка не вписується в типову структуру системи.

Ще один виклик полягає в навчанні персоналу – незважаючи на наявність техпідтримки, освоєння навіть інтуїтивно зрозумілого інтерфейсу потребує часу. Якщо платформа не локалізована українською мовою, це

ускладнює її використання у вітчизняних умовах. Також трапляються випадки, коли частина функціоналу, потрібна викладачеві, доступна лише у старших тарифах або корпоративних планах.

У результаті такі CRM краще підходять для великих або середніх закладів із налагодженими процесами, ресурсами для технічної адаптації та потребою в повному циклі цифрового управління. Для малих мовних шкіл або самозайнятих викладачів подібні системи часто виявляються надлишковими або фінансово недоцільними. У таких випадках доцільно звернути увагу на створення власного кастомного рішення, яке точно відповідатиме реальним потребам та навантаженню.

1.2.3 Розробка власного кастомного рішення (Teachers Hub)

Одним із найбільш гнучких і стратегічно доцільних підходів до організації освітнього процесу є створення власного кастомного вебзастосунку, який враховує специфіку роботи конкретного навчального закладу або викладацької практики. Яскравим прикладом такого підходу є розроблений у межах даної роботи застосунок Teachers Hub – цифрова платформа, що забезпечує повний цикл керування навчанням у мовній школі.

Створення власного інструменту дозволяє уникнути обмежень, притаманних готовим CRM-системам. У випадку Teachers Hub, архітектура та логіка системи розроблені таким чином, щоб максимально відповідати реальним щоденним завданням викладача та адміністратора. Застосунок підтримує реєстрацію та управління студентами з можливістю зберігати детальну інформацію: ім’я, рівень знань, регулярність занять, опис, замітки тощо. Кожен урок можна планувати та фіксувати зі статусом «заплановано», «проведено» або «скасовано», що значно полегшує звітність та аналіз відвідуваності. Також реалізовано систему обліку оплат, де для кожного заняття можна вказувати статус платежу: сплачено, не сплачено, очікує

підтвердження.

Teachers Hub підтримує рольову модель доступу – у системі існують два типи користувачів: викладач і адміністратор. Розмежування прав доступу дозволяє забезпечити контроль над даними, уникнути несанкціонованих змін і забезпечити конфіденційність фінансової інформації. Викладачі мають доступ лише до власних студентів, уроків і фінансів, тоді як адміністратор може бачити повну статистику всієї школи, змінювати статуси уроків, керувати викладачами й переглядати аналітику.

Ще однією перевагою кастомного підходу є повна інтеграція з безпечними та сучасними технологіями зберігання та обробки даних. У межах Teachers Hub для цих цілей було обрано Supabase – хмарну платформу на основі PostgreSQL, яка забезпечує надійне зберігання даних та підтримку тонкої системи доступу (Row-Level Security). У поєднанні з сервісом Clerk, який відповідає за автентифікацію та авторизацію, це створює захищене середовище для обробки конфіденційної інформації. Система не потребує класичного серверного бекенду, оскільки вся логіка інтегрована в клієнтський інтерфейс через Supabase SDK. Це дозволяє пришвидшити розробку та зменшити складність підтримки.

Серед важливих переваг Teachers Hub – можливість адаптації системи під будь-який сценарій. Наприклад, можна легко додати нові функції: управління словником, фіксація домашніх завдань, завантаження файлів, аналітика по кожному студенту тощо. Власне рішення не обмежується рамками тарифного плану чи шаблонного інтерфейсу – кожен елемент інтерфейсу або функціоналу може бути змінений відповідно до потреб конкретної школи.

Проте, варто враховувати, що створення кастомного рішення вимагає наявності певних ресурсів – як технічних, так і часових. Для запуску системи необхідні розробники, які володіють сучасним стеком технологій (Next.js, TypeScript, Supabase, Clerk), а також час на проектування, розробку, тестування та публікацію. Після запуску потрібно забезпечити технічну

підтримку, оновлення системи та можливу інтеграцію з іншими сервісами (наприклад, платіжними або аналітичними).

Незважаючи на початкові витрати, власна система, така як Teachers Hub, є економічно вигідною у довгостроковій перспективі. Вона не передбачає абонплати або залежності від сторонніх сервісів, а також дозволяє повністю контролювати як дані, так і спосіб їх використання.

Створення власного застосунку, такого як Teachers Hub, є найбільш доцільним варіантом для невеликих мовних шкіл, які прагнуть ефективно керувати навчальним процесом, не витрачаючи зайвих коштів на сторонні сервіси (рис. 1.1 – 1.2).

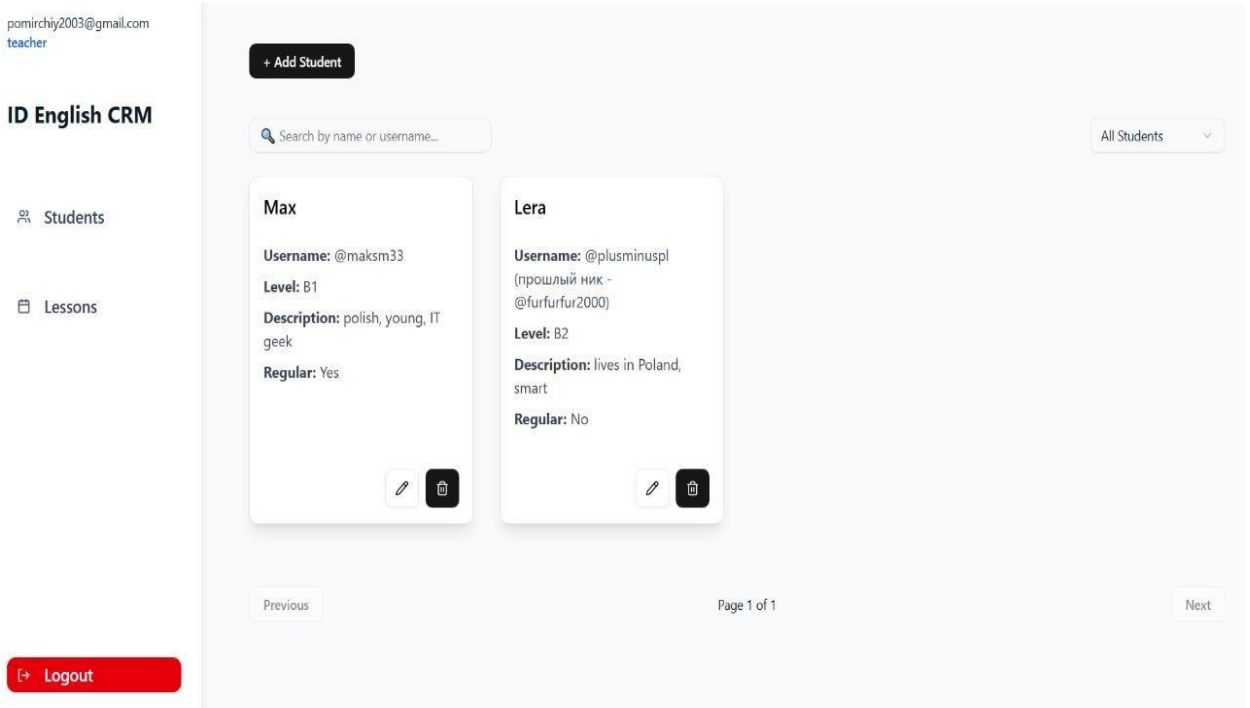

Рисунок 1.1 – Інтерфейс Teachers Hub: головний екран викладача з управлінням студентами

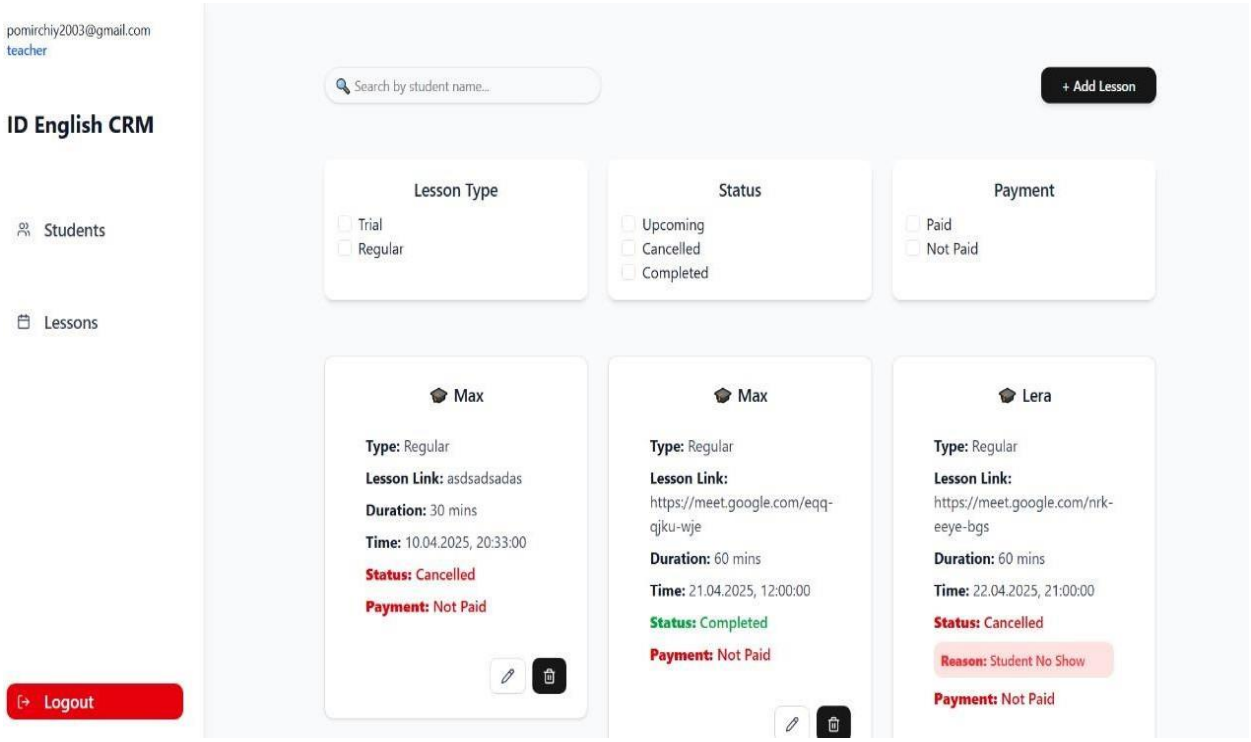

Рисунок 1.2 – Інтерфейс Teachers Hub: головний екран викладача з управлінням уроками

1.3 Порівняльний аналіз існуючих рішень для керування студентами та уроками

У сучасному освітньому середовищі існує чимало онлайн-інструментів, які допомагають викладачам і школам організовувати навчальний процес. Серед найпопулярніших платформ – Off2Class, Planboard та MySchool. Кожен з цих сервісів має свої переваги та обмеження, що робить їх більш або менш придатними до використання в контексті невеликих приватних мовних шкіл чи індивідуальних викладачів. Off2Class є платформою, орієнтованою на викладання англійської мови. Основна увага приділяється структуруванню уроків, наданню готових матеріалів та відстеженню прогресу учнів. Втім, його функціонал для керування оплатами або зручної взаємодії з базою студентів доволі обмежений. Інтерфейс зрозумілий, але досить шаблонний. Головний екран після входу показано на рисунку 1.3.

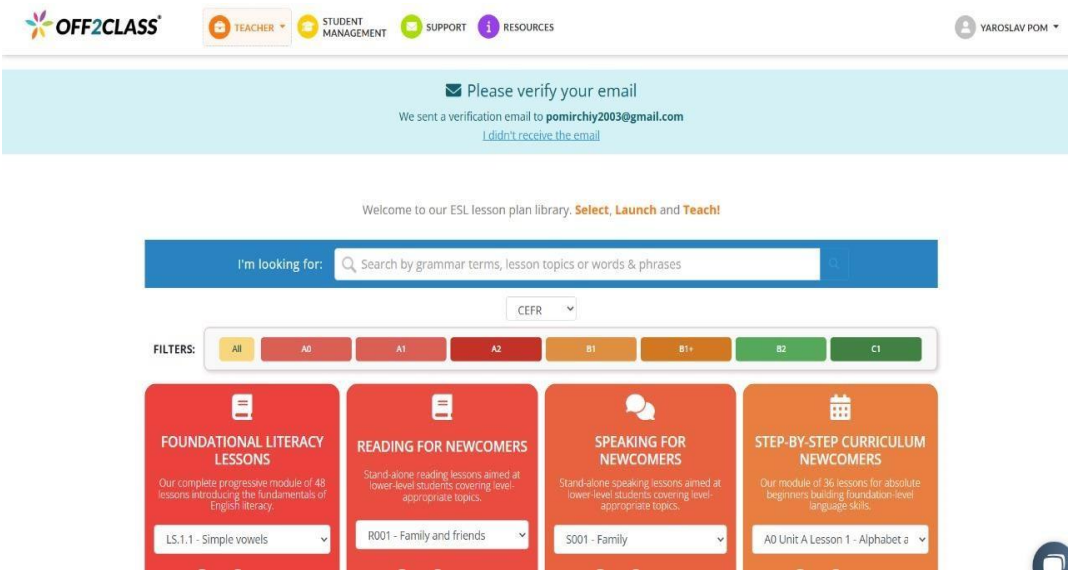

Рисунок 1.3 – Скріншот вигляду UI застосунку Off2Class після реєстрації користувача

Planboard – більш універсальна платформа для створення навчальних планів, яка дозволяє викладачам створювати уроки, планувати тиждень та зручно організовувати розклад. При цьому система майже не має можливостей для ведення бази студентів або обліку оплат. Вона більше підходить для шкільних вчителів, ніж для мовних шкіл. Вигляд інтерфейсу Planboard представлено на рисунку 1.4.

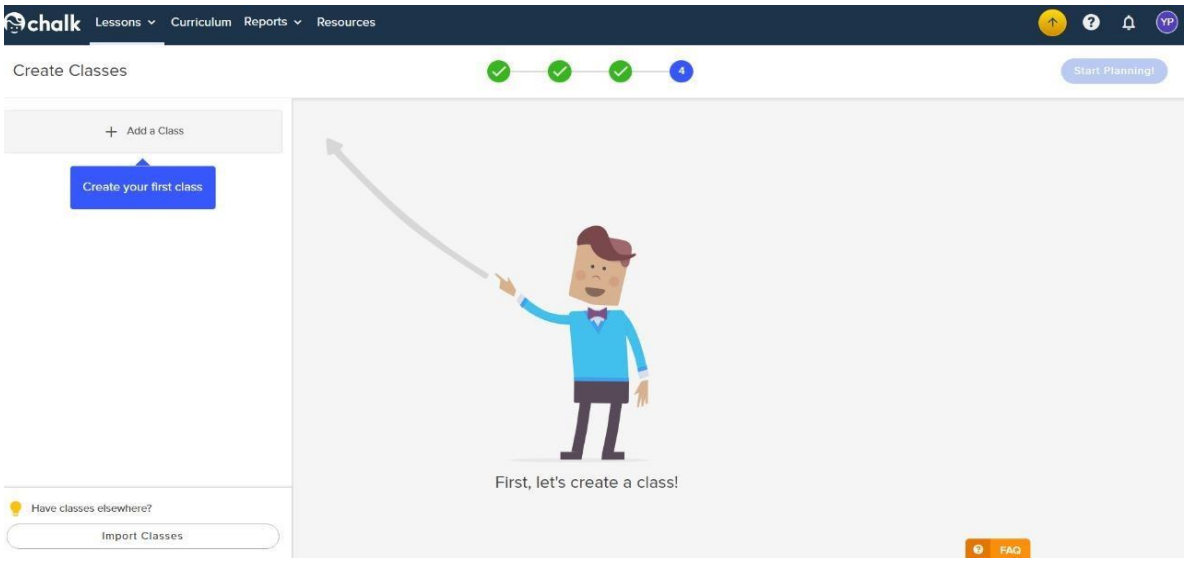

Рисунок 1.4 – Скріншот вигляду UI застосунку Planboard після реєстрації користувача

Таблиця 1.1 – Порівняння обраних застосунків по критеріям

Критерій	Planboard	Off2Class	Teachers Hub
Облік студентів	Частково (замітки)	Так	Так
Створення та облік уроків	Так	Так	Так
Облік оплат	Ні	Частково (через сторонні сервіси)	Так
Ролі користувачів (учитель/адмін)	Ні	Частково	Так
Можливість відмічати статус уроку (проведено/скасовано)	Ні	Частково	Так
Простота інтерфейсу	Висока	Середня	Висока
Кастомізація під потреби	Низька	Низька	Висока
Орієнтація на приватні школи	Ні	Частково	Так

1.4 Постановка задачі

Таким чином, створення власного CRM-застосунку для керування студентами та уроками є актуальним рішенням для невеликих мовних шкіл та приватних викладачів. Тому ставиться завдання розробки інтуїтивно зрозумілого, зручного та функціонального застосунку, що забезпечить централізоване керування навчальним процесом.

Об’єктом роботи є система для керування студентами, уроками,

оплатами та адміністративними діями у малих освітніх закладах.

Метою даної роботи є створення сучасного, функціонального та безпечного вебзастосунку для ефективного керування навчальним процесом у малих приватних мовних школах. Такий застосунок має забезпечувати зручний облік студентів, планування уроків, контроль оплат та розмежування прав доступу між викладачами й адміністраторами. Ключовим завданням було реалізувати систему, яка поєднує простоту у користуванні, швидкодію, адаптивний інтерфейс та можливість масштабування функціоналу без втрати продуктивності.

2 МОДЕЛІ ДЛЯ ПЕРЕДАЧІ ТА ОБРОБКИ ДАНИХ У ВЕБСАЙТІ «TEACHERS HUB»

2.1 Аналіз використаних інструментів для реалізації застосунку

Для реалізації Teachers Hub – сучасної платформи керування учнями, уроками та оплатами для мовних шкіл було використано сучасні технології веброзробки та гнучку архітектуру з чітким розділенням відповідальностей між інтерфейсом користувача, логікою бізнес-процесів та системою збереження даних.

Інтерфейс користувача реалізовано з використанням фреймворку Next.js, побудованого на основі бібліотеки React [11]. Next.js забезпечує поєднання зручного компонентного підходу React з потужною серверною логікою та підтримкою функціональностей, які критично важливі для сучасних вебзастосунків. Завдяки цьому фреймворку стало можливим реалізувати не лише класичну клієнтську логіку, але й серверний рендеринг сторінок, що покращує продуктивність і SEO-оптимізацію. Крім того, Next.js підтримує маршрутизацію на основі файлів, динамічне завантаження контенту, API-обробники та автоматичну оптимізацію сторінок, що значно прискорює завантаження навіть на повільних інтернет-з'єднаннях. Також фреймворк чудово інтегрується з платформою Vercel, яка є розробником самого Next.js, що дозволяє легко розгортати застосунок і масштабувати його в залежності від навантаження. Завдяки гнучкості архітектури Next.js, стало можливим швидко реалізувати як базовий функціонал, так і складні бізнес-процеси, включаючи інтеграцію з базою даних, автентифікацією та захистом даних. Розробка велась із використанням мови програмування TypeScript, яка забезпечує сувору типізацію, підвищує надійність коду та спрощує його підтримку в довгостроковій перспективі.

Для UI-компонентів використано бібліотеку ShadCN UI, побудовану на

основі Radix UI [12] та стилізовану за допомогою Tailwind CSS [13]. Це рішення дозволяє створювати сучасні, адаптивні та доступні інтерфейси з урахуванням вимог доступності (accessibility). ShadCN UI підтримує типізацію на основі TypeScript, що покращує інтеграцію з іншими частинами застосунку та зменшує кількість помилок під час розробки. Компоненти бібліотеки легко кастомізуються, що дозволяє змінювати їхній вигляд відповідно до візуального стилю проєкту без необхідності створення всього з нуля. Завдяки використанню Tailwind CSS, стилізація компонентів виконується безпосередньо у класах, що пришвидшує процес верстки та робить інтерфейс єдиним і узгодженим. Radix UI, що лежить в основі ShadCN UI, забезпечує доступність компонентів згідно з стандартами WCAG, включаючи правильну навігацію з клавіатури, ARIA-атрибути та інші критично важливі аспекти UX. У результаті застосунок Teachers Hub отримав професійний інтерфейс, який швидко працює, добре масштабовується та відповідає сучасним стандартам веброзробки.

Інструмент Clerk використано для організації автентифікації користувачів та управління їхніми ролями – викладач або адміністратор [14]. Clerk є сучасним рішенням для авторизації, яке дозволяє швидко інтегрувати безпечну систему входу в застосунок без необхідності самостійно реалізовувати логіку реєстрації, скидання пароля, підтвердження електронної пошти або керування сесіями. Clerk надає вбудований UI для авторизації, що легко адаптується до дизайну застосунку. У межах Teachers Hub використано механізм авторизації через Clerk з підтримкою поділу доступу на основі ролей – це означає, що після входу в систему кожен користувач бачить лише той функціонал, який дозволений його роллю. Наприклад, викладач має доступ до управління студентами та уроками, тоді як адміністратор може переглядати всі уроки, змінювати статуси оплат та редагувати профілі інших користувачів. Clerk також автоматично зв’язується з Supabase, дозволяючи створювати безпечні правила доступу до бази даних на рівні кожного окремого користувача. Такий підхід забезпечує гнучку та надійну систему доступу, яка

не потребує додаткових серверних налаштувань.

Збереження даних реалізовано на основі Supabase – хмарної платформи з відкритим кодом, яка надає розширений інтерфейс до бази даних PostgreSQL [15] та дозволяє реалізувати повноцінний backend без необхідності створення окремого сервера. Supabase підтримує зберігання структурованих даних, авторизацію, завантаження медіафайлів і створення REST або RPC API – усе це з мінімальними зусиллями з боку розробника. У застосунку Teachers Hub Supabase використовується не лише як база даних, а й як центр обробки всіх запитів – саме через його SDK здійснюється взаємодія між інтерфейсом користувача та логікою збереження інформації. Серед головних переваг Supabase – підтримка Row-Level Security (RLS), яка дозволяє реалізувати тонке налаштування доступу до даних на рівні окремих рядків у таблицях. Крім того, Supabase має вбудований механізм зберігання файлів, який можна використовувати для домашніх завдань або презентацій, а також підтримує реальний час і сповіщення при оновленні даних. Завдяки цьому Supabase став оптимальним вибором для створення швидкого, безпечного та масштабованого CRM-застосунку без необхідності окремої серверної інфраструктури.

Архітектура застосунку побудована за принципом трьохрівневої моделі – з чітким розмежуванням обов’язків між інтерфейсом користувача (frontend), логікою обробки запитів (умовний API) та рівнем зберігання даних (база даних Supabase). Такий підхід дозволяє розділити відповідальність кожного з компонентів системи, що значно спрощує процес розробки, налагодження та майбутньої підтримки. Кожен рівень виконує власну роль: користувацький інтерфейс відповідає за взаємодію з користувачем, логіка – за обробку запитів, а база даних – за збереження та безпеку інформації. Це забезпечує масштабованість застосунку, підвищує надійність, дозволяє вносити зміни до одного з рівнів без порушення роботи інших частин системи, а також робить реалізацію гнучкою та готовою до подальшого розширення функціоналу.

Як видно з рисунку 2.1, бібліотека React стабільно займає лідерські

позиції серед інструментів для створення інтерфейсів користувача. Саме тому React був обраний як основа для розробки frontend частини Teachers Hub. Це забезпечує не лише високу швидкодію, а й зручну модульну структуру інтерфейсу.

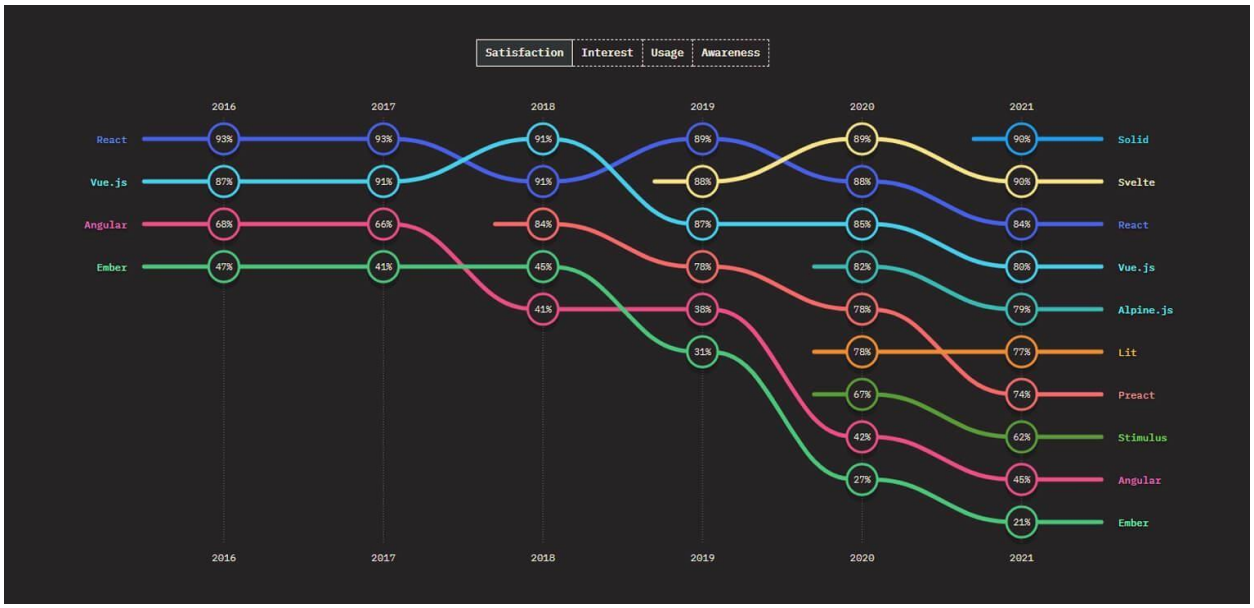

Рисунок 2.1 – Рейтинг бібліотек та фреймворків для розробки інтерфейсу користувача станом на 2025 рік

Так, набір обраних технологій забезпечив швидку розробку MVP Teachers Hub, при цьому залишивши достатній простір для розширення та додавання нових модулів у майбутньому.

Усі інструменти, які були використані у створенні вебзастосунку Teachers Hub, логічно поєднуються в межах трьохрівневої архітектури. Цей підхід дозволяє ефективно розділити обов’язки між різними частинами системи, забезпечуючи стабільність, масштабованість та зручність розробки.

Трьохрівнева архітектура є одним з найпоширеніших підходів при побудові сучасних вебзастосунків. Вона передбачає розділення системи на три логічні рівні: Presentation layer (інтерфейс користувача), Application layer (логіка застосунку) та Data layer (рівень даних).

Перший рівень – інтерфейс користувача – реалізовано з використанням

React, Next.js та Tailwind CSS. Саме цей рівень забезпечує взаємодію викладача або адміністратора зі всіма функціями системи: переглядом студентів, редагуванням уроків, перевіркою оплат тощо. Завдяки використанню сучасних технологій, інтерфейс залишається інтуїтивно зрозумілим, адаптивним та швидким у роботі. Його основне завдання – збір даних від користувача та виведення відповідної інформації після обробки.

Другий рівень – логічний – відповідає за обробку усіх бізнес-процесів: додавання нових уроків, призначення домашніх завдань, зміну статусу платежу, керування ролями користувачів тощо. У випадку Teachers Hub ця логіка частково реалізована через функціональність Supabase (RPC-запити та serverless-функції), а також через інтерфейс взаємодії з Clerk. Вся бізнес-логіка чітко розділена від інтерфейсу, що дозволяє легко вносити зміни, не зачіпаючи інші частини проєкту.

Третій рівень – рівень даних – забезпечує збереження та доступ до інформації про користувачів, студентів, уроки, домашні завдання, статуси оплат та інші об’єкти системи. У Teachers Hub для цього використовується база PostgreSQL, інтегрована через Supabase. Таке рішення дозволяє не лише зберігати дані у структурованому вигляді, але й забезпечити гнучкі права доступу, безпечну аутентифікацію, швидкий пошук і фільтрацію.

Завдяки чіткому поділу на три рівні, архітектура Teachers Hub забезпечує зручність підтримки, високу безпеку та можливість легкого розширення функціоналу в майбутньому. Схематичне зображення побудови цієї архітектури наведено на рисунку 2.2.

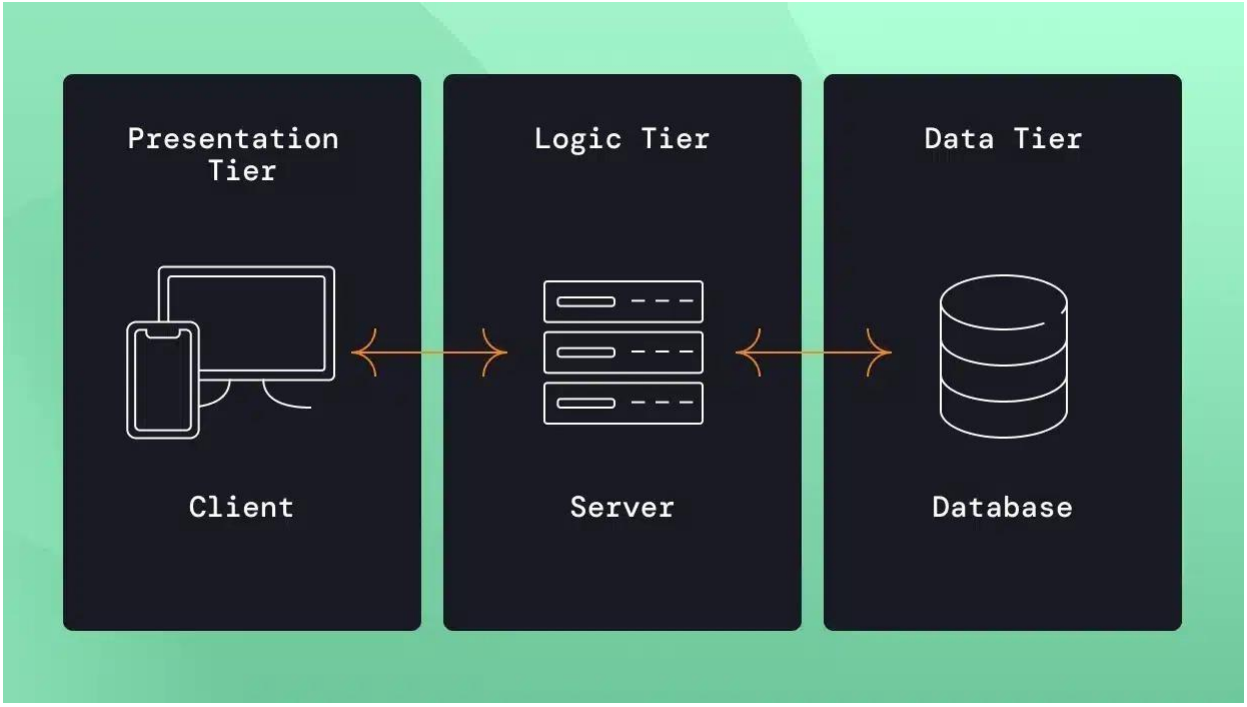

Рисунок 2.2 – Схема трьохрівневої архітектуру сучасного застосунку

2.2 Моделі для інтеграції та передачі даних у застосунку Teachers Hub

У сучасних вебзастосунках особливе значення має організація ефективної та безпечної передачі даних між інтерфейсом користувача, бізнес-логікою та базою даних. Teachers Hub — це кастомний CRM- застосунок для мовних шкіл, який інтегрує кілька ключових модулів: облік студентів, планування занять, домашні завдання та календар.

Для реалізації взаємодії між клієнтською частиною (frontend) та серверною логікою (backend) у Teachers Hub використовується REST API — один із найпоширеніших підходів для обміну даними у вебзастосунках. [16]

Технологія REST API чудово підходить для застосунків із чітко структурованими ресурсами, такими як студенти, уроки, оплати, завдання тощо [17]. Крім того, JSON як формат передачі даних є зручним для роботи як з боку frontend (React), так і backend (Node.js/Express). [18]

Таблиця 2.1 – Порівняння технологій для реалізації API

Критерії порівняння	**REST**	**SOAP**	**GraphQL**
Архітектура	Ресурсно-орієнтована	Гнучка, запит лише необхідних полів	RPC-орієнтована
Формат передачі даних	JSON, XML, HTML	XML	JSON
Тип передачі даних	Синхронна	Синхронна та асинхронна	Синхронна та асинхронна
Кешування	Підтримується кешування запитів	Підтримується кешування запитів	Не підтримується
Масштабування	Легке та ефективне масштабування	Складне масштабування	Легке масштабування
Версіювання	Підтримується	Підтримується	Підтримується
Надійність	Надійніший та стійкий до збоїв	Надійний, але менш стійкий до збоїв	Надійний, але менш стійкий до збоїв

2.2.1 Моделі використані для API Teachers Hub

У застосунку Teachers Hub не використовується класичний підхід з окремим сервером API, натомість реалізовано безпосередню інтеграцію з базою даних через Supabase SDK**.** Це стало можливим завдяки використанню Supabase як backend-as-a-service, що поєднує в собі базу даних PostgreSQL, автентифікацію, авторизацію, реальний час та простий клієнтський SDK.

Застосунок побудовано з використанням Next.js (App Router) на

клієнтській та серверній сторонах. Замість традиційного REST API, взаємодія з даними відбувається через Supabase JavaScript SDK, який дозволяє виконувати запити до бази напряму з клієнта або серверних обробників. [19]

```
const { data, error } = await supabase
  .from('students')
  .select('*')
  .eq('teacher_id', user.id);
```

Рисунок 2.3 – Приклад отримання списку студентів для поточного вчителя

2.2.2 Моделі використані у застосунку Teachers Hub

Teachers Hub – це вебзастосунок для керування навчальним процесом у мовних школах, який дозволяє викладачам і адміністраторам ефективно взаємодіяти з базою студентів, розкладом занять, оплатами тощо. Для реалізації цього функціоналу не використовувались сторонні API, натомість застосунок працює безпосередньо з власною базою даних за допомогою Supabase.

Supabase – це backend-as-a-service, що надає потужну SQL-базу PostgreSQL, інтерфейс для взаємодії через REST та SDK-клієнти. У випадку Teachers Hub дані не передаються через окремий backend – замість цього, застосунок побудовано за принципами «serverless» із прямою інтеграцією між frontend та базою через Supabase SDK.

Одним із ключових інструментів, що забезпечують зберігання та захист даних у Teachers Hub, є Supabase. У зв’язці з Clerk ця платформа дозволяє реалізувати надійну реєстрацію та автентифікацію користувачів – з чітким розмежуванням прав доступу між адміністраторами та викладачами. Supabase

також відповідає за збереження основних сутностей системи – таких як студенти, уроки, статуси оплат та типи занять [20]. Важливою перевагою цього рішення є підтримка Row-Level Security (RLS) – механізму, що забезпечує контроль доступу до даних безпосередньо на рівні бази, надаючи доступ лише до інформації, дозволеної для перегляду відповідній ролі користувача.

Для кожного з основних об’єктів у системі (студенти, уроки, оплати) створено окремі таблиці з чітко визначеною структурою. Наприклад, таблиця students містить такі поля: id, teacher_id, full_name, level, description.

Дані зберігаються централізовано в базі даних Supabase та обробляються через клієнтські запити з використанням @supabase/supabase-js бібліотеки. Це дозволяє не лише скоротити час розробки, а й уникнути створення окремого серверного шару.

Такий підхід має ряд важливих переваг. По-перше, він забезпечує швидку взаємодію між клієнтом і базою даних, що позитивно впливає на загальну продуктивність системи. По-друге, розгортання застосунку є доволі простим і легко масштабованим у разі зростання кількості користувачів або обсягу даних. Крім того, відсутність окремого backend-сервера зменшує загальні витрати на підтримку проєкту. Ще однією перевагою є гнучка модель доступу до даних, яку можна реалізувати безпосередньо на рівні бази даних завдяки політикам безпеки (RLS).

Однак, разом з перевагами, цей підхід має і певні недоліки. Вся логіка застосунку виконується на стороні клієнта, що створює додаткові вимоги до безпеки — необхідно ретельно контролювати, які саме запити можуть надсилатися та які дані повертаються. Важливою умовою є правильне налаштування політик RLS, оскільки лише вони забезпечують розмежування доступу до даних між різними користувачами. У випадку потреби в складній серверній логіці або бізнес-правилах, відсутність окремого backend може стати обмеженням, оскільки не всі обчислення зручно реалізовувати на стороні клієнта чи у базі даних.

2.2.3 Моделі для організації поширення даних всередині застосунку Teachers Hub

Одним із ключових аспектів розробки вебзастосунку Teachers Hub було забезпечення ефективного, структурованого й безпечного обміну даними між різними частинами системи – інтерфейсом користувача, логікою обробки та базою даних Supabase. Оскільки застосунок не передбачає використання окремого серверного бекенду, уся логіка взаємодії з даними реалізована безпосередньо у фронтенд-частині, побудованій на базі Next.js, із використанням офіційного SDK Supabase. Це означає, що особливу увагу було приділено моделюванню структури даних і контролю за тим, як саме інформація передається, зберігається й оновлюється.

Моделі відіграють ключову роль у забезпеченні цілісності системи. Вони дозволяють уніфікувати формат інформації, що передається між компонентами, і забезпечити стабільну типізацію, яка особливо важлива при масштабуванні застосунку. Весь проєкт реалізований мовою TypeScript, що дозволяє чітко визначати, які саме дані повинні надходити у форму створення уроку, які – у форму редагування студента, а які – виводитись в інтерфейсі адміністратора. Наприклад, модель уроку містить такі обов’язкові поля, як ідентифікатор викладача, ідентифікатор студента, дата заняття, тип уроку та його статус. При цьому система не дозволить зберегти неповні або некоректні дані – помилка буде виявлена ще на етапі розробки.

Передача інформації у застосунку завжди відбувається через чітко визначені моделі. Для кожного процесу – створення студента, фіксація уроку, оновлення статусу оплати або завантаження домашнього завдання – використовується власна структура даних. Це дозволяє не лише контролювати, які саме поля потрапляють у запит до Supabase, а й обмежувати користувача від доступу до інформації, яка йому не призначена. Наприклад, при перегляді списку студентів викладач бачить лише загальну інформацію: ім’я, рівень і короткий опис. Внутрішні службові поля або інші конфіденційні

дані не передаються на клієнт.

Моделі дозволяють легко керувати відображенням і фільтрацією даних в інтерфейсі. Вони чітко розділяють інформацію, яка потрібна для логіки (ідентифікатори, статуси, мітки), від тієї, яка необхідна для користувача (імена, дати, коментарі). Це особливо зручно при роботі з великими обсягами інформації – наприклад, під час перегляду журналу занять або історії оплат. Система відображає лише мінімально необхідні поля, що знижує навантаження на браузер і пришвидшує інтерфейс.

Окрему увагу було приділено безпеці. Усі запити до бази даних проходять перевірку прав доступу через політику Supabase RLS (Row-Level Security), а моделі додатково фільтрують поля, які взагалі не повинні передаватись на клієнт. Це забезпечує додатковий рівень захисту, навіть у випадках, коли користувач намагається втрутитися в логіку застосунку.

Ще однією перевагою чіткої структури моделей є їхня масштабованість. У випадку додавання нового поля в базу даних або оновлення логіки, розробник може змінити модель у TypeScript-файлі, і всі частини застосунку, що використовують цю модель, автоматично отримають відповідне оновлення через систему типів. Це знижує ймовірність виникнення логічних помилок і значно пришвидшує розробку нових функцій.

У результаті застосування моделей у вебзастосунку Teachers Hub дозволяє досягти високої надійності, зрозумілої структури коду та узгодженої роботи між усіма частинами системи. Такий підхід не лише спрощує підтримку, а й дозволяє безпечно масштабувати проєкт у майбутньому, додаючи нові типи уроків, формати звітності або функції без ризику для стабільності основного функціоналу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ TEACHERS HUB

3.1 Обґрунтування вибору середовища

У процесі розробки вебзастосунку Teachers Hub було прийнято рішення реалізовувати програмний продукт із використанням сучасного технологічного стеку, який забезпечує швидку розробку, гнучке масштабування, високу продуктивність і надійний захист даних. Такий вибір обумовлений потребою створити вебзастосунок, який був би доступний через звичайний браузер, не вимагав складного розгортання, міг обробляти численні запити в реальному часі та забезпечував зручну взаємодію для викладачів і адміністраторів мовної школи.

Основу клієнтської частини становить фреймворк Next.js, який побудовано на базі бібліотеки React. Цей інструмент дозволяє створювати односторінкові застосунки (SPA) із підтримкою серверного рендерингу, що позитивно впливає на швидкість завантаження сторінок, продуктивність і SEO-оптимізацію. Особливістю Next.js є можливість обробки запитів як на клієнтському, так і на серверному боці, що забезпечує гнучкість у реалізації складних логік. Крім того, застосунок реалізовано з використанням App Router – нової архітектури маршрутизації, що дозволяє чітко розділяти макети, сторінки та динамічні маршрути.

Для написання логіки використовується мова TypeScript, яка є надбудовою над JavaScript і забезпечує строгішу типізацію. Це значно підвищує безпеку коду, зменшує ймовірність помилок під час компіляції та дозволяє зручно працювати з великими об'єктами, типами моделей та взаємозалежними даними. У випадку такого застосунку, як Teachers Hub, де є кілька пов’язаних сутностей (студенти, уроки, оплати, завдання), використання TypeScript стало критично важливим рішенням для підтримки цілісності структури даних.

Користувацький інтерфейс було реалізовано за допомогою бібліотеки

ShadCN UI, яка використовує компоненти Radix UI у поєднанні з утилітарною системою стилізації Tailwind CSS. Такий підхід дозволив створити чистий, сучасний та адаптивний інтерфейс без потреби в окремих CSS-файлах. Усі стилі застосовуються безпосередньо в класах компонентів, що пришвидшує розробку, полегшує підтримку та забезпечує єдину візуальну логіку на всіх сторінках.

Radix UI в основі ShadCN надає доступні за стандартами WCAG компоненти, які підтримують ARIA-атрибути, клавіатурну навігацію, коректну поведінку фокусів і модальних вікон [27]. Така доступність є важливою умовою для освітнього середовища, оскільки дозволяє взаємодіяти з платформою користувачам із різними потребами. На рисунку 3.1 зображено вигляд головної сторінки застосунку Teachers Hub, яка демонструє приклад використання компонентів інтерфейсу.

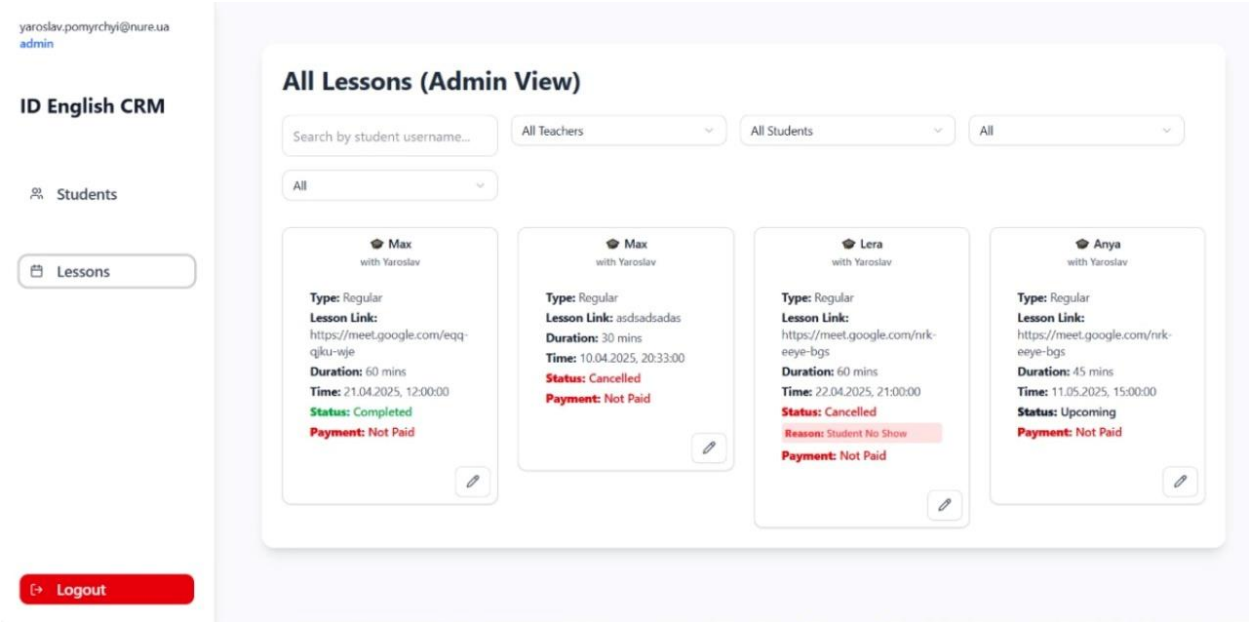

Рисунок 3.1 – Головна сторінка інтерфейсу Teachers Hub

Ще одним важливим елементом реалізації є система автентифікації. У межах проєкту було використано Clerk – готове рішення для обробки реєстрації, входу в систему, сесій користувачів та управління ролями. Clerk

дозволяє швидко додати реєстраційні форми, підтвердження електронної пошти, відновлення паролю, вхід за допомогою соцмереж, а головне – реалізувати розмежування прав доступу. У застосунку Teachers Hub передбачено два основні типи ролей: викладач і адміністратор. Після входу кожен користувач отримує доступ лише до того функціоналу, який відповідає його ролі.

Clerk має готові компоненти для Next.js, що дозволяє реалізувати як клієнтське, так і серверне керування сесіями. У результаті, замість написання власного бекенду для автентифікації, було обрано стабільне рішення, що забезпечує високу безпеку та швидку інтеграцію.

На відміну від класичних підходів, у яких розробник створює окремий сервер на Node.js або Express, у Teachers Hub роль бекенду виконує Supabase – хмарна платформа з відкритим кодом, яка надає доступ до бази даних PostgreSQL, підтримує механізми авторизації, керування медіафайлами, написання серверної логіки (через Edge Functions), а також має SDK для взаємодії з усіма компонентами з клієнтської сторони.

Використання Supabase дозволило уникнути необхідності створення окремих серверних роутів: усі операції над базою (читання, запис, оновлення, видалення) реалізуються напряму через бібліотеку `@supabase/supabase-js`. На рисунку 3.2 представлено загальну структуру бази даних, яка містить таблиці для користувачів, студентів, уроків, оплат, домашніх завдань, словникового запасу тощо.

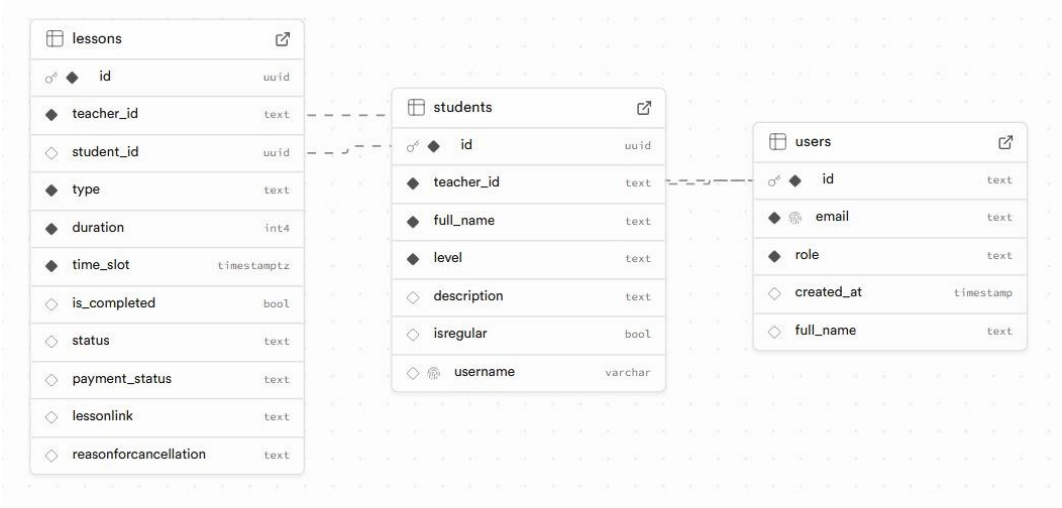

Рисунок 3.2 – Структура бази даних у Supabase

Інтерфейс працює за реактивною логікою – усі зміни, які виконує користувач (наприклад, створення нового студента, оновлення статусу уроку чи редагування оплати), миттєво відображаються на екрані без потреби перезавантаження сторінки. Це забезпечує приємний користувацький досвід і дає змогу працювати із застосунком навіть на мобільних пристроях. На рисунку 3.3 показано приклад форми редагування уроку, яка використовується викладачем для внесення змін до заняття або створення нового запису.

Всі дані, які зберігаються в Supabase, додатково захищені за допомогою Row-Level Security (RLS) – системи політик, яка дозволяє налаштувати доступ до конкретних записів у таблицях. Наприклад, викладач бачить лише власних студентів і уроки, тоді як адміністратор має глобальний доступ до всієї бази. Це гарантує безпечне середовище, у якому неможливо отримати чужі дані навіть при спробі змінити запит вручну.

У підсумку, обраний стек технологій – Next.js + TypeScript + ShadCN UI + Clerk + Supabase – дозволив реалізувати повноцінний вебзастосунок без необхідності окремого сервера, забезпечити надійну роботу з даними, зручний інтерфейс і чітке розмежування доступу. Така архітектура є чудовим прикладом сучасного підходу до створення MVP-продукту з високим потенціалом для подальшого масштабування та розвитку в напрямку SaaS або

корпоративного рішення.

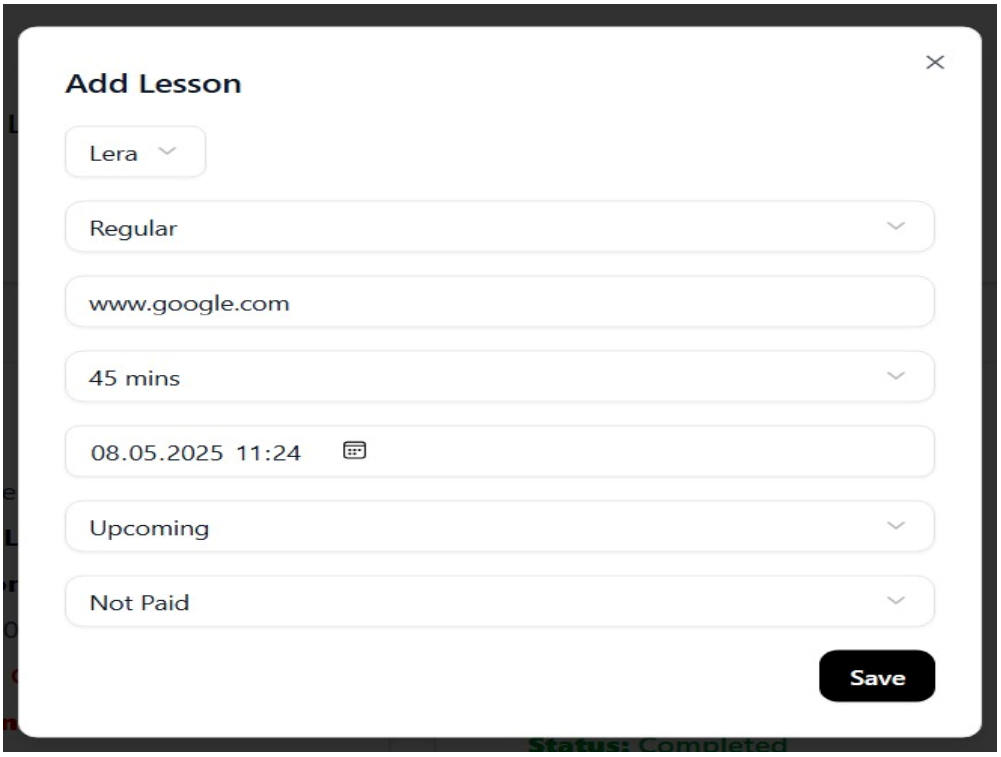

Рисунок 3.3 – Форма створення або редагування уроку

3.2 Програмна реалізація застосунку Teachers Hub

Уся бізнес-логіка, компоненти та структури даних були реалізовані з використанням мови TypeScript. Це дало змогу забезпечити сувору типізацію коду, уникнути поширених помилок на етапі компіляції, а також спростити процес рефакторингу і масштабування у разі зростання кількості функцій. TypeScript став надійним інструментом для розробки такого застосунку, де важливо чітко розмежовувати ролі користувачів, керувати зв’язками між сутностями та обробляти запити до бази даних.

Як середовище розробки було обрано WebStorm, що забезпечило стабільну роботу з великим кодовим базисом [21]. Завдяки інтелектуальному автодоповненню, зручній навігації, інтеграції з системою контролю версій Git та підтримці сучасних фреймворків, WebStorm дав

можливість прискорити цикл розробки і зменшити кількість потенційних помилок під час написання коду.

Візуальна частина інтерфейсу реалізована з використанням бібліотеки ShadCN UI, яка базується на Radix UI та стилізується за допомогою Tailwind CSS. Такий підхід забезпечив створення адаптивного, доступного та естетично привабливого інтерфейсу, який легко налаштовується відповідно до потреб школи. Компоненти системи, такі як таблиці, форми, діалоги, сповіщення, побудовані на основі Radix UI, відповідають стандартам WCAG та дозволяють підтримувати доступність для користувачів з особливими потребами. Tailwind CSS, своєю чергою, забезпечив зручний утилітарний підхід до стилізації, що суттєво зменшило кількість зовнішніх CSS-файлів і прискорило верстку інтерфейсу [28].

Користувацька панель викладача реалізована у вигляді інтерактивного кабінету, в межах якого можна працювати з усіма ключовими функціями без потреби перемикання між окремими сторінками. В одному екрані викладач має змогу переглядати список студентів, редагувати їхні дані, додавати нові профілі, керувати розкладом, створювати уроки, змінювати їхній статус та фіксувати оплату. Окрему увагу приділено можливості фільтрації за рівнем англійської, регулярністю занять, датою або типом уроку. Вигляд панелі викладача зображено на рисунку 3.4.

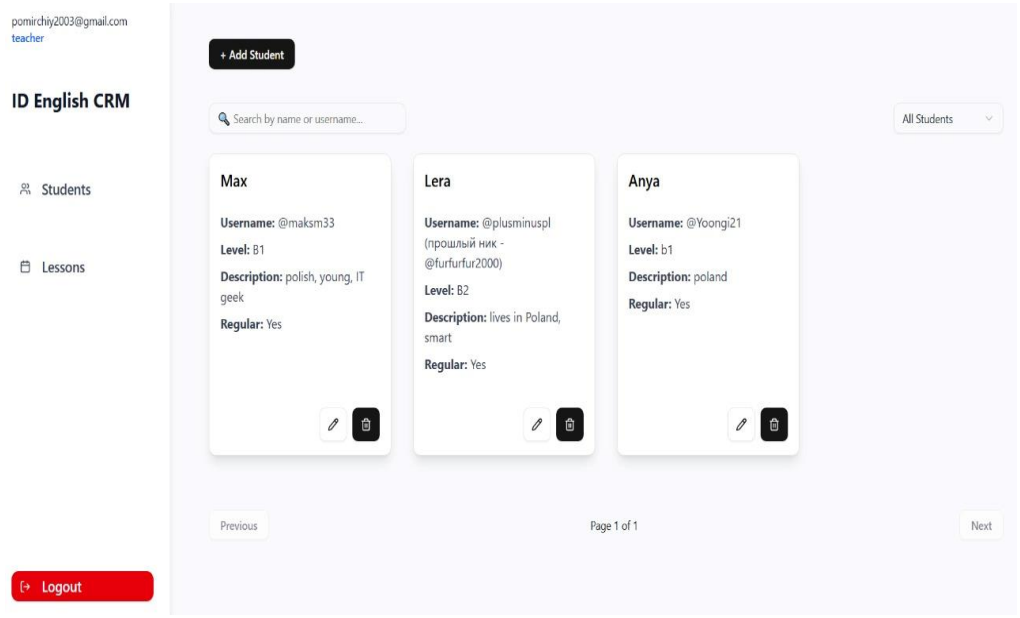

Рисунок 3.4 – Головна панель викладача із вкладкою «Мої студенти»

Механізм авторизації реалізований на основі сервісу Clerk [23]. Завдяки інтеграції цього інструменту користувачі системи можуть реєструватися, входити у власні облікові записи, а також отримувати доступ лише до дозволених функцій, відповідно до своєї ролі [29]. Рольова модель включає щонайменше два типи користувачів – викладач і адміністратор. Після входу в систему Clerk автоматично визначає роль користувача і обмежує або надає доступ до відповідних функцій. Наприклад, адміністратор може переглядати дані всіх викладачів і студентів, редагувати статуси уроків та контролювати оплату, тоді як викладач має доступ лише до своїх студентів і занять.

Збереження інформації реалізовано на основі Supabase – хмарної платформи [24], що надає інтерфейс до бази даних PostgreSQL, а також підтримує систему авторизації, зберігання медіа та виконання серверної логіки. У межах Teachers Hub для кожної основної сутності створено окрему таблицю: студенти, уроки, оплати, типи занять, домашні завдання. За безпеку даних відповідає система політик Row-Level Security, яка дає змогу обмежити доступ до кожного рядка таблиці залежно від ролі або ID користувача. Таким чином, жоден викладач не може отримати доступ до студентів іншого викладача навіть через прямі SQL-запити. [25]

Усі запити до бази даних реалізуються через бібліотеку Supabase SDK, що дозволяє інтегрувати роботу з даними без створення окремого серверного API. Під час створення нового уроку, наприклад, система формує запит до відповідної таблиці, додаючи до нього ідентифікатор викладача, дату, студента, тип заняття та початковий статус. Ці дії виконуються безпосередньо з клієнтської частини, а відповіді повертаються асинхронно, що забезпечує швидкість взаємодії.

Інтерфейс уроків реалізовано у вигляді таблиці з підтримкою сортування, фільтрації та зміни статусу. Система дозволяє швидко позначати уроки як проведені, скасовані або перенесені. Завдяки адаптивному дизайну інтерфейс однаково добре працює як на великих екранах, так і на мобільних пристроях, що дозволяє викладачам користуватись системою в будь-якому місці.

Для фінансового контролю створено окрему вкладку, що дозволяє відстежувати статус оплати кожного уроку. Уроки можуть мати статус оплати «сплачено», «не сплачено» або «очікує підтвердження». Адміністратор школи може переглядати аналітику по викладачах та формувати звіти, а в майбутньому передбачається інтеграція функції експорту звітності у форматах Excel або PDF.

Домашні завдання реалізовано як окрему модель у базі. Кожне завдання містить прив’язку до студента, текстове поле, опціональне прикріплення файлів, дедлайн і статус виконання. Студент може переглядати свої завдання, завантажувати файли та позначати їх як виконані через власний кабінет.

Весь код було організовано за модульним підходом. Для кожної частини застосунку створено окремі компоненти з власною логікою та стилями. Усі сторінки побудовані на основі маршрутизатора App Router у Next.js, що дозволило реалізувати як статичні, так і динамічні маршрути, включаючи сторінки з параметрами URL.

Програмна реалізація Teachers Hub поєднує сучасні технології, зручний користувацький досвід та безпечну роботу з даними. Застосунок дозволяє

автоматизувати ключові аспекти роботи викладача та адміністратора мовної школи без необхідності використовувати сторонні сервіси.

3.3 Публікація та розгортання застосунку Teachers Hub

Завершення основного етапу розробки вебзастосунку Teachers Hub супроводжувалося необхідністю розмістити проєкт у такому середовищі, яке забезпечить швидкий, безпечний і безперебійний доступ до системи з будь-якого пристрою через браузер. Для цієї мети було обрано платформу Vercel — сучасний хмарний сервіс, оптимізований для розгортання проєктів, створених на основі фреймворку Next.js. Саме ця платформа надає інтегровану підтримку серверного рендерингу, безперервної інтеграції, масштабування та автоматичної оптимізації сторінок.

Вихідний код застосунку Teachers Hub зберігається у приватному репозиторії на платформі GitHub [22], що дозволяє вести контроль версій, працювати з гілками та забезпечує зручну співпрацю в команді. Репозиторій було інтегровано з обліковим записом Vercel, завдяки чому стало можливим автоматичне розгортання змін із основної гілки main. Кожен новий коміт запускає процес збірки проєкту, виконання перевірок та оновлення продуктивного середовища без ручного втручання з боку розробника. Такий підхід забезпечує гнучкість і стабільність, а також пришвидшує внесення змін у застосунок, що особливо важливо в умовах постійного розвитку функціоналу.

Платформа Vercel не потребує окремого налаштування серверів або виділеного хостингу [26]. Вона автоматично обробляє маршрути, оптимізує сторінки, виконує SSR або ISR (інкрементальну генерацію) залежно від конфігурації, а також кешує статичні елементи для покращення продуктивності. Завдяки цьому застосунок Teachers Hub працює стабільно навіть при збільшенні кількості користувачів.

На етапі розгортання було налаштовано набір змінних середовища,

необхідних для взаємодії з зовнішніми сервісами. Зокрема, було додано ключі для підключення до бази даних Supabase (адреса проєкту та публічний API-ключ), а також параметри конфігурації Clerk – frontend API URL, publishable key та інші параметри, що забезпечують авторизацію користувачів. Також було обрано домен, за яким застосунок доступний кінцевим користувачам, і здійснено тестування всіх інтегрованих сервісів у бойовому середовищі.

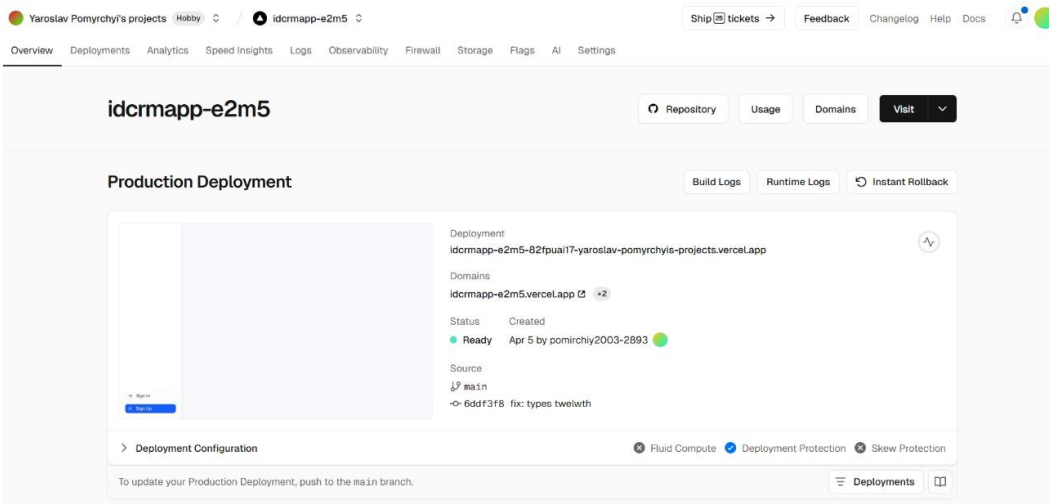

Рисунок 3.5 – Панель керування Vercel з активним проєктом Teachers Hub

У якості продакшн-бази використовувалася робоча база Supabase, яка вже містила таблиці користувачів, студентів, уроків, оплат і домашніх завдань. Усі дії, які виконуються викладачем або адміністратором (створення уроків, оновлення інформації, додавання студентів або завдань), моментально зберігаються до цієї бази даних. Це дозволяє працювати з реальними даними в режимі реального часу, що особливо важливо для навчального середовища, де актуальність інформації є критичною.

Після успішного завершення процесу деплою застосунок Teachers Hub став доступним для використання у будь-який час і з будь-якого пристрою — як для викладачів, так і для адміністраторів мовної школи. [30] Усі функції, реалізовані в межах системи (реєстрація, додавання студентів, створення та ведення уроків, контроль оплат, призначення домашніх завдань), повноцінно працюють у публічному середовищі та підтримують усі правила доступу й

безпеки, задані в базі Supabase та сервісі Clerk.

Окрім автоматично згенерованої адреси, Vercel також дозволяє підключити персоналізований домен – наприклад, офіційний домен мовної школи. Хоча в межах цього проєкту таку інтеграцію не було реалізовано, платформа передбачає можливість додавання DNS-записів і сертифікатів безкоштовно, що спрощує майбутнє розширення проєкту або його комерціалізацію.

Публікація на Vercel забезпечила стабільну роботу застосунку без необхідності підтримки серверної інфраструктури, а також гарантує високу доступність, автоматичне масштабування при зростанні навантаження та зручну інтеграцію з іншими сервісами. Застосунок Teachers Hub, розгорнутий у продакшн-середовищі, готовий до подальшого розвитку, масштабування функціоналу та реального використання у навчальному процесі.

ВИСНОВКИ

У межах кваліфікаційної роботи було здійснено моделювання інформаційної системи для управління освітнім процесом у форматі приватного викладання англійської мови. Розробка моделі Teachers Hub дозволила сформувати цілісне уявлення про майбутній застосунок, його логіку, функціональні зв’язки та поведінку системи в умовах реального використання. Завдяки моделюванню вдалося передбачити взаємодію основних компонентів – викладачів, адміністраторів, студентів і бази даних – а також визначити, як саме здійснюється передача даних між інтерфейсом і сховищем інформації без використання окремого бекенду.

Особливу увагу було приділено опису ключових бізнес-процесів: додавання студентів, планування та фіксація уроків, облік оплат і призначення домашніх завдань. Створення логічної моделі дало змогу проаналізувати ці процеси до початку програмної реалізації, усунути зайві кроки й уникнути логічних суперечностей у структурі застосунку.

Також модель стала основою для ухвалення рішень щодо архітектури системи. Було визначено, що застосунок повинен мати модульну структуру з можливістю гнучкого масштабування в майбутньому. Важливою вимогою стала наявність чіткої системи прав доступу, яка гарантує, що кожен користувач бачитиме лише ті дані, які йому дозволені.

Моделювання Teachers Hub дозволило ще до написання коду досягти узгодження між очікуваннями кінцевих користувачів і технічними можливостями реалізації. Це забезпечило високу відповідність системи практичним потребам викладачів і адміністрації мовної школи та створило міцну основу для подальшої розробки функціоналу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Next.js documentation. (2024). Retrieved from https://nextjs.org/docs (accessed 20.05.2025).

2. TypeScript Handbook. (2024). Retrieved from https://www.typescriptlang.org/docs/ (accessed 20.05.2025).

3. Clerk documentation – Authentication for modern apps. (2024). Retrieved from https://clerk.dev/docs (accessed 20.05.2025).

4. Tailwind CSS documentation. (2024). Retrieved from https://tailwindcss.com/docs (accessed 20.05.2025).

5. Radix UI – Primitives for building high-quality UI components. (2024). Retrieved from https://www.radix-ui.com/docs (accessed 20.05.2025).

6. ShadCN UI – Customizable UI components built on Radix. (2024). Retrieved from https://ui.shadcn.com/docs (accessed 20.05.2025).

7. PostgreSQL documentation. (2024). Retrieved from https://www.postgresql.org/docs/ (accessed 20.05.2025).

8. Vercel deployment platform documentation. (2024). Retrieved from https://vercel.com/docs (accessed 20.05.2025).

9. JavaScript SDK – Supabase JavaScript Library. (2024). Retrieved from https://supabase.com/docs/reference/javascript/introduction (accessed 20.05.2025).

10. TypeScript Handbook. (2024). Retrieved from https://www.typescriptlang.org/docs/ (accessed 20.05.2025).

11. React documentation. (2024). Retrieved from https://react.dev/learn (accessed 20.05.2025).

12. WebStorm IDE by JetBrains. (2024). Retrieved from https://www.jetbrains.com/webstorm/documentation/ (accessed 20.05.2025).

13. GitHub Docs – Collaborative development platform. (2024). Retrieved from https://docs.github.com (accessed 20.05.2025).

14. JSON – JavaScript Object Notation. (2024). Retrieved from https://www.json.org/json-en.html (accessed 20.05.2025).

15. Serverless Functions. (2024). Retrieved from https://vercel.com/docs/functions/introduction (accessed 20.05.2025).

16. Planboard – Teacher Planning Platform by Chalk. (2024). Retrieved from https://www.chalk.com/planboard (accessed 20.05.2025).

17. Off2Class – ESL Platform for Teachers. (2024). Retrieved from https://www.off2class.com (accessed 20.05.2025).

18. MySchool – School Management Software. (2024). Retrieved from https://www.myschoolmanagement.com (accessed 20.05.2025).

19. UX Collective. (2023). Designing dashboards for teachers and schools. Retrieved from https://uxdesign.cc (accessed 20.05.2025).

20. Medium. (2023). Building scalable apps with Supabase and Next.js. Retrieved from https://medium.com (accessed 20.05.2025).

21. FreeCodeCamp. (2024). How to deploy a Next.js app with Supabase backend. Retrieved from https://www.freecodecamp.org/news (accessed 20.05.2025).

22. MDN Web Docs. (2024). Working with LocalStorage and IndexedDB. Retrieved from https://developer.mozilla.org (accessed 20.05.2025).

23. Google Developers. (2024). OAuth 2.0 for Web Server Applications. Retrieved from https://developers.google.com/identity/protocols/oauth2 (accessed 20.05.2025).

24. DigitalOcean. (2024). Understanding Serverless Architecture. Retrieved from https://www.digitalocean.com/community/tutorials (accessed 20.05.2025).

25. Stripe. (2024). Online Payments Documentation. Retrieved from https://stripe.com/docs (accessed 20.05.2025).

26. Statista. (2024). Online education platforms usage statistics. Retrieved from https://www.statista.com (accessed 20.05.2025).

27. Supabase Blog – Role-based access control (RBAC) with Supabase. (2024). Retrieved from https://supabase.com/blog/role-based-access-control (accessed 20.05.2025).

28. Smashing Magazine – Building Accessible React Apps. (2024). Retrieved from https://www.smashingmagazine.com/2024/01/accessible-react-apps/ (accessed 20.05.2025).

29. CSS-Tricks. (2024). Best practices for responsive UI design with Tailwind CSS. Retrieved from https://css-tricks.com (accessed 20.05.2025).

30. Dev.to. (2024). Implementing authentication with Clerk and Supabase in Next.js. Retrieved from https://dev.to (accessed 20.05.2025).