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:

«Онлайн-платформа дистанційного навчання»

Виконала студентка групи ІПЗ-20-2
ГОЛЯЧЕНКО Аліна Анатоліївна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2024

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
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОЛЯЧЕНКО Аліна Анатоліївна**
Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**
Тема роботи: **«Онлайн-платформа дистанційного навчання»**,
затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**
Вихідні дані для роботи: **розробка онлайн-платформи дистанційного навчання зосереджена на забезпеченні учнів навчальних закладів можливістю навчатись в домашніх умовах. Платформа має надати учням та вчителям зручний доступ до інформації про навчальний процес та забезпечити комунікацію для уможливлення навчання на відстані.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Чижмотря О .В.	27.02.2024р.	27.02.2024р.
2	Чижмотря О .В.	27.02.2024р.	27.02.2024р.
3	Чижмотря О .В.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра включає онлайн-платформу дистанційного навчання та пояснювальну записку. Пояснювальна записка складається з тексту обсягом 68 сторінок друкованого тексту, 32 сторінок додатків, списку використаних джерел на 30 найменувань, що займає 3 сторінки. Робота містить 45 рисунків і 17 таблиць, загальний обсяг роботи становить 103 сторінки.

Мета роботи – розробка онлайн-платформи дистанційного навчання для сприяння покращенню якості освіти та оптимізації навчального процесу в умовах відсутності можливості відвідувати заклади освіти очно.

Під час виконання роботи було досліджено подібні веб-додатки, проаналізовано їх функціональні можливості та недоліки.

Результатом роботи стала розробка онлайн-платформи для дистанційного навчання, яка значно полегшує навчальний процес як для учнів, так і для вчителів, забезпечуючи простий та інтуїтивно зрозумілий інтерфейс. Платформа дозволяє учням легко організовувати свій навчальний процес та здійснювати комунікацію з вчителями та іншими учнями школи.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-ПЛАТФОРМА, ЧАТ, НАВЧАННЯ, ДИСТАНЦІЙНЕ НАВЧАННЯ, БАЗА ДАНИХ, ВЕБ-ДОДАТОК, КОРИСТУВАЧ.

ABSTRACT

Bachelor's Thesis includes an online distance learning platform and an explanatory note. The explanatory note consists of 68 pages of printed text, 32 pages of appendices, a list of 30 references occupying 3 pages. The work contains 45 figures and 17 tables, with a total volume of 103 pages.

The aim of the work is to develop an online distance learning platform to improve the quality of education and optimize the learning process in the absence of the ability to attend educational institutions in person.

During the work, similar web applications were researched, their functional capabilities and shortcomings were analyzed.

The result of the work is the development of an online distance learning platform that significantly facilitates the learning process for both students and teachers by providing a simple and intuitive interface. The platform allows students to easily organize their learning process and communicate with teachers and other students.

KEYWORDS: ONLINE PLATFORM, CHAT, LEARNING, DISTANCE LEARNING, DATABASE, WEB APPLICATION, USER.

ЗМІСТ

РЕФЕРАТ **3**

ABSTRACT **4**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ **6**

ВСТУП **7**

Розділ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ ОНЛАЙН-ПЛАТФОРМИ ДИСТАНЦІЙНОГО НАВЧАННЯ **9**

1.1 Постановка задачі 9

1.2 Аналіз аналогів програмного продукту 10

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 16

Висновки до першого розділу 19

Розділ 2. ПРОЕКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ **21**

2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи 21

2.2 Розробка бази даних системи 27

2.3 Проектування та реалізація алгоритмів роботи системи 37

2.4 Реалізація онлайн-платформи дистанційного навчання 40

Висновки до другого розділу 46

Розділ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-ПЛАТФОРМОЮ ДИСТАНЦІЙНОГО НАВЧАННЯ **48**

3.1 Порядок встановлення та налаштування параметрів платформи 48

3.2 Структура інтерфейсу онлайн-платформи дистанційного навчання ..48

3.3 Тестування платформи 61

Висновки до третього розділу 67

ВИСНОВКИ **68**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **69**

ДОДАТКИ **72**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DOM – Document Object Model

API – прикладний програмний інтерфейс

PK – первинний ключ (Primary Key)

FK – зовнішній ключ (Foreign Key)

JSON – формат обміну даними (JavaScript Object Notation)

CRUD – основні операції для роботи з даними (створення, читання, оновлення, видалення)

UI/UX – Дизайн користувацького інтерфейсу та користувацький досвід

SQL – структурована мова запитів (Structured Query Language)

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

JWT – стандартизований формат токенів у формі JSON для передачі інформації

SEO – Пошукова оптимізація (Search Engine Optimization)

ВСТУП

Актуальність теми розробки онлайн-платформи дистанційного навчання є надзвичайно важливою в сучасних умовах. Під час піндемії COVID-19 неабияк зросла популярність платформ для навчання онлайн, адже люди шукали можливість отримувати знання в умовах, коли очна взаємодія з викладачами неможлива. А російська агресія привела до того, що діти вимушені не відвідувати школи задля власної безпеки. Розробка онлайн-платформи, яка забезпечує якісний та неперервний доступ до освіти та зручність використання для учнів і вчителів, не лише відповідає сучасним вимогам, але й відкриває нові перспективи для покращення навчального процесу і забезпечення доступу до освіти в умовах обмежень.

Метою цього дослідження є створення ефективної та функціональної онлайн-платформи дистанційного навчання з використанням передових технологій розробки. Основними завданнями є розробка веб-додатку із зручним інтерфейсом, який дозволить навіть дітям легко користуватись ним та забезпечить їм доступ до навчання у будь-який час та в будь-якому місці. Також важливо дослідити можливості і вплив сучасних технологій на удосконалення якості освіти та результативність навчального процесу.

Для розробки онлайн-платформи дистанційного навчання потрібно:

1. Провести аналіз конкурентів і оцінити їх функціональні можливості, щоб визначити слабкі сторони. Це допоможе зрозуміти, як вдосконалити кінцевий продукт, яких функцій потребують користувачі, як зробити веб-додаток більш зручним та комфортним у використанні.
2. Визначити технології, які варто використати для розробки платформи. Для забезпечення швидкодії та надійності потрібно уважно обирати інструменти для створення клієнтської та серверної частин, а також бази даних.
3. Розробити онлайн-платформу, враховуючи зібрану інформацію. А саме: створити зручний та простий у розумінні інтерфейс, забезпечити

додаток функціями, найбільш потрібними користувачеві платформи, і упевнитись у безпеці користувацьких даних.

Об'єктом дослідження є дистанційне навчання в контексті сучасних веб-технологій. Досліджується увесь процес навчання, включаючи взаємодію учнів і вчителів та керування навчальним процесом. Також досліджується вплив веб-технологій на ефективність освітнього процесу. Серед них – Next.js та WebSocket (зокрема бібліотека Socket.io).

Предметом дослідження є створення та впровадження онлайн-платформи дистанційного навчання, яка буде заснована на новітніх технологіях розробки та відповідатиме нагальним вимогам сучасного освітнього середовища. Досліджуються функції даної платформи, її можливості та чим вона краща за інші засоби дистанційного навчання.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ ОНЛАЙН-ПЛАТФОРМИ ДИСТАНЦІЙНОГО НАВЧАННЯ

1.1 Постановка задачі

Організація навчального процесу це непроста задача навіть в умовах очного навчання. А враховуючи, що в умовах дистанційного навчання учням і так важко відчути себе “на уроці”, при використанні в якості освітньої платформи, скажімо, Viber, організувати всіх учнів до виконання спільної класної та домашньої роботи ще важче.

Через це з’являється нагальна потреба у забезпеченні зручного та неперервного доступу до освітнього середовища.

Цільове призначення веб-додатку:

- Забезпечити можливостю навчатись усіх, хто має доступ до Інтернету, незалежно від географічного положення та інших чинників.
- Надати можливість оцінки успішності та відвідуваності через інтерактивний журнал.
- Забезпечити можливість комунікації між користувачами з одного навчального закладу.
- Надати можливість створювати домашні завдання.
- Надати засоби керування інформацією про себе або про інших користувачів та навчальні заклади, за наявністю відповідних прав доступу.

Основні етапи розробки веб-додатку:

Аналіз вимог:

- Визначити конкурентне програмне забезпечення на ринку та провести аналіз.
- Визначити потреби користувачів платформ дистанційного навчання.
- Визначити функціональні та нефункціональні вимоги до системи.

– Розробити сценарії використання та визначити основні потреби користувачів платформи.

Проектування інтерфейсу:

– Визначити структуру та навігацію платформи.
– Розробити дизайн інтерфейсу, враховуючи аспекти користувацього досвіду (UX) [22].

Розробка основного функціоналу:

– Створення логіки авторизації існуючих користувачів, та реєстрацію інших користувачів через обліковий засіб адміністратора.
– Створення бази даних та основних операцій з даними (CRUD).
– Розмежування доступу за ролями.
– Створення адміністративної панелі для різних типів адміністраторів.
– Розробка взаємодії користувачів з різними рівнями доступу з додатком.

Завдяки виконанню даних кроків буде створена якісна та проста у використанні платформа, яка надасть користувачам доступ до освітнього процесу навіть на відстані.

1.2 Аналіз аналогів програмного продукту

Перш ніж розпочати розробку онлайн-платформи дистанційного навчання потрібно проаналізувати конкурентні інструменти для навчання на ринку. Це дозволить краще спланувати розробку, включаючи усі необхідні користувачам функції та можливості, а також виявити потенційні недоліки та шляхи їх вирішення.

Перша конкурентна веб-платформа яку потрібно розглянути – це Всеосвіта (рис. 1.1) [1]. Всеосвіта – це онлайн-платформа, яка пропонує широкий спектр освітніх ресурсів для українських вчителів та учнів. Серед них плани уроків, методичні розробки, електронні підручники тощо. Інтерфейс є зручним у використанні та не перевантажений. Платформа також надає можливість вчителям та учням спілкуватись.

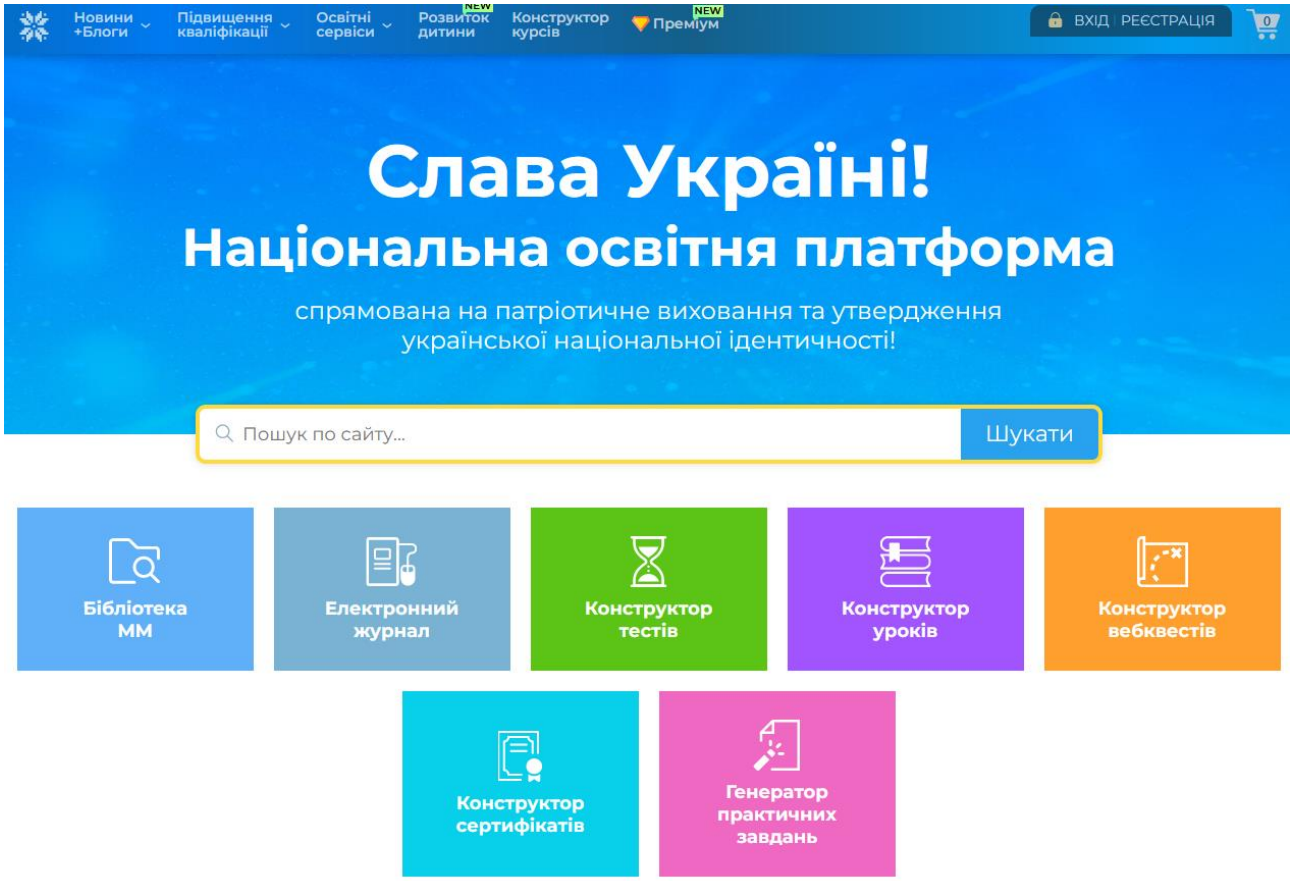


Рисунок 1.1 – Інтерфейс веб-платформи Всеосвіта

Переваги:

– Широкий спектр ресурсів: Вчителі та учні можуть знайти матеріали на свій смак та використовувати їх для навчання.
– Зручний інтеріейс: Завдяки зрозумілому інтерфейсу використовувати даний веб-додаток буде нескладно.
– Спілкування: Всеосвіта надає можливість комунікації, що дозволить учням отримувати відповіді на їх запитання.
– Багато корисних функцій: Платформа надає доступ до різноманітних функцій, корисних як вчителям, так і учням. А саме: електронний журнал, конструктор тестів та уроків, генератор практичних завдань тощо.

Недоліки:

– Якість ресурсів: Деякі ресурси є дуже високоякісними, але теж саме не можна сказати про інші.

– Модерація: На платформі немає достатнього рівня модерації, через що може бути опубліковано неякісний або навіть шкідливий контент.

Всеосвіта є цінним ресурсом як для учнів, так і для вчителів. Вона має багато необхідного функціоналу та є зручною у використанні. Завдяки можливості листуватись учні та вчителі зможуть комунікувати між собою, а функції оцінки успішності та перегляду розкладів полегшать організацію освітнього процесу.

Наступною платформою, яку було проаналізовано, є Єдина Школа. (рис 1.2) [2]. Єдина школа – це комплексна онлайн-платформа, призначена для оптимізації освітнього процесу в українських закладах освіти. Вона пропонує широкий спектр функцій для вчителів, учнів та батьків, роблячи навчання більш зручним, ефективним та прозорим.

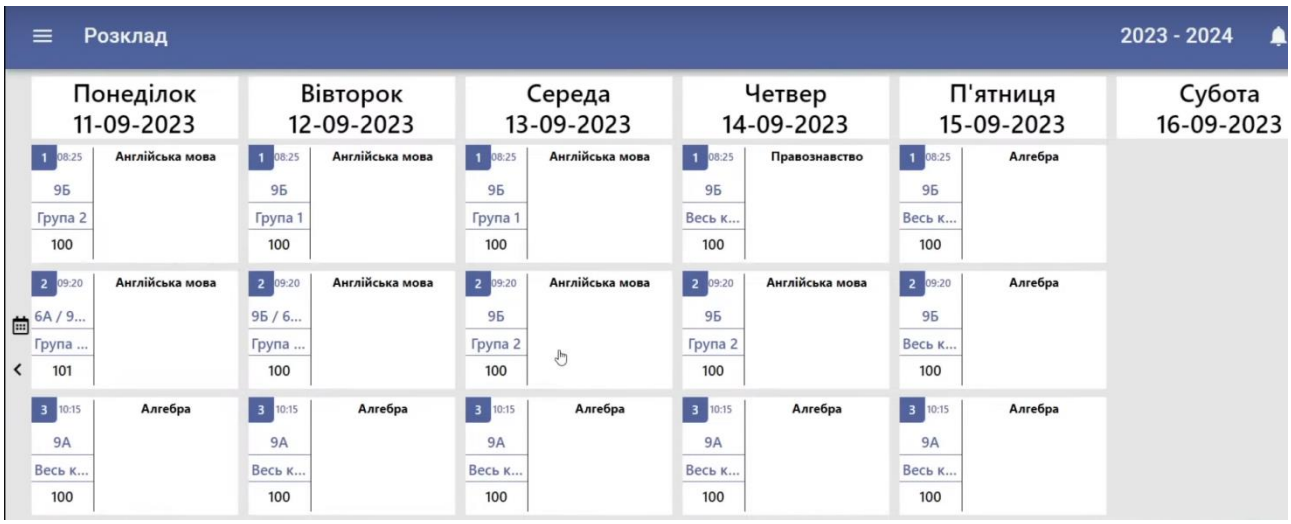


Рисунок 1.2 – Інтерфейс онлайн-платформи Єдина Школа

Переваги:

– Спілкування: Платформа полегшує спілкування між вчителями, учнями та батьками, що може призвести до кращого розуміння та співпраці.
– Підвищення ефективності навчання: Онлайн-уроки, електронні щоденники та інші інструменти можуть допомогти вчителям зробити навчання більш ефективним та цікавим для учнів.

– Економія часу: Платформа автоматизує багато завдань, які вчителі зазвичай виконують вручну, що може звільнити їм час для більш важливих справ.
– Мобільний додаток: Платформою можна користуватись чреез смартфон, що підвищує зручність використання.

Недоліки:

– Складність використання: Платформа складна у використанні, особливо для тих, хто не має досвіду роботи з онлайн-платформами навчання. Тому потрібен буде час та відповідальна особа, щоб навчити як учнів так і вчителів користуватись даною платформою.
– Проблемний інтерфейс: Інтерфейс даної платформи є перевантаженим та нехтує принципами користувацького досвіду (UX) [23], що ускладнює її використання.
– Обмежений функціонал мобільного додатку: Хоч додаток і пдівищує зручність використання в моменти, до прикладу, відсутності електроенергії, проте його функціонал обмежений порівняно з веб-версією.
– Фокус на адмініструванні: Платформа більше розрахована на адміністрування навчального закладу та аналітику освітнього процесу, ніж на покращення дистанційного навчання для учнів.

Остання платформа дистанційного навчання яку було розглянуто – це EdEra (рис. 1.3) [3]. Це українська онлайн-платформа освіти, яка пропонує широкий спектр навчальних послуг для учнів 5-11 класів. Її мета – зробити якісну освіту доступною для всіх українців.

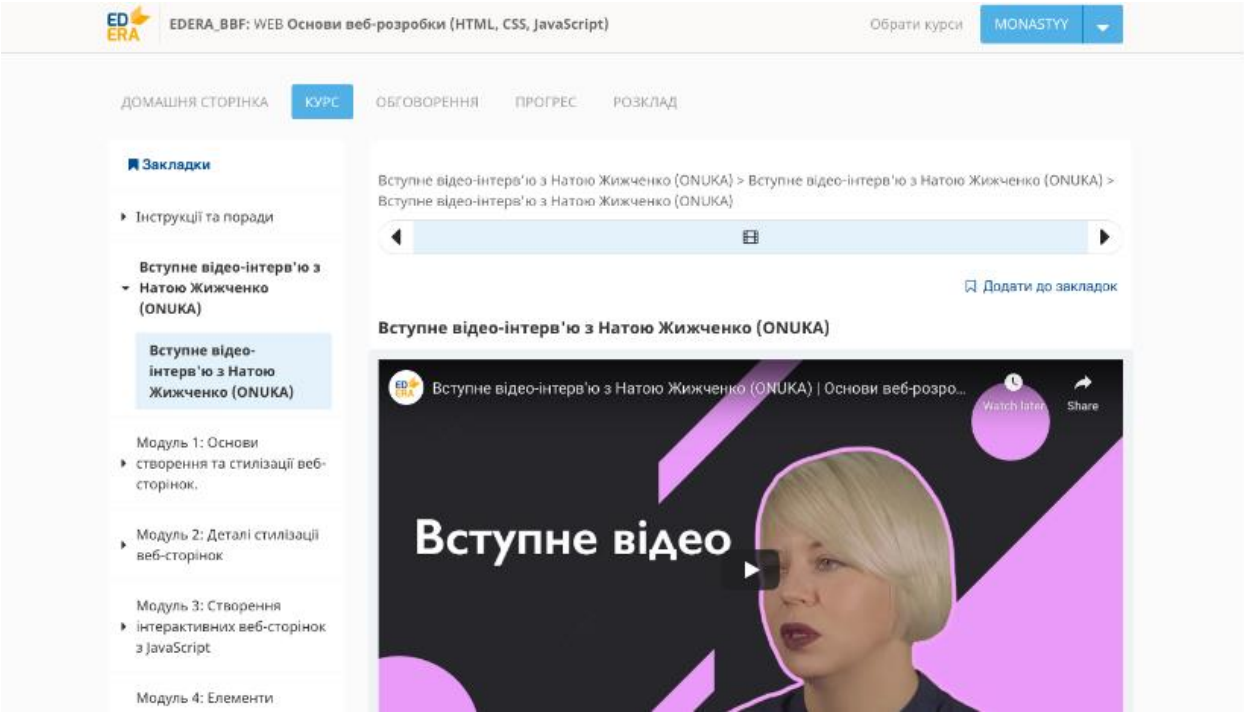


Рисунок 1.3 – Інтерфейс платформи EdEra

Переваги:

– Онлайн-курси: EdEra пропонує широкий спектр онлайн-курсів з різних предметів. Всі вони розроблені кваліфікованими вчителями та відповідають шкільній програмі.
– Тестування: Учні можуть проходити тести з різних предметів, щоб перевірити свої знання та отримати зворотний зв'язок.
– Відстеження успішності: Учні та батьки можуть відстежувати успішність учня за допомогою електронного щоденника.
– Додаткові матеріали: На платформі доступні методичні розробки, презентації, електронні підручники та інші навчальні матеріали.

Недоліки:

– Обмежений набір предметів: На платформі представлені не всі предмети, які вивчаються в школі.
– Обмежена взаємодія: На платформі немає можливості для спілкування між учнями та вчителями.

– Вікові обмеження: Платформа орієнтована лише на учнів 5-11 класів, тоді як потребу в зручній навчальній платформі мають і учні молодших класів.

Усі вищеописані освітні платформи зроблені для покращення освітнього процесу в умовах дистанційного навчання. Нижче наведено порівняльну таблицю (табл. 1.1) даних платформ.

Таблиця 1.1

Порівняння інструментів освітніх платформ

	Всеосвіта	Єдина Школа	EdEra
Тестування	+	+	+
Відстеження успішності	+	+	+
Додаткові матеріали	+	+	+
Персоналізація	-	-	+
Спілкування	+	+	-
Онлайн-уроки	-	-	-
Наявність усіх шкільних предметів	+	+	-
Розклад уроків	+	+	-
Базові функції адміністрування навчального закладу	+	+	-

Отже Всеосвіта пропонує широкий спектр функцій для вчителів, учнів, батьків та шкіл, включаючи онлайн-курси, тестування, оцінювання, відстеження успішності, додаткові матеріали, спілкування та базове адміністрування школи. Єдина школа орієнтована на адміністрування шкіл, надаючи інструменти для ведення обліку учнів, формування класів, розподілу навантажень на вчителів та доступу до розкладу уроків. EdEra, з іншого боку, фокусується на онлайн-курсах з обмеженого набору предметів, але також пропонує функції тестування, оцінювання та відстеження успішності.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментальних засобів для реалізації онлайн-платформи дистанційного навчання ґрунтується на декількох ключових факторах. Функціональність та масштабованість забезпечить фреймворк Next.js, в основі якого лежить бібліотека React.js та Node.js – середовище виконання мови JavaScript [30]. Зручність використання забезпечить бібліотека React.js та SCSS – препроцесор мови CSS. Вони дозволять створити зручний користувацький інтерфейс. Надійність платформи забезпечить СУБД MySQL. Це популярна система управління базами даних, яка є масштабованою, безпечною та надійною. Спілкування між користувачами буде забезпечено технологією WebSocket, а саме бібліотекою Socket.io.

Вимоги до апаратного забезпечення повинні буди достатніми для забезпечення доступу до великого обсягу даних та оптимальної швидкості використання платформи.

Next.js є JavaScript-фреймворком, що поєднує в собі серверний рендеринг, статичну генерацію сторінок та маршрутизацію [4]. Це робить його оптимальним вибором для розробки динамічних веб-додатків.

Переваги Next.js:

- Серверний рендеринг: Next.js генерує HTML на сервері, що робить його більш SEO-дружнім та покращує час завантаження сторінок. Також це робить його кращим вибором для сайтів, які потребують високої ранжируваності в пошукових системах та швидкого завантаження для користувачів.
- Статична генерація сторінок: Next.js може генерувати статичні сторінки під час збирання, роблячи їх ще швидшими та економнішими з точки зору ресурсів. Це особливо корисно для сторінок, які не потребують динамічних даних, таких як інформаційні сторінки, наприкла, розклад занять.

– Проста маршрутизація: Next.js використовує власну систему маршрутизації, яка робить його простим у використанні та налаштуванні. Це полегшує створення структури сайту та навігації по ньому.

Node.js – Next.js використовує Node.js для виконання свого коду на сервері. Це середовище виконання JavaScript, яке дозволяє запускати JavaScript-код поза браузером [5]. Він побудований на потужному JavaScript-двигуні V8 від Chrome, що забезпечує ефективне та високопродуктивне виконання JavaScript-коду.

Переваги Node.js:

– Неблокуючий ввід-вивід: Node.js використовує неблокуючу модель вводу-виводу, що означає, що він може обробляти декілька запитів одночасно, не чекаючи завершення кожного з них. Це робить його дуже ефективним для обробки великої кількості запитів, роблячи його ідеальним для веб-додатків з високим трафіком.
– Архітектура, керована подіями: Node.js використовує архітектуру, керовану подіями, яка асинхронно обробляє запити. Це дозволяє йому ефективно керувати великою кількістю одночасних з'єднань, не витрачаючи багато ресурсів.

React.js – це JavaScript-бібліотека для створення динамічних інтерфейсів користувачів (UI) веб-застосунків [7]. React.js використовується для створення повторно використовуваних компонентів, які можна складати для створення складних інтерфейсів.

Переваги React.js:

– Компонентний підхід: React використовує компонентний підхід, який робить код більш модульним, гнучким та легким у підтримці.
– Багатий набір бібліотек: Існує безліч бібліотек та інструментів, які можна використовувати з React, що робить його доволі потужною бібліотекою.

– Віртуальний DOM: React використовує віртуальний DOM, який робить оновлення інтерфейсу користувача більш ефективними.

Socket.io – це бібліотека JavaScript, яка дозволяє встановлювати двосторонній зв'язок у реальному часі між клієнтом та сервером [6]. Це робить його ідеальним для створення динамічних веб-додатків, які потребують постійного оновлення даних, таких як чати на різних платформах.

Серед переваг Socket.io:

– Двосторонній зв'язок: Socket.io дозволяє серверу та клієнту “спілкуватися” в реальному часі. Це робить його ідеальним для реалізації чатів та інших функцій, які потребують постійного обміну даними.
– Низька затримка: Socket.io використовує WebSockets або HTTP з довгим опитуванням для встановлення з'єднання, що робить його дуже швидким та з низькою затримкою. Це важливо для забезпечення чутливого та швидкого реагування на дії користувачів.
– Масштабованість: Socket.io може обробляти велику кількість користувачів, що робить його ідеальним для великих веб-додатків. Це пов'язано з його архітектурою, яка базується на подіях та може обробляти багато одночасних з'єднань.

MySQL – це популярна система управління реляційними базами даних (СУБД), яка використовується для зберігання та керування даними [9]. Це надійне та масштабоване рішення, яке може обробляти великі обсяги даних.

Основні переваги MySQL:

– Надійність: MySQL – це перевірена та надійна СУБД, яка використовується протягом багатьох років. Це робить її стійкою до збоїв та гарантує безпечне зберігання даних.
– Масштабованість: MySQL є дуже потужною та може обробляти великі обсяги даних та велику кількість користувачів.

– Широкий функціонал: MySQL пропонує широкий спектр функціональних можливостей, які роблять її ідеальним вибором для розробки онлайн-платформи. Окрім базових CRUD-операцій серед них – транзакції, процедури, та багато іншого.

SCSS – це препроцесор CSS, який дозволяє використовувати змінні, функції та міксини, що робить код CSS більш організованим та керованим [8].

Переваги SCSS:

– Організованість: SCSS допомагає організувати код CSS, що робить його більш читабельним та простішим у підтримці. Це особливо корисно для великих проектів з багатьма CSS-файлами.
– Керованість: SCSS дозволяє використовувати змінні, функції та міксини для повторного використання коду. Це робить код CSS більш компактним та лаконічним, а також полегшує його модифікацію.
– Стислість: SCSS можна компілювати в звичайний CSS, що робить його більш стислим та ефективним. Завдяки цьому сторінки завантажуються швидше.

Висновки до першого розділу

Проведено аналіз трьох популярних онлайн-платформ дистанційного навчання: Всеосвіта, Єдина Школа та EdEra. Аналіз охопив функціональні можливості, педагогічні інструменти та загальну зручність використання платформ. "Всеосвіта" добре підходить для створення та публікації навчальних матеріалів, проведення онлайн-уроків та тестування знань учнів. Проте, якість її ресурсів важко перевірити, а недостатня модерація та безпека може привести до завантаження шкідливого контенту. "Єдина Школа" пропонує широкий спектр інструментів для спілкування та співпраці між учителями та учнями. Однак, платформа більше розрахована на керування навчальним закладом, ніж на зручність здобуття освіти дистанційно, а складність інтерфейсу та платформи вцілому робить використання дуже складним та потребує попереднього

навчання. EdEra фокусується на створенні інтерактивних курсів та гейміфікації навчання. Проте, платформа не надає можливості спілкування між учнями та вчителями та має обмежений педагогічний функціонал в контексті навчання у навчальному закладі.

Зважаючи на переваги та недоліки кожної з платформ, а також на специфічні потреби проекту, було обрано наступний набір інструментів для розробки онлайн-платформи дистанційного навчання: Next.js (разом з React.js та Node.js), MySQL, Socket.io та SCSS. Дані технології забезпечать ефективність, надійність, функціональність та продуктивність платформи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ

2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи

Розпочинаючи розробку онлайн-платформи, критично важливо чітко окреслити її майбутній функціонал. Це потребує ретельного аналізу потреб користувачів та освітніх закладів, щоб створити продукт, який буде одночасно корисним та потрібним.

Створення прецедентів використання, або сценаріїв взаємодії користувачів з додатком, є важливим кроком перед розробкою платформи та її бази даних. Вони дають чітке розуміння потреб користувачів та необхідних функцій, що дозволяє оптимізувати структуру бази даних та визначити, які дані потребують зберігання

Відповідність структури бази даних потребам функціоналу додатку є ключовим фактором для ефективного зберігання та доступу до інформації. Це мінімізує ризик виникнення неузгодженостей між функціоналом та базою даних. Таким чином процес розробки буде простішим, а підтримувати платформу в майбутньому буде значно легше [29].

Вимоги користувачів:

1. Редагування персональних даних
2. Для вчителів – доступ до педагогічного функціоналу, а саме перегляд класів та учнів, власного та інших розкладів, створення, редагування та видалення домашнього завдання тощо
3. Для учнів – перегляд розкладу занять свого класу, перегляд домашнього завдання, власної успішності і так далі
4. Для адміністратора платформи: керування інформацією про користувачів та навчальні заклади. Реєстрація нового навчального закладу, створення користувачів з правами адміністрітора навчального закладу

5. Для адміністратора навчального закладу – керування інформацією про вчителів та учнів закладу, про наявні класи, керування розкладом і так далі
6. Зручний інтерфейс
7. Електронне листування через чати
8. Перегляд профілів інших користувачів

Функціональні вимоги:

1. Реєстрація та авторизація
2. Електронне листування через чати
3. Створення, редагування та видалення домашніх завдань, оцінок, предметів у розкладі, предметів як таких, класів тощо
4. Редагування, видалення та додавання нових навчальних закладів та користувачів
5. Вид профілю користувача зі сторони як самого користувача, так і інших користувачів платформи
6. Зміна пароля
7. Активація акаунту через електронну пошту

Нефункціональні вимоги:

1. Безпека:
   - Хешування паролів
   - Захист від несанкціанованого доступу
   - Обмеження доступу до певних сторінок
2. Надійність:
   - Стійкість до підвищення навантаження.
3. Продуктивність:
   - Швидкість завантаження сторінок
   - Масштабованість

Системні вимоги:

1. Вимоги до середовища виконання:

- Оперативна пам'ять: мінімум 2 ГБ.
- Процесор: тактова частота не менше 1.8 ГГц.
- Підключення до мережі Інтернет: підключення має бути стабільним.
- Веб-браузер: будь-який із популярних браузерів.

Прецедент «Реєстрація користувача»

Опис: Адміністратор надає необхідну інформацію для створення нового користувача та відправляє її через форму.

Кроки виконання:

1. Вхід на сайт
2. Заповнення поля реєстрації
3. Підтвердження реєстрації кнопкою «Зареєструватися»
4. Валідація введених даних
5. Створення нового користувача
6. Користувач наразі має змогу увійти в систему

Виключні ситуації:

- Якщо дана пошта або нікнейм зайняті іншим користувачем виводиться помилка
- Якщо введені дані не відповідають валідації реєстрація неможлива

Прецедент «Авторизація»

Опис: Користувач здійснює вхід на сайт через сторінку авторизації.

Кроки виконання:

1. Неавторизований користувач потрпляє на сторінку авторизації
2. Він заповнює форму, ввівши електронну пошту та пароль
3. Після натискання користувачем кнопки “Увійти” здійснюється вхід в акаунт
4. У cookies записується JWT-токен, що позначає авторизованого користувача

Виключні ситуації:

- Користувач залишив хоча б одне поле пустим і авторизацію не було здійснено
- Під час серверної валідації було виявлено, що пароль невірний або користувача з вказаним email не існує, і авторизація не відбулася

Прецедент «Відновлення паролю»

Опис: Користувач забув свій пароль та намагається його відновити.

Кроки відтворення:

1. На сторінці авторизації користувач натискає на напис-посилання «Забули пароль?»
2. Користувач перенаправляється на сторінку з формою введення електронної пошти
3. Користувач натискає кнопку “Підтвердити”
4. Користувачеві на пошту приходить лист з посиланням на сторінку з формою скидування паролю
5. При переході за посиланням відкривається форма для введення нового паролю
6. Після введення даних та натиснення кнопки “Підтвердити” здійснюється зміна паролю на вказаний

Виключні ситуації:

- Користувач залишив хоча б одне поле пустим і пароль не було скинуто
- Строк дії листа вийшов, користувачеві потрібно повторити процедуру

Прецедент «Редагування особистих даних»

Опис: Користувач редагує дані у своєму профілі.

Кроки відтворення:

1. Користувач заходить у свій особистий кабінет
2. Доступні дві форми: одна для зміни особистих даних, інша для зміни паролю

3. При заповненні однієї з форм та натисненні кнопки “Зберегти” відбувається оновлення даних користувача

Виключні ситуації:

- При введенні некоректних даних спрацьовує валідація та оновлення даних не відбувається

Прецедент «Контроль успішності та відвідуваності»

Опис: Вчитель хоче виставити оцінку або відмітити відсутність учня на уроці.

Кроки відтворення:

1. Якщо користувач є вчителем, він може перейти на вкладку “Оцінки”, на якій розміщено електронний журнал
2. Користувач двічі клікає на будь-яку клітинку журналу для зміни оцінки, або виставленя оцінки чи “н-ки”
3. Користувач підтверджує операцію шляхом натискання кнопки “Enter”

Виключні ситуації:

- При введенні оцінки, меншої за 1 чи більшої за 12, або коли не обрано предмет, з якого виставляється оцінка, дія не відбувається, а користувач отримує попередження у вигляді сповіщення.

Після опису базових прецедентів можна перейти до створення бази даних, але перед цим варто визначити її структуру та потрібні таблиці (рис. 2.1). Базу даних потрібно моделювати зважаючи на потреби учасників освітнього процесу. Створення схеми бази даних та зв’язків між її таблицями дозволить унаочнити взаємодію між різними сутностями системи та убереже від помилок та неузгодженостей під час розробки.

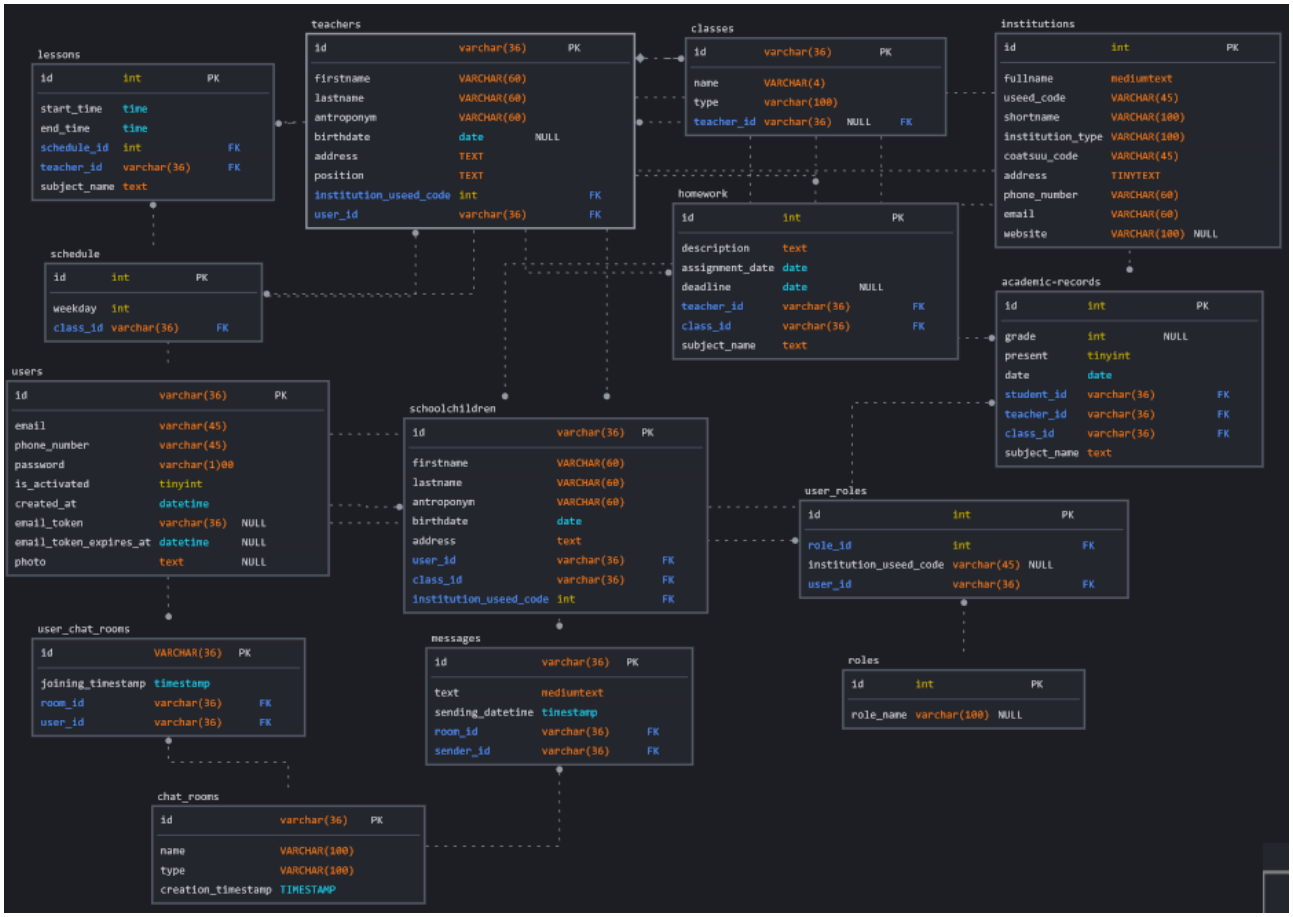


Рисунок 2.1 – Початкова діаграма

Базова діаграма бази даних складається з таких таблиць:

1. Users – дана таблиця міститиме базову інформацію про користувача: email, номер телефону, пароль тощо.
2. Schoolchildren – ця таблиця посилається на Users та містить додаткову інформацію, що стосується учні. Наприклад, адреса, код ЄДЕБО школи та посилання на клас.
3. Teachers – за суттю ця таблиця така сама, як і попередня. Вона теж посилається на Users, але також містить інформацію про вчителя. Наприклад, код ЄДЕБО школи, посада, адреса тощо.
4. User_roles – ця таблиця містить ролі кожного користувача та зберігає можливість масштабування за потреби значного розширення платформи.
5. Roles – на цю таблицю посилається User_roles. Вона містить назви ролей користувачів.

6. Schedule – таблиця, що представляє розклад для певного класу на певний день тижня.
7. Lesson – дана таблиця представляє собою заняття в у певний час та певному розкладі, яке проводить призначений вчитель.
8. Homework – містить інформацію про домашнє завдання та хто з вчителів його задав.
9. Academic_records – ця таблиця зберігає дані про успішність учня, а саме оцінку, чи було відвідане заняття, хто з вчителів виставив оцінку тощо.
10. Institutions – у цій таблиці зберігаються дані про навчальний заклад, такі як назва, адреса, код ЄДЕБО та інші.
11. Chat_rooms – дана таблиця містить інформацію про “кімнату” для листувань між користувачами. На неї посилається таблиця нижче.
12. Messages – таблиця, що представляє собою повідомлення. Вона містить текст повідомлення, ідентифікатор відправника, посилання на чат-кімнату, час відправки та інше.

2.2 Розробка бази даних системи

Для збереження інформації розробляєма платформа використовує MySQL [28]. Це система, що дозволяє керувати реляційною базою даних. База даних системи складається з шістнадцяти таблиць:

1. Users – у цій таблиці зберігається інформація, спільна для усіх користувачів. Наприклад, електронна пошта, пароль, номер телефону, посилання на фото профілю і так далі.
2. Schoolchildren та Teachers доповнюють таблицю Users та містять дані користувача, які можуть різнитись в залежності від його ролі. Наприклад, в таблиці Schoolchildren окрім загальної інформації міститься ще посилання на клас, в якому вчиться учени, а в таблиці Teachers окрім решти міститься також посада вчителя.

3. Classes – це таблиця, яка представляє собою клас, в якому навчаються учні. У даній таблиці буде назва класу та його тип (молодша чи старша школа)
4. Roles – дана таблиця зберігає різні ролі користувачів.
5. User_roles – ця таблиця посилається на таблицю Roles та зв'язує роль з ідентифікатором користувача.
6. Schedule – це таблиця, що представляє розклад для певного класу на певний день тижня.
7. Lesson – це таблиця, яка представляє собою заняття в у певний час та за певним розкладом. Вона також містить посилання на викладача уроку.
8. Subjects – дана таблиця містить назви предметів та до яких класів, старших чи молодших, вони відносяться.
9. Homework – це таблиця, яка містить інформацію про домашнє завдання, хто з вчителів його задав, до коли потрібно зробити та з якого предмету.
10. Academic_records – ця таблиця зберігає оцінки та відвідуваність учнями кожного уроку за певним предметом. Вона містить посилання на вчителя, що виставив оцінку, а також поля, що позначають чи було заняття відвідане та яку оцінку отримав конкретний учень.
11. Institutions – у цій таблиці зберігаються усі потрібні дані про навчальний заклад. Серед них назва, адреса, код ЄДЕБО та інші. Ця таблиця буде попередньо заповнена для того, щоб користувачам було зручніше шукати необхідний заклад, не витрачаючи час на заповнення інформації.
12. Settlements – це таблиця, яка містить усі населені пункти України та інформацію для них, щоб генералізувати формат внесення населеного пункту та дати користувачам можливість обрати його з випадаючого списку.
13. Chat_rooms – дана таблиця містить інформацію про "кімнату", де відбувається листування користувачів.

14. User_chat_rooms – це таблиця, яка забезпечує зв’язок багато до багатьох між користувачами та кімнатами листувань. Це надає можливість користувачеві бути членом різних кімнат для листувань.
15. Messages – це таблиця, що представляє повідомлення. В ній є такі поля як текст повідомлення, ідентифікатор відправника, посилання на кімнату листування, час відправки та інше.

Опис кожної з таблиць бази даних наведений у таблицях 2.1 – 2.16.

Одна з головних таблиць це «Users» (Користувачі). Унікальний ідентифікатор користувача представлений у вигляді випадково згенерованої послідовності. Він використовуватиметься у різних таблицях для забезпечення зв’язку з даними користувача.

Таблиця 2.1

Опис таблиці Users

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	varchar	+	-	Ідентифікатор
Email	varchar	-	-	Електронна пошта
Phone_number	varchar	-	-	Номер телефону
Password	varchar	-	-	Пароль
Is_activated	tinyint	-	-	Статус активації акаунту
Created_at	datetime	-	-	Дата створення
Email_token	varchar	-	-	Активаційний токен
Email_token_expires_at	datetime	-	-	Дата, до якої дійсний токен
photo	text	-	-	Посилання на фото

Таблиця «Schoolchildren» (учні) пов’язана з таблицею Users. Вона містить інформацію про учнів, що навчаються в конкретному навчальному закладі.

Таблиця 2.2

Опис таблиці Schoolchildren

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	varchar	+	-	Ідентифікатор

Firstname	varchar	-	-	Ім’я
Lastname	varchar	-	-	Прізвище
Antroponym	varchar	-	-	По-батькові чи по-матері
Birthdate	date	-	-	Дата народження
Country	varchar	-	-	Країна
Region	varchar	-	-	Регіон (область)
Settlement	mediumtext	-	-	Населений пункт
Address	text	-	-	Адреса
User_id	varchar	-	+	Ідентифікатор користувача
Institution_useed_code	varchar	-	+	Код ЄДЕБО навчального закладу
Class_id	varchar	-	+	Ідентифікатор класу

Таблиця «Teachers» (вчителі) також пов’язана з таблицею «Users». Вона містить інформацію про вчителів конкретного навчального закладу.

Таблиця 2.3

Опис таблиці Teachers

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	varchar	+	-	Ідентифікатор
Firstname	varchar	-	-	Ім’я
Lastname	varchar	-	-	Прізвище
Antroponym	varchar	-	-	По-батькові або по матері
Birthdate	date	-	-	Дата народження
Country	varchar	-	-	Країна
Region	varchar	-	-	Регіон (область)
Settlement	mediumtext	-	-	Населений пункт
Address	text	-	-	Адреса
Position	text	-	-	Посада
User_id	varchar	-	+	Ідентифікатор користувача
Institution_useed_code	varchar	-	+	Код ЄДЕБО навчального закладу

У таблиці «Classes» (класи) зберігається інформація про клас, а саме назва класу та тип (клас старшої чи молодшої школи).

Таблиця 2.4

Опис таблиці Classes

Поле	Тип даних	ПК	ЗК	Опис
Id	varchar	+	-	Ідентифікатор
Name	varchar	-	-	Назва класу
Type	varchar	-	-	Тип класу
Teacher_id	varchar	-	+	Ідентифікатор вчителя (класного керівника)

Таблиця «Roles» (ролі). Сюди зберігаються назви ролей користувачів.

Таблиця 2.5

Опис таблиці Roles

Поле	Тип даних	ПК	ЗК	Опис
Id	int	+	-	Ідентифікатор
Role_name	varchar	-	-	Назва ролі

Таблиця «User_roles» (ролі корстувачів). Сюди зберігаються ролі різних користувачів.

Таблиця 2.6

Опис таблиці User_roles

Поле	Тип даних	ПК	ЗК	Опис
Id	int	+	-	Ідентифікатор
User_id	varchar	-	+	Ідентифікатор користувача
Role_id	int	-	+	Ідентифікатор ролі
Institution_u seed_code	varchar	-	+	Код ЄДЕБО навчального закладу

Таблиця «Schedule» (розклад) містить день тижня та до якого класу належатиме розклад.

Таблиця 2.7

Опис таблиці Schedule

Поле	Тип даних	ПК	ЗК	Опис
Id	int	+	-	Ідентифікатор
Class_id	varchar	-	+	Ідентифікатор класу

Weekday	int	-	-	День тижня

Таблиця «Lessons» (заняття) представляє певне заняття у розкладі. Вона містить час початку та закінчення уроку, ідентифікатор предмету, ідентифікатор вчителя, що проводитиме урок, тощо. Кожен урок посилається на розклад, в якому він знаходиться.

Таблиця 2.8

Опис таблиці Lessons

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Subject_id	int	-	+	Ідентифікатор предмету
Teacher_id	varchar	-	+	Ідентифікатор вчителя
Schedule_id	Int	-	+	Ідентифікатор розкладу
Start_time	time	-	-	Час початку уроку
End_time	Time	-	-	Час закінчення уроку

Таблиця «Subjects» (предмети) зберігає назву предмету та до яких класів він відноситься – старших чи молодших.

Таблиця 2.9

Опис таблиці Subjects

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Name	varchar	-	-	Назва предмету
Classes_type	varchar	-	-	Тип класу, для якого викладається предмет

Таблиця «Homework» (домашнє завдання) містить поля для збереження даних про задане домашнє завдання. Наприклад, до якого числа його потрібно здати, ким задане завдання, для його воно класу і так далі.

Таблиця 2.10

Опис таблиці Homework

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор

Subject_id	int	-	+	Ідентифікатор предмету
Teacher_id	varchar	-	+	Ідентифікатор вчителя
Class_id	varchar	-	+	Ідентифікатор класу
Description	text	-	-	Опис завдання
Assignment_ date	date	-	-	Дата, коли було задано завдання
Deadline	date	-	-	Дата, до якої потрібно виконати завдання

Таблиця «Academic_records» (оцінки) створена для збереження оцінок та відвідуваності учнів, а також того хто саме виставив оцінку чи “н-ку”.

Таблиця 2.11

Опис таблиці Academic_records

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Grade	int	-	-	Оцінка
Present	tinyint	-	-	Чи був учень присутнім на уроці
Student_id	varchar	-	+	Ідентифікатор учня
Teacher_id	varchar	-	+	Ідентифікатор вчителя
Class_id	varchar	-	+	Ідентифікатор класу
Subject_id	int	-	+	Ідентифікатор предмету
date	date	-	-	Дата занаття, за яке виставлено оцінку

Таблиця «Institutions» (навчальні заклади) створена для зберігання детальної інформації про навчальний заклад. Зокрема повну адресу, коди КОАТУУ та ЄДЕБО, повну та скорочену назву тощо.

Таблиця 2.12

Опис таблиці Institutions

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Fullname	mediumtext	-	-	Повна назва
Useed_code	varchar	-	-	Код ЄДЕБО
Shortname	varchar	-	-	Скорочена назва

State	varchar	-	-	Чи працює наразі заклад
Institution_type	varchar	-	-	Тип закладу
Ownership_form	varchar	-	-	Форма власності
Coatsuu_code	varchar	-	-	Код КОАТУУ
Region	varchar	-	-	Регіон (область)
Settlement	varchar	-	-	Населений пункт
Address	tinytext	-	-	Адреса
Governing_body_in_charge_of_education	mediumtext	-	-	Орган, до сфери управління якого належить заклад освіти
Phone_number	varchar	-	-	Номер телефону закладу
Email	varchar	-	-	Електронна пошта закладу
Website	varchar	-	-	Веб-сайт закладу
Admin_user_fullname	varchar	-	-	ПІБ адміністратора закладу
Admin_user_id	varchar	-	+	Ідентифікатор користувача, що є адміністратором закладу

Таблиця «Settlements» (населені пункти) потрібна для зберігання інформації про населені пункти, щоб генералізувати формат, в якому зберігаються населені пункти в інших таблицях, а також для пізнішого використання при реєстрації та інших діях.

Таблиця 2.13

Опис таблиці Settlements

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Region_coatsuu_code	mediumtext	-	-	Код КОАТУУ регіону
District_coatsuu_code	varchar	-	-	Код КОАТУУ району
Community_coatsuu_code	varchar	-	-	Код КОАТУУ громади
Settlement_coatsuu_code	varchar	-	-	Код КОАТУУ населеного пункту
Category	varchar	-	-	Категорія населеного пункту
Name	mediumtext	-	-	Назва населеного пункту
Coatsuu_code	varchar	-	-	Код КОАТУУ
Region	mediumtext	-	-	Регіон (область)

District	mediumtext	-	-	Район

Таблиця «Chat_rooms» (кімнати чатів) зберігатиме загальну інформацію про “кімнату”, яка в подальшому використовуватиметься для листувань між користувачами.

Таблиця 2.14

Опис таблиці Chat_rooms

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
Name	varchar	-	-	Назва чату
Type	varchar	-	-	Тип чату
Creation_timestamp	timestamp	-	-	Час створення чату у вигляді timestamp

Таблиця «User_chat_rooms» зберігає зв’язки між користувачами та чат-кімнатами, в яких вони знаходяться. Це допомагає реалізувати зв’язок “багато до багатьох”, адже користувач може бути у великій кількості різних чатів, так само як і в кожному чаті може знаходитись не один користувач.

Таблиця 2.15

Опис таблиці User_chat_rooms

Поле	**Тип даних**	**ПК**	**ЗК**	**Опис**
Id	int	+	-	Ідентифікатор
User_id	varchar	-	+	Ідентифікатор користувача
Room_id	varchar	-	+	Ідентифікатор кімнати
Joining_timestamp	timestamp	-	-	Час, коли користувач приєднався до кімнати у вигляді timestamp

Таблиця «Messages» (повідомлення) створена для збереження повідомлень користувачів. Вона має базові поля, що описують повідомлення: ідентифікатор відправника, текст повідомлення, час відправки тощо. Кожне повідомлення належить до певного чату, тому таблиця має посилання на “кімнату”.

Таблиця 2.16

Опис таблиці Messages

Поле	Тип даних	ПК	ЗК	Опис
Id	int	+	-	Ідентифікатор
Room_id	varchar	-	+	Ідентифікатор чату
Sender_id	varchar	-	+	Ідентифікатор відпарвника
Text	mediumtext	-	-	Текст повідомлення
Sending_datetimg	timestamp	-	-	Час відправлення повідомлення у вигляді timestamp

Отже, у створеній базі даних кожну таблицю було сконструйовано маючи на увазі структуру веб-платформи, її функціонал та потреби користувачів. Таким чином система буде надійнішою, а дані будуть зберігатись строго структуровано, що збереже від помилок як у розробці, так і при користуванні платформою. (рис. 2.2) [27].

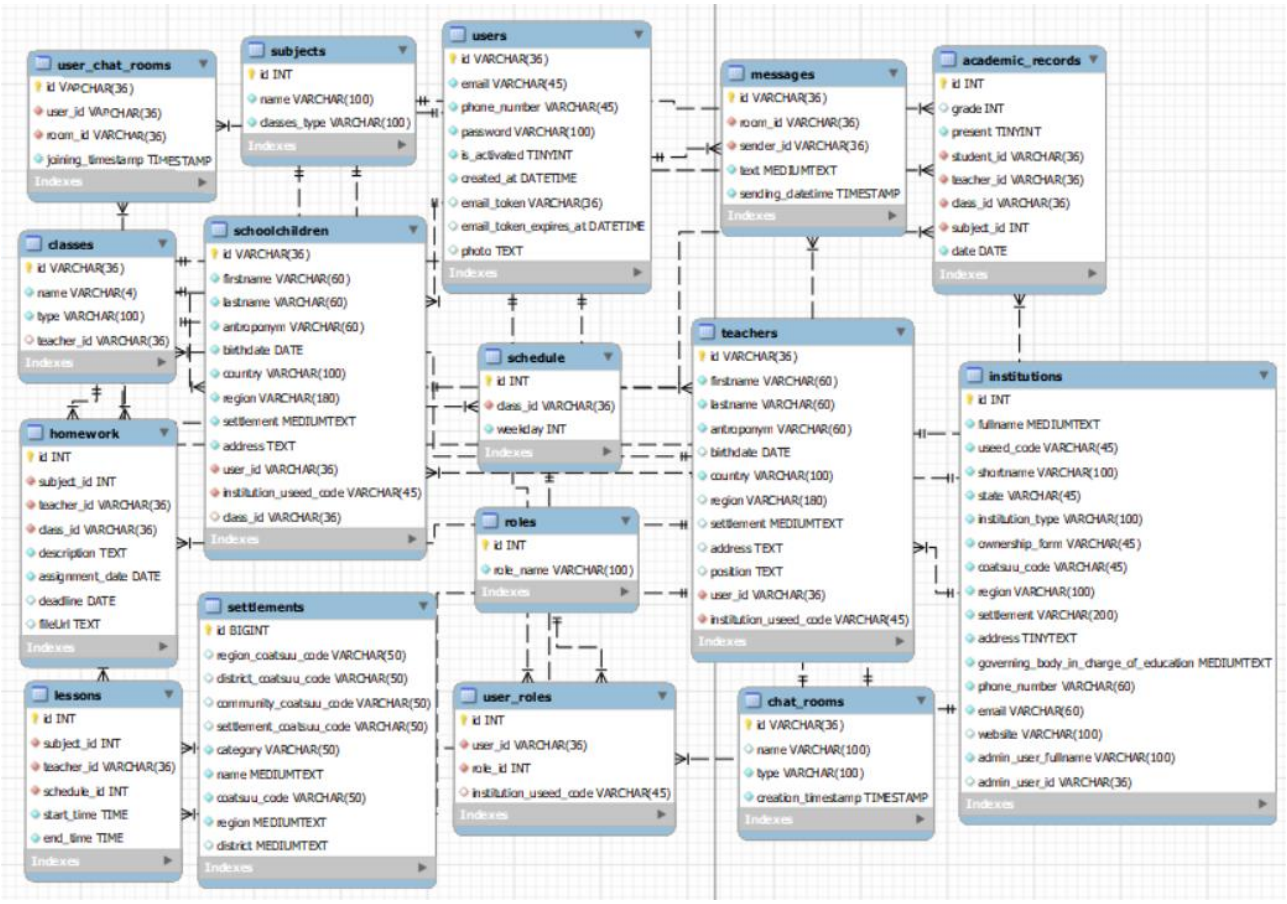


Рисунок 2.2 – Діаграма бази даних

2.3 Проектування та реалізація алгоритмів роботи системи

Потрапивши на сайт, авторизований користувач опиняється у власному профілі. Там він має можливість редагувати персональні дані, змінювати фото профілю чи пароль, Також користувач може користуватись навігаційною панеллю зверху екрану. В залежності від ролі користувача, доступні в навігаціній панелі сторінки будуть відрізнятись.

Головний адміністратор може редагувати, змінювати або видаляти дані користувачів та навчальних закладів, а також призначати навчальним закладам власних адміністраторів. Вони, у свою чергу, матимуть можливість створювати облікові записи учнів та вчителів, керувати їх даними, додавати та видаляти класи, переміщувати учнів з одного класу в інший, а також керувати розкладом та предметами, що вивчаються в певному навчальному закладі.

Права доступу вчителя є більш обмеженими, але він все ще може керувати власним обліковим записом, а також може додавати, видаляти та редагувати домашні завдання для різних класів, ставити оцінки чи відмічати присутність учнів. Також вчитель може переглядати як власний розклад, так і розклад будь-якого класу, може переглядати списки класів та листуватися з учнями або іншими вчителями.

Учень же має загалом права перегляду. Наприклад, перегляд домашніх завдань, своїх оцінок з усіх предметів, розкладу свого класу. Учень так само має можливість змінювати власну інформацію в профілі, а також пароль та фото профілю. Також учень може листуватись з іншими учнями та вчителями через чат.

Нижче на діаграмі описано варіанти використання онлайн-платформи дистанційного навчання в залежності від ролі користувача (рис. 2.3 - 2.6):

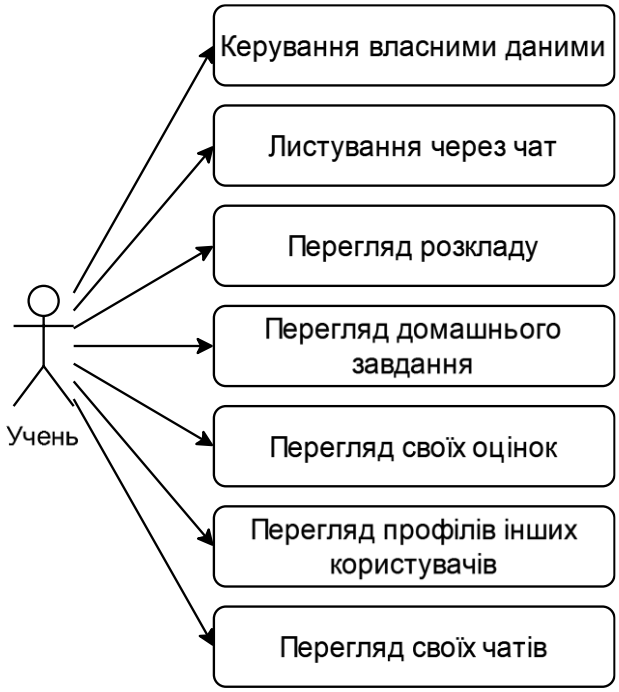


Рисунок 2.3 – Варіанти використання платформи учнем

Рисунок 2.4 – Варіанти використання платформи вчителем

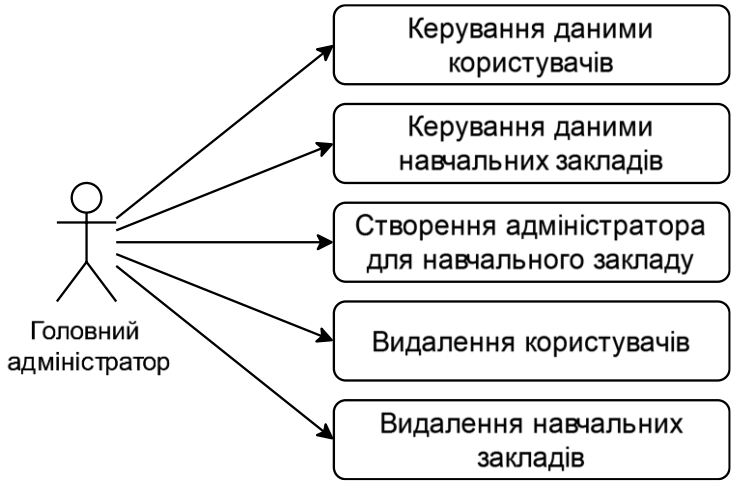


Рисунок 2.5 – Варіанти використання платформи головним адміністратором

Рисунок 2.6 – Варіанти використання платформи адміністратором навчального закладу

2.4 Реалізація онлайн-платформи дистанційного навчання

Для розробки платформи було обрано Next.js, завдяки чому можна чітко розмежувати клієнтську та серверну чатину додатку. Для створення клієнтської частини було використано бібліотеку React.js, яка імплементується фреймворком Next.js. Завдяки цьому додаток є динамічним, швидким та зручним у використанні. На кожну дію користувач отримує відгук або у вигляді повідомлення, або у вигляді оновлення певних даних.

Для розробки серверної логіки також використано фреймворк Next.js. Він використовує Node.js та бекенду, завдяки чому для запуску чатів було створено окремий сервер, а також реалізовано API для зв'язку з базою даних. Окрім вже названих технологій було також використано бібліотеку jose для роботи з JSON WebToken та Socket.io для уможливлення листування в реальному часі.

Можна перейти до розгляду основних частин веб-платформи та їх роботи.

Онлайн-платформа дистанційного навчання постійно оперує конфіденційними даними користувачів, такі як спрвжні ім'я та прізвище, адреса проживання, номер телефону та адреса електронної пошти. Тому реєстрацію нових користувачів було винесено в адміністративну панель .

Перш ніж дані користувача буде збережено в базі даних, його пароль хешується з допомогою бібліотеки bcrypt [16]. У разі успішного збереження даних для користувача генерується унікальний токен, який у вигляді посилання надсилається користувачеві на електронну пошту, щоб перевірити, чи вона є справжньою, та одночасно повідомити користувача, що його було додано до освітньої платформи.

Хешування паролю:

```
const password = await bcrypt.hash(body.password, 10);
```

Відправка токену на електронну пошту користувача:

```
const transporter = nodemailer.createTransport({
  host: "smtp.gmail.com",
  port: 587,
  secure: false,
  auth: {
    user: "ipz202_gaa@student.ztu.edu.ua",
    pass: "sutupziwvkvpwjqm ",
  },
```

```
});

export async function sendVerificationEmail(email, token) {
  const verificationUrl = `http://localhost:3000/auth/activate/${token}`;
  const html = `<!DOCTYPE html>
<html lang="en" xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:v="urn:schemas-microsoft-com:vml">
...інший html-код
</html>`;


  await transporter.sendMail({
    from: '"ConnectedMinds" <ipz202@student.ztu.edu.ua>',
    to: email,
    subject: "Підтвердіть свою пошту",
    html: html
  });
}
```

Для активації користувач натискає на кнопку в листі, який надійшов йому електронною поштою. Це переносить його на сторінку активації, де токен, що отримується з посилання, за яким перейшов користувач, порівнюється з токеном, збереженим в базі даних. При повному співпадінні, акаунт відмічається як активований.

При авторизації користувача створюється JSON WebToken [24], що зберігається в cookies. Він генерується на основі унікального ідентифікатора користувача:

```
export async function encrypt(payload) {
  return new SignJWT(payload)
    .setProtectedHeader({ alg: 'HS256' })
    .setIssuedAt()
    .setExpirationTime('7d')
    .sign(encodedKey)
}

export async function createSession(userId) {
    const expiresAt = new Date(Date.now() + 7 * 24 * 60 * 60 * 1000)
    const session = await encrypt({ userId, expiresAt })

    cookies().set('session', session, {
      httpOnly: true,
      secure: true,
      expires: expiresAt,
      sameSite: 'lax',
      path: '/',
    })
}
```

В подальшому згенерований токен буде використано для підтведження того, чи користувача авторизовано:

```
export const verifySession = async () => {
```

```
    if (cookies().get("session")?.value) {
        const cookie = cookies().get("session").value;
        const session = await decrypt(cookie);

        if (!session.userId) {
            redirect("/auth/sign-in");
        }


        return { isAuth: true, userId: session.userId };
    } else return null;
};
```

Ось приклад використання вищенаведеної функції для отримання інформації про авторизованого користувача, включаючи його роль:

```
export const getUser = async () => {
    const session = await verifySession();

    if (!session) return null;


    try {
        const result = await User.findById(session.userId);
        const user = result.data;


        return user;
    } catch (error) {
        console.error("Failed to fetch user");
        return null;
    }
};

export const getUserRole = async () => {
    const user = await getUser();
    if (user) {
        const userRoles = await Role.findByUserId(user.id);
        const role = await Role.findById(userRoles.data.role_id);
        return role.data.role_name;
    } else return null;
};
```

Як було сказано вище, спілкування користувачів даної платформи можливо завдяки окремому серверу на Node.js та використанні бібліотки Socket.io, яка і забезпечує миттєве відображення нових повідомлень.

Створення серверу для листування:

```
const httpServer = http.createServer();
const io = new Server(httpServer, {
    cors: {
        origin: "http://localhost:3000",
        methods: ["GET", "POST"],
        allowedHeaders: ["my-custom-header"],
        credentials: true,
    },
});
```

```
io.on("connection", (socket) => {
    console.info("A user connected:", socket.id);

    socket.on("join_room", (roomId) => {
        socket.join(roomId);
        console.info(`user with id-${socket.id} joined room - ${roomId}`);
    });

    socket.on("send_msg", (data) => {
        socket.to(data.room_id).emit("receive_msg", data);
    });

    socket.on("disconnect", () => {
        console.info("A user disconnected:", socket.id);
    });
});

const PORT = process.env.PORT || 3001;
httpServer.listen(PORT, () => {
    console.info(`Socket.io server is running on port ${PORT}`);
});
```

Проглянувши вищенаведений код можна прослідкувати, як реалізується листування. Спочатку створюється свервер. Після цього, коли сервер запущено, він починає реагувати на зміни. Наприклад, на приєднання користувача до “кімнати” (join_room), або на надсилання повідомлення (send_msg).

Ці методи для реагування на зміни також використовуються у клієнтській частині чату. Як приклад можна привести функцію, яка спрацьовує при натисканні на кнопку відправки повідомлення:

```
const sendData = async (e) => {
        e.preventDefault();
        if (currentMsg !== "") {
            const msgData = {
                room_id: room.room_id,
                sender_id: currentUser.id,
                text: currentMsg,
                sending_datetime: new Date().toISOString(),
            };
```

Повідомлення спочатку зберігається в стан, щоб користувач одразу помітив нове повідомлення в чаті. Реагування UI на зміну стану забезпечує вже згадана React.js [29].

```
            setMessages((prevMessages) => [...prevMessages, msgData]);
            const response = await fetch("/api/user/chats/messages", {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
```

```
                },
                body: JSON.stringify(msgData),
            });

            if (response.ok) {
                socket.emit("send_msg", msgData);
                setCurrentMsg("");
            } else {
                console.error("Failed to send message");
                setMessages((prevMessages) =>
                    prevMessages.filter((msg) => msg !== msgData)
                );
                setCurrentMsg("");
            }
        }
    };
```

Далі повідомлення зберігається в базі даних, щоб користувач міг пізніше повернутись в чат і, за потреби, переглянути його. В разі успішного збереження повідомлення в базі даних викликається send_msg.

Також окремо варто продемонструвати роботу електронного журналу. При плануванні розробки сторінки для оцінки успішності та відмічання відвідуваності учнів було враховано те, вчителі роками користувались класичним паперовим журналом, який мав стовпчик з іменами та прізвищами учнів, рядок з датами занять та клітинками для виставлення оцінок. Саме тому було вирішено повторити цей звичний формат, реалізувавши виставлення оцінок в інтерактивний електронний журнал.

Для створення нової оцінки або редагування існуючої варто просто клікнути в будь-яку клітинку. Завдяки цьому зміниться стан компонента, на який зреагує бібліотека React.js, і в клітинці з’явиться поле <input/> для вказання нової оцінки або ж виставлення “н-ки”:

```
const handleCellDoubleClick = (rowIndex, colIndex) => {
    if (!subject.length || subject.length === 0) {
        toast.error("Спочатку оберіть предмет");
        return;
    }
    setSelectedCell({ row: rowIndex, col: colIndex });
};
```

При зміни обраної клітинки спрацьовує хук useEffect(), який забезпечує ререндеринг компоненту у разі зміни стану.

```
useEffect(() => {
    document.body.addEventListener("keydown", handleKeyDown);
    document.body.addEventListener("focusout", handleCellBlur);
```

```
        return () => {
            document.body.removeEventListener("keydown", handleKeyDown);
            document.body.removeEventListener("focusout", handleCellBlur);
        };
    }, [selectedCell]);
```

Коли вчитель увів оцінку, він може натиснути клавішу “Enter” і таким чином виконати збереження оцінки:

```
const handleKeyDown = async (event) => {
    if (event.key === "Enter") {
        const { relatedTarget } = event;
        if (
            selectedCell &&
            (!relatedTarget || !tableRef.current.contains(relatedTarget))
        ) {
```

Щоб користувач не робив надто багато запитів на збереження одночасно, здійснюється перевірка, чи не зроблено вже запит.

```
if (!isRequestInProgress) {
    const subjectName = subject;
    const classesType = classObj.type;
    const date = `${
        selectedCell.col + 1 < 10
            ? "0" + (selectedCell.col + 1)
            : selectedCell.col + 1
    }.${month < 10 ? "0" + month : month}.`;
    setPrevSelectedCell((prev) => ({
        ...prev,
        row: selectedCell.row,
        col: selectedCell.col,
    }));
    setSelectedCell(null);
    const data = {
        grade: null,
        present: null,
        subjectName: subjectName,
        classesType: classesType,
        classId: classObj.id,
        studentId: schoolchildren[selectedCell.row].id,
        teacherUserId: user.id,
        date: date,
    };
```

Перед збереженням оцінки або “н-ки” в базу даних виконується перевірка, чи оцінка знаходиться в межах допустимих значень та чи є вона цифрою або буквою “н”. Якщо ж ні, користувача буде повідомлено про помилку.

```
if (
        parseInt(event.target.value) &&
        (parseInt(event.target.value) > 1 &&
            parseInt(event.target.value) < 12)
    ) {
        data.present = 1;
```

```
                                data.grade = parseInt(event.target.value);
                            } else if (event.target.value.toLowerCase() === "н") {
                                data.present = 0;
                                data.grade = null;
                            } else if (event.target.value === "") {
                                data.present = 1;
                                data.grade = null;
                            } else {
                                toast.error(
                                    'В клітинці може бути лише оцінка від 1 до 12,
"н-ка" або зовсім нічого'
                                );
                                return;
                            }
                            setIsRequestInProgress(true);
                            const response = await fetch(
                                `/api/institution/academic-records`,
                                {
                                    method: "POST",
                                    headers: { "Content-Type": "application/json" },
                                    body: JSON.stringify(data),
                                }
                            );
                            const result = await response.json();
                            if (!result.success) {
                                toast.error(result.data);
                                setIsRequestInProgress(false);
                            } else {
                                toast.success(result.data);
                                setIsRequestInProgress(false);
                                fetchRecords();
                            }
                        }
                    }
                }
            }
        };
```

В разі успішного збереження оцінки користувача також буде повідомлено. Окрім того, поки оцінка зберігається, буде показано спінер завантаження, щоб користувач бачив, що сайт працює і сторінка не “зависла” [18].

Висновки до другого розділу

Для того, щоб розробити якісну, зручну та надійну онайн-платформу дистанційного навчання, потрібно було виконати декілька важливих кроків. Спочатку потрібно було визначити, який вона матиме функціонал. Це робилося з допомогою розробки прецедентів використання платформи, і визначення системних, функціональних та нефункціональних вимог. Завдяки проведеній роботі було отримано достатньо інформації для проектування бази даних, в якій

кожна таблиця має своє призначення, та, за потреби, логічно пов’язана з іншими таблицями.

Після цього було створено онлайн-платформу на базі фреймворку Next.js, що використовує бібліотеки React.js та Node.js для клієнтської та серверної частин відповідно. Також було використано Socket.io для створення листування між користувачами в реальному часі.

Завдяки правильному підходу для розробки вдалося створити надійну та безпечну платформу для навчання онлайн.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-ПЛАТФОРМОЮ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1 Порядок встановлення та налаштування параметрів платформи

Щоб налаштувати платформу варто спочатку встановити потрібне програмне забезпечення.

А саме:

- Node.js (версія 18.18.0 або вище)
- MySQL (версія 8.3.0 або вище)

Для підключення до бази даних потрібно вказати необхідні параметри в .env.local:

- MYSQL_HOST: IP-адреса або доменне ім’я, з якого здійснюється підключення до бази даних.
- MYSQL_PORT: Порт, через який буде виконано підключення до бази даних.
- MYSQL_DATABASE: Назва бази даних.
- MYSQL_USER: Ім'я користувача до бази даних.
- MYSQL_PASSWORD: Пароль користувача бази даних.

3.2 Структура інтерфейсу онлайн-платформи дистанційного навчання

Перше, що бачить користувач, опинившись на сайті — це сторінка авторизації (рис. 3.1). Вона містить форму авторизації, а також напис “Забули пароль?”, на який користувач може натиснути, якщо бажає скинути свій пароль. Форма авторизації містить валідацію полів, а також перевірку пароля та існування користувача. Те саме стосується і форми для введення електронної пошти на сторінці відновлення пароля (рис. 3.2).

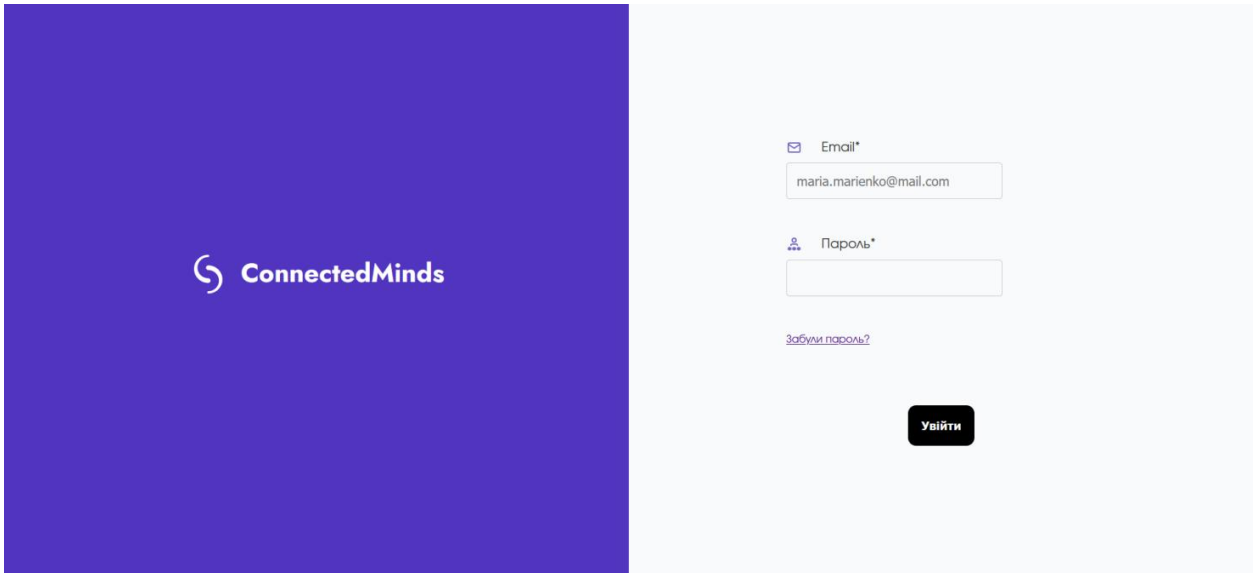


Рисунок 3.1 – Сторінка з авторизацією користувача

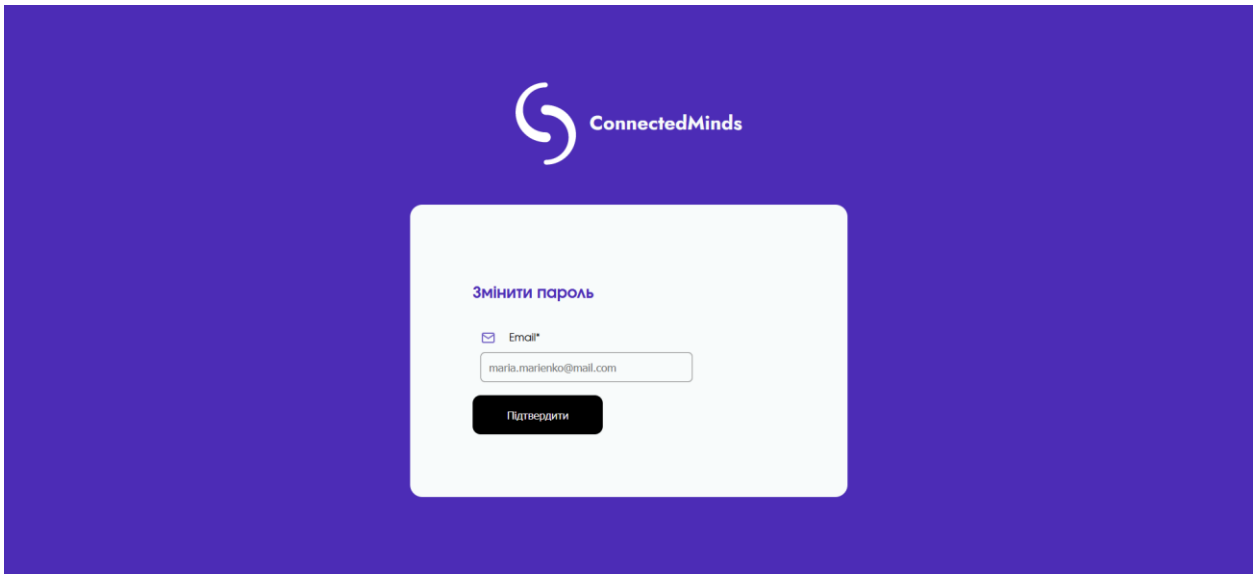


Рисунок 3.2 – Сторінка для відновлення забутого пароля

Далі все залежить від ролі користувача. Якщо це учень або вчитель, то після входу він опиниться в своєму особистому кабінеті.

Інформація в особистому кабінеті учня (рис. 3.3 – 3.4) частково представлена у вигляді форми. Так можна швидко і зручно оновити потрібні дані. Поряд з формою з особистими даними знаходиться форма зміни паролю. Для зміни паролю з профілю користувача не потрібно відправляти повідомлення з токеном, як це робиться при відновленні паролю неавторизованого користувача, адже достовірну інформацію користувача вже отримано.

Можна також змінити власне фото профілю. Саме фото завантажується у хмарне сховище Cloudinary [14], а посилання на нього зберігається в базу даних.

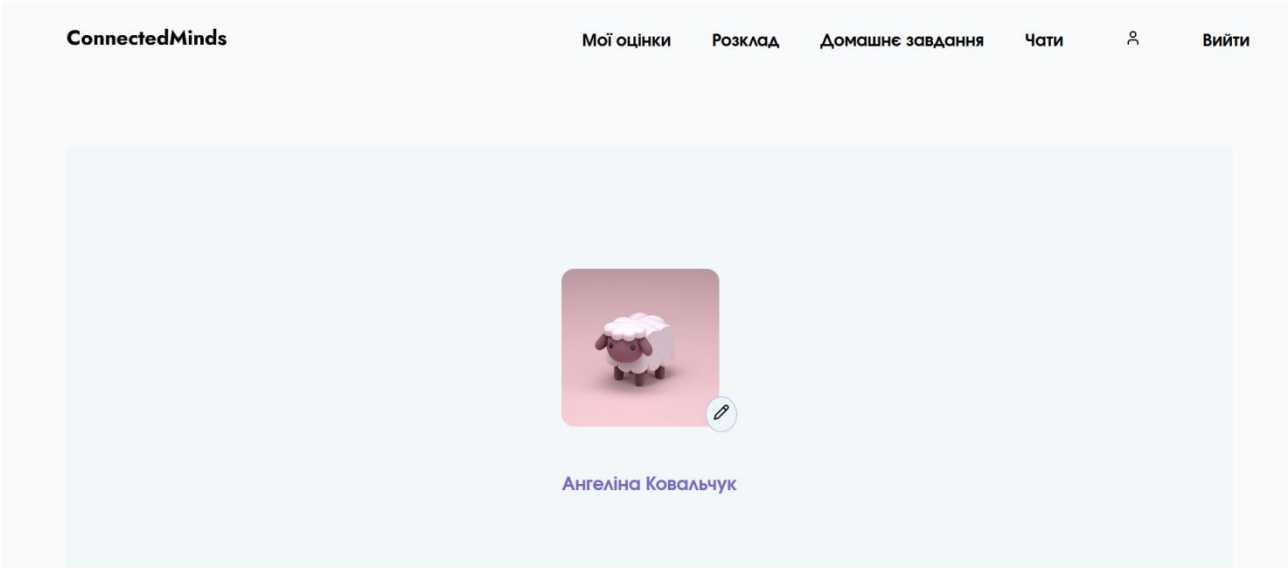


Рисунок 3.3 – Особистий кабінет учня

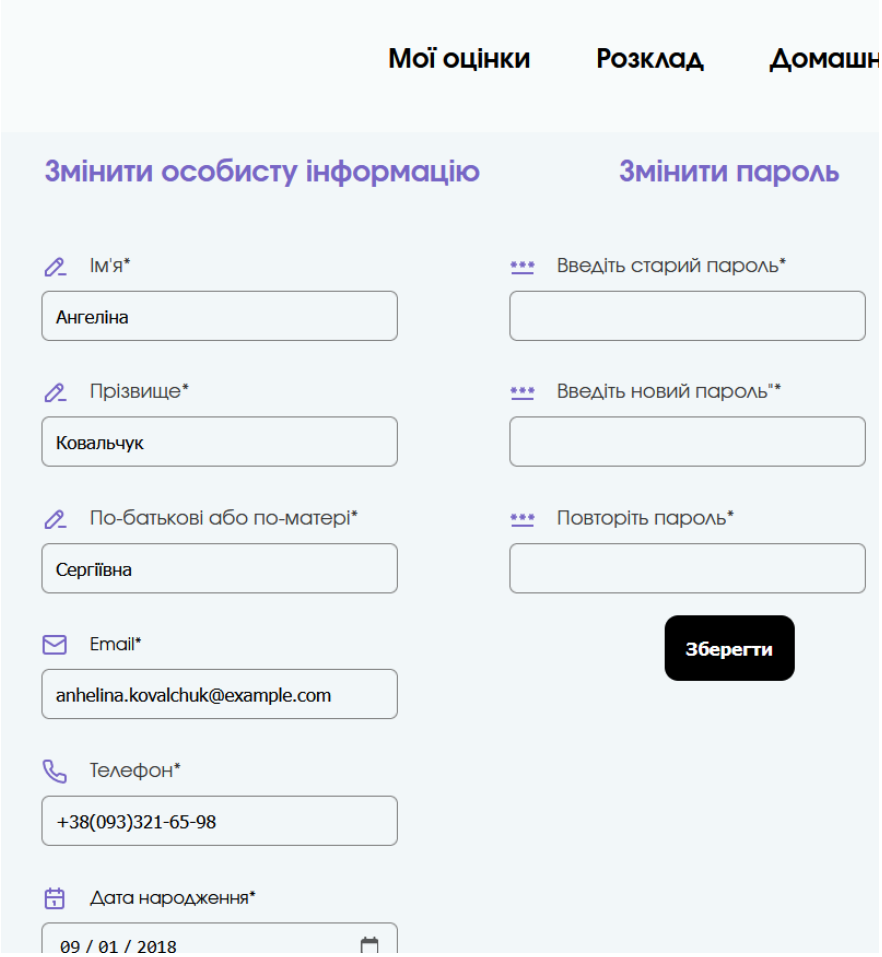


Рисунок 3.4 – Особистий кабінет учня

Особистий кабінет вчителя несильно відрізняється. Найголовніша відмінність — наявність додаткових кнопок під ім’ям та прізвищем (рис. 3.5), адже вчитель має доступ до більшої кількості сторінок. Перша кнопка перенаправляє вчителя на сторінку з класами, а друга — на сторінку з розкладом, де можна самостійно обрати клас, розклад якого хочеться переглянути [12].

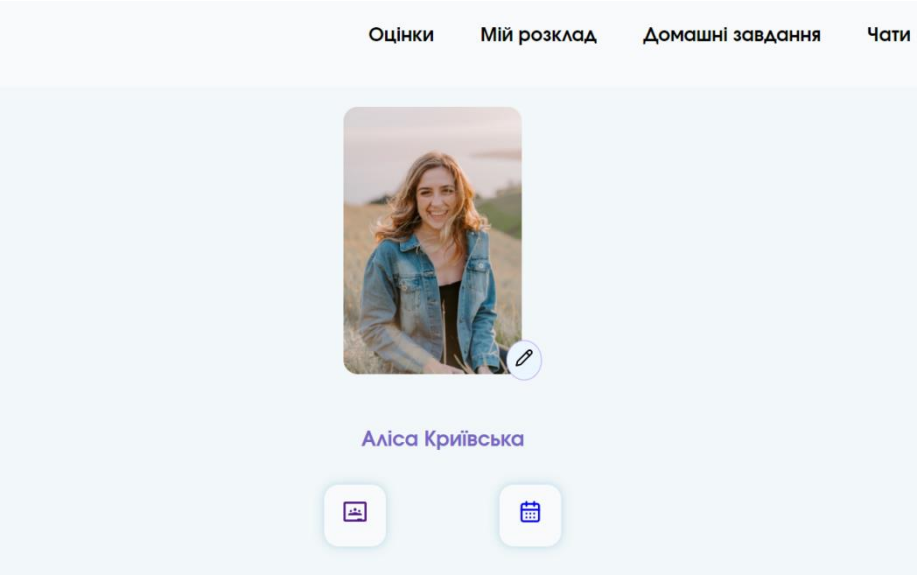
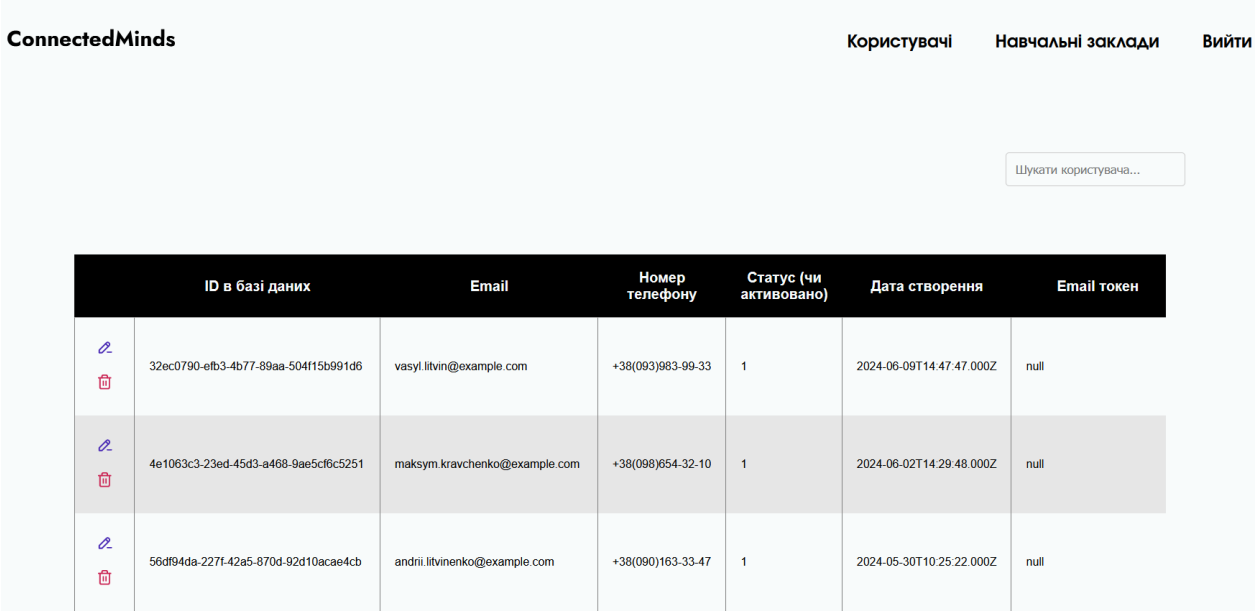


Рисунок 3.5 – Додаткові кнопки в особистому кабінеті вчителя

Якщо авторизований користувач є головним адміністратором, він опиниться на сторінці з даними користувачів (рис. 3.6 – 3.7).



	ID в базі даних	Email	Номер телефону	Статус (чи активовано)	Дата створення	Email токен
	32ec0790-efb3-4b77-89aa-504f15b991d6	vasyl.litvin@example.com	+38(093)983-99-33	1	2024-06-09T14:47:47.000Z	null
	4e1063c3-23ed-45d3-a468-9ae5cf6c5251	maksym.kravchenko@example.com	+38(098)654-32-10	1	2024-06-02T14:29:48.000Z	null
	56df94da-227f-42a5-870d-92d10acae4cb	andrii.litvinenko@example.com	+38(090)163-33-47	1	2024-05-30T10:25:22.000Z	null

Рисунок 3.6 – Сторінка з даними користувачів у адмін-панелі

	6e24247d-88c3-4587-931f-2e8dd6dbb587	volodymyr.sydorenko@example.com	+38(096)144-34-62	1	2024-06-08T22:30:55.000Z	null
	71d91f9d-1456-4d09-af1c-19240959c0bc	iryna.siryk@example.com	+38(099)163-46-77	1	2024-06-02T07:45:32.000Z	null
	73f6e336-cc63-4e71-9dff-85087c878d92	stanislav_boyovyi25@mail.com	+380(97)255-54-32	1	2024-06-07T23:45:38.000Z	null
	8ac6dd56-1b64-42c0-af73-bfc66d1d7c8f	oleksandr.tkachenko@example.com	+38(097)613-75-28	1	2024-06-05T11:20:48.000Z	null
	9a97161d-3b53-47a7-b67f-12cbbda1c5e4	mariia.siryk@example.com	+38(093)134-24-45	1	2024-05-28T00:55:17.000Z	null

1 2 3

Рисунок 3.7 – Сторінка з даними користувачів у адмін-панелі

Для зручності користування адмін-панеллю було додано пагінацію та пошук [10]. Поля таблиці можна редагувати за деякими обмеженнями. Наприклад, не можна відредагувати дату створення профілю користувача або унікальний ідентифікатор навчального закладу в базі даних. За виключенням таких полів записи в таблиці можна редагувати та видаляти. При натисненні на іконку-олівець в таблиці з’являться поля вводу для доступних до редагування полів (рис. 3.8).

Призначити адміністратора

Шукати заклад...

	ID в базі даних	Повна назва	Код ЄДЕБО	Коротка назва	Стан	Тип закладу	Форма власності	Код КОАТУУ	Область	Населений пункт	Адреса
	37719	Гімназія №150 міста Києва	148431	Гімназія № 150	працює	гімназія	Комунальна	8036100000	Київська область	Київ	вулиця Пирогівськ шлях, 148

Рисунок 3.8 – Редагування запису на сторінці з навчальними закладами

Сторінка з навчальними закладами та інші таблиці адмін-панелі головного адміністратора та адміністратора навчального закладу виглядають та працюють аналогічно завдяки повторному використанні компонентів інтерфейсу.

З адмін-панелі можна редагувати та видаляти дані. Якщо це сторінка з навчальними закладами, то є ще можливість додати адміністратора закладу (рис. 3.9).

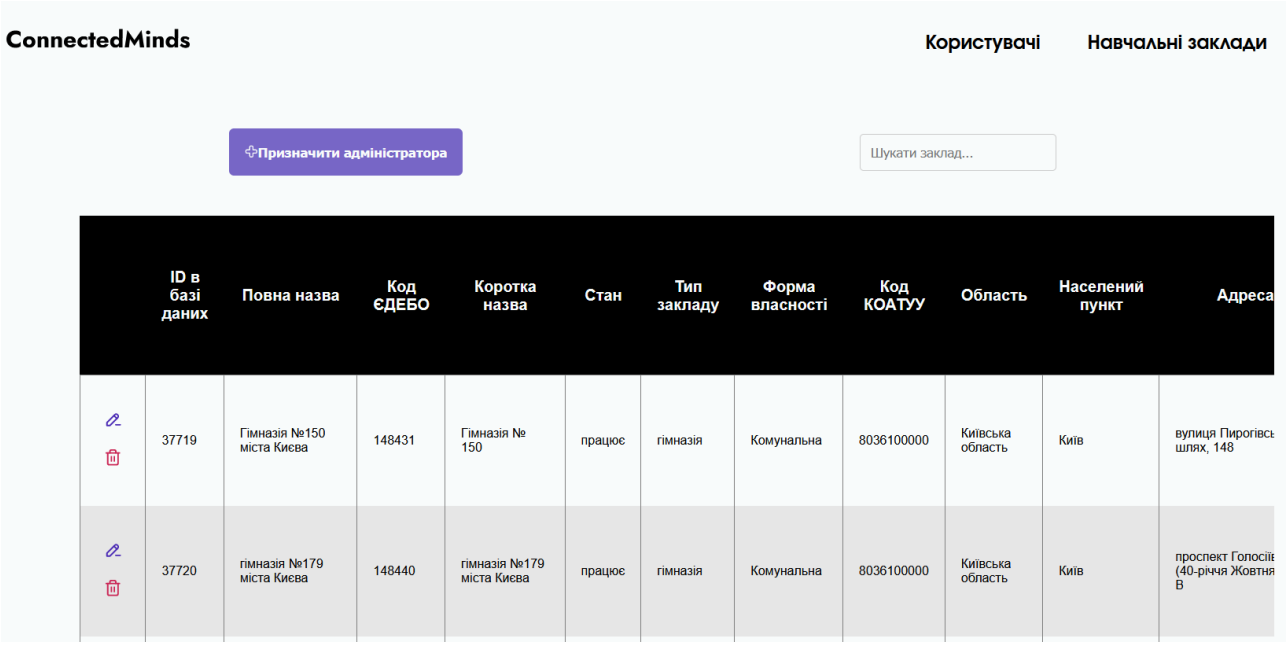


Рисунок 3.9 – Сторінка з навчальними закладами у адмін-панелі

При натисканні на кнопку “Призначити адміністратора” відкриється модальне вікно, куди потрібно внести усі дані майбутнього адміністратора (рис. 3.10).

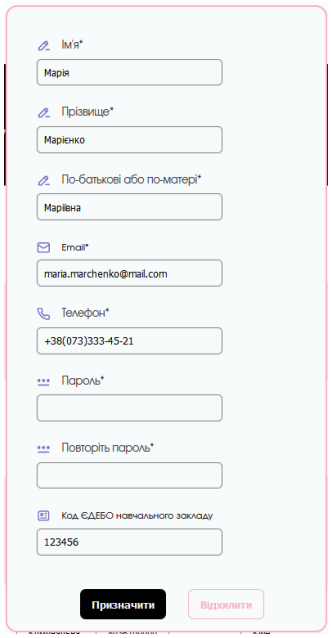


Рисунок 3.10 – Модальне вікно для додавання адміністратора навчального закладу

Після призначення користувачеві, якого було призначено адміністратором, на пошту надійде лист з проханням активувати акаунт (рис. 3.11) [11].

Модальне вікно для додавання учня практично аналогічне. Але воно також містить випадаючий список областей та введення населених пунктів з автодоповненням, щоб пришвидшити реєстрацію користувача та уникнути помилкового введення даних [19].

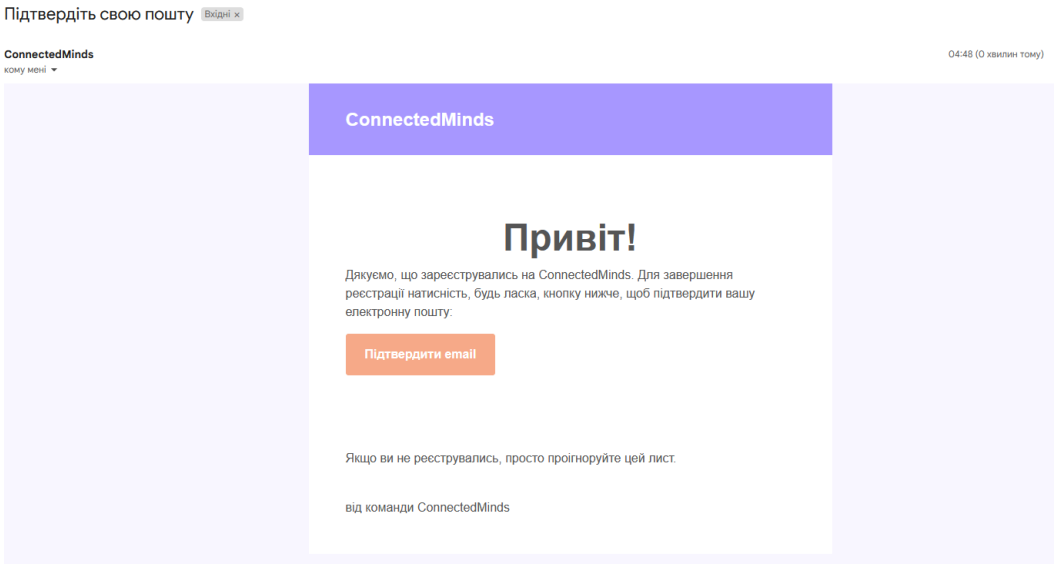


Рисунок 3.11 – Сторінка з листом з електронної пошти новопризначеного адміністратора

При натисканні на кнопку відбудеться перехід на сторінку активації, де користувача повідомлять, чи активація пройшла успішно (рис. 3.12).

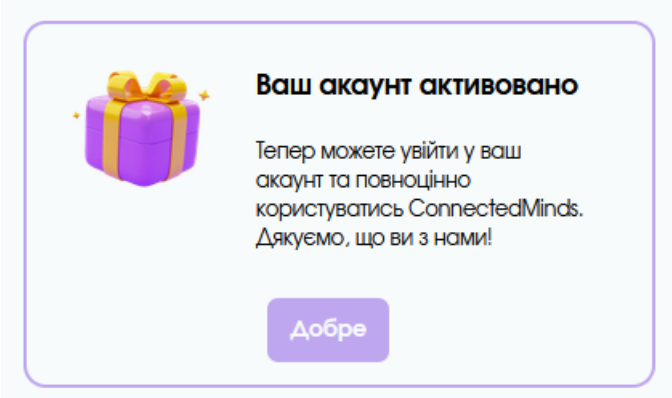


Рисунок 3.12 – Повідомлення про успіх на сторінці активації акаунту

Тепер можна увійти з акаунту адміністратора навчального закладу (рис. 3.13). Після авторизації користувач опиняється на сторінці з класами. На ній міститься інформація про всі класи школи, хто в них навчається та хто є класним

керівником. При натисканні на плюсик або поряд з областю з кружечками фотографій учнів відкриється повний список учнів класу (рис. 3.14) [23]:

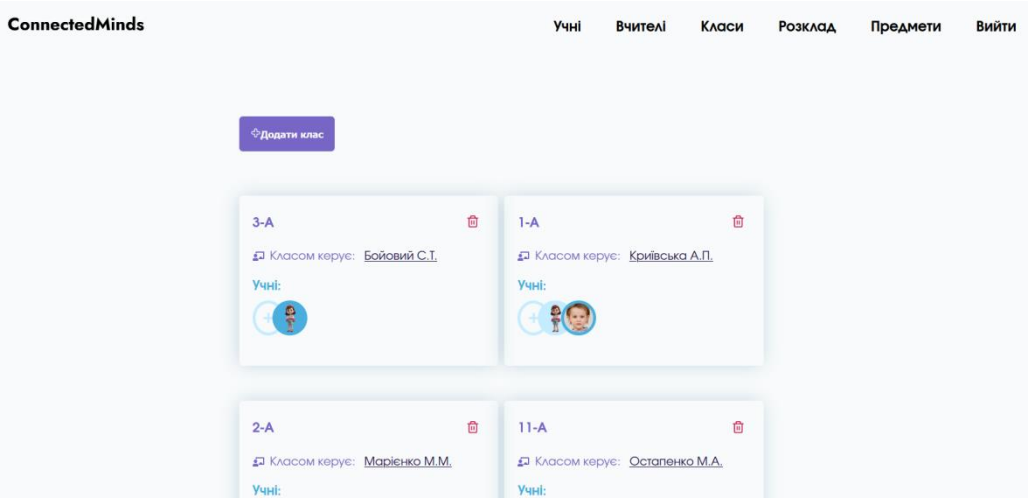


Рисунок 3.13 – Сторінка з класами

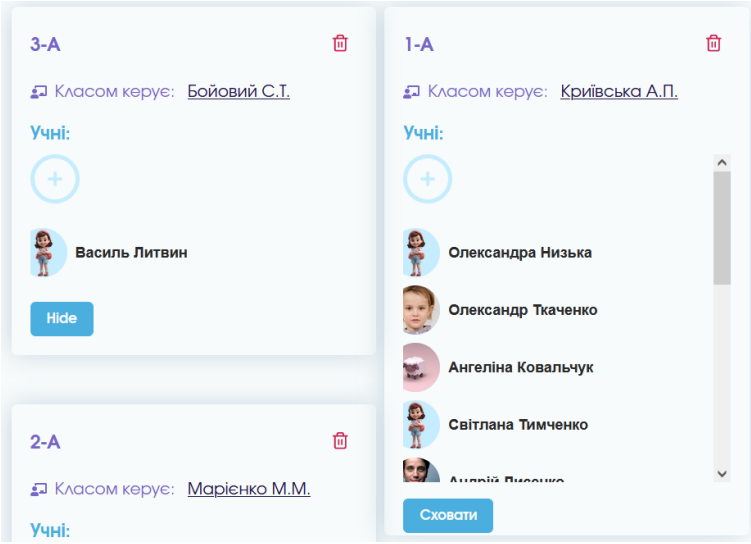


Рисунок 3.14 – Повний список учнів класу

У розгорнутому вигляді можна переводити учнів з класу в клас шляхом перетягання (drag and drop). Це дозволить швидко та зручно змінити клас певному учню [12].

При натисканні на прізвище та ініціали класного керівника або на прізвища та ім’я учня буде здійснено перехід на профіль обраного користувача.

Перейдемо на сторінку керування розкладом (рис. 3.15). Дана сторінка надає можливості для редагування, додавання та видалення предметів у розкладі. За замовчуванням кожен клас має шість пустих робочих днів, у які адміністратор навчального закладу може, за потреби, додати новий урок. При натисканні на

іконку-олівець поля з даними у блоці дня буде замінено на елементи форми й можна буде відредагувати будь-який предмет у розкладі (рис. 3.16) [20].

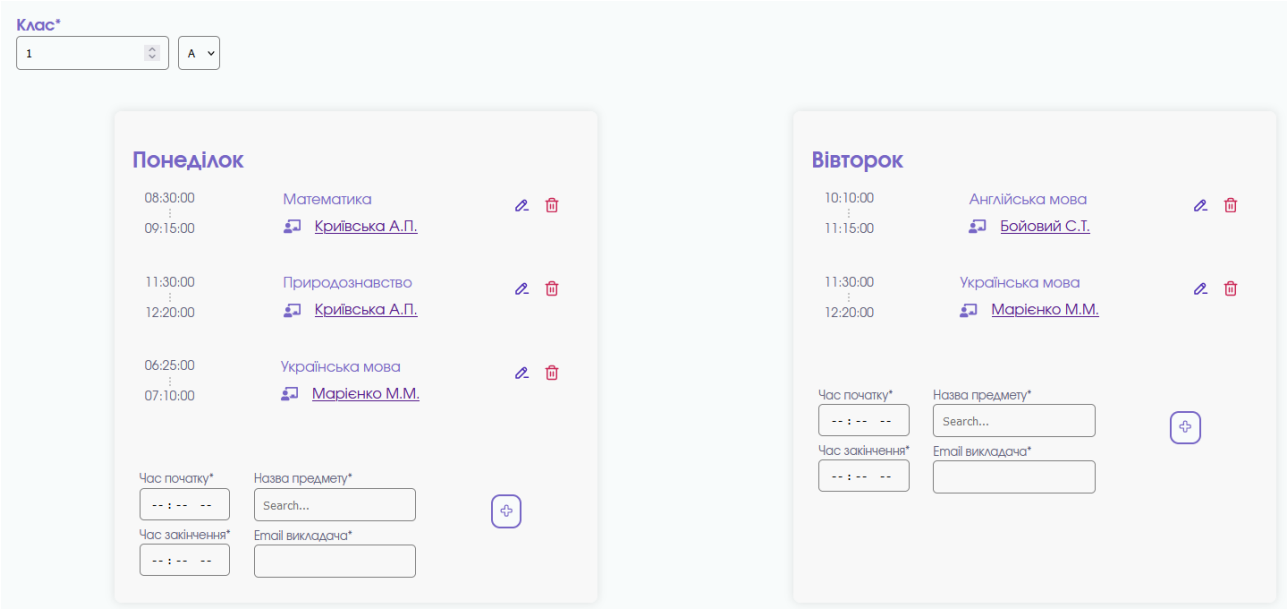


Рисунок 3.15 – Керування розкладом

Щоб вказати вчителя, який буде вести урок, потрібно вписати його електронну пошту. Якщо адміністратор готовий зберегти даний урок, він натискає на “галочку”, якщо він хоче все відмінити – то на хрестик (рис. 3.15) [24].

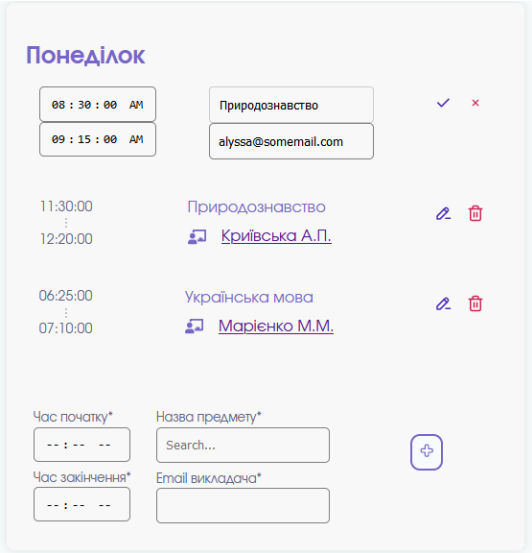


Рисунок 3.16 – Редагування предмету у розкладі

Предмет можна обирати з випадаючого списку, який з’являється при вводі кількох перших літер назви. Так користувачі уникнуть помилок при внесенні уроку з певного предмету до розкладу [13].

Для демонстрації інтерфейсу та роботи чатів потрібно зайти з профілю учня або вчителя. Після цього перейдемо на вкладку “Чати” у навігаційній панелі. Буде здійснено перехід на сторінку з чатами поточного користувача (рис. 3.17). Якщо даний користувач ще ні з ким не листувався, можна знайти потрібного користувача, скориставшись пошуком, та перейти на його профіль, щоб звідти написати йому (рис. 3.18) [17].

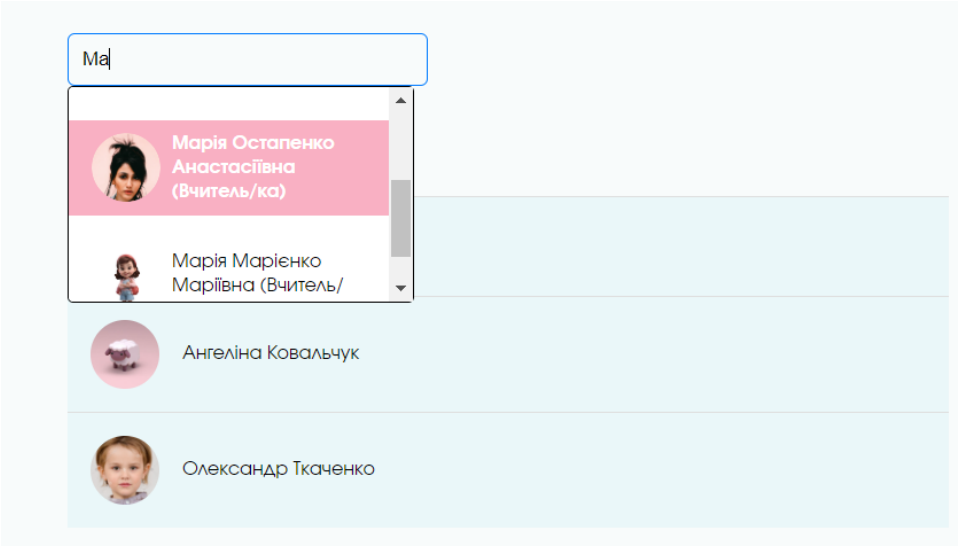


Рисунок 3.17 – Чати поточного користувача та пошук користувача

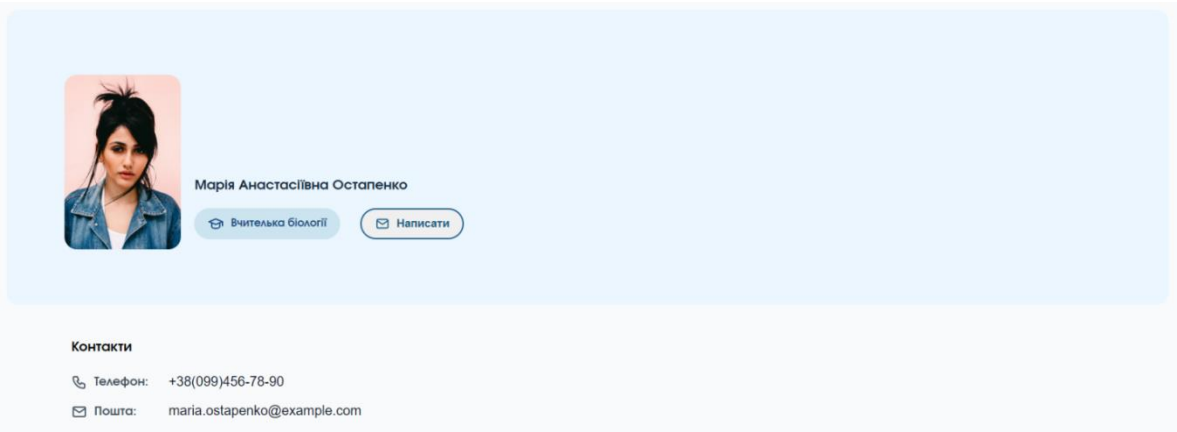


Рисунок 3.18 – Профіль вчителя

При натисканні на кнопку “написати” буде здійснено перехід у чат з даним користувачем (рис. 3.19). Зверху буде відображено ім’я та прізвище користувача, посередині – повідомлення, а знизу буде панель для введення та відправки повідомлення.

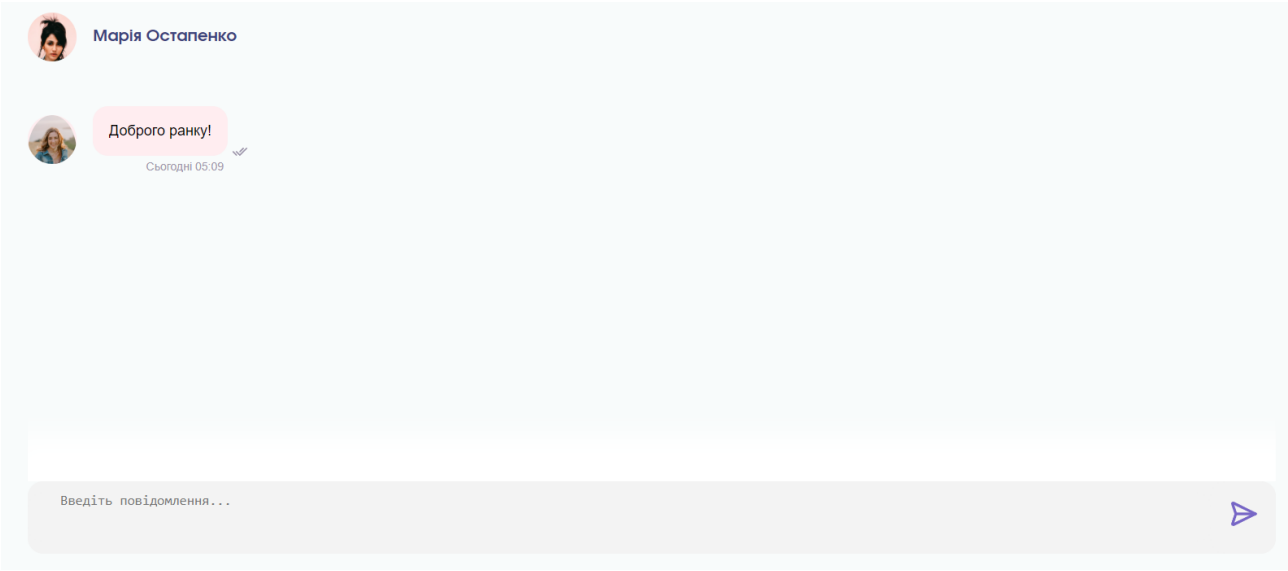


Рисунок 3.19 – Листування через чат

Варто розглянути також сторінку з оцінками та відвідуваністю учнів (рис. 3.20). Вона зроблена схожою на традиційний паперовий журнал, щоб користування даною сторінкою було інтуїтивно зрозумілим [15].

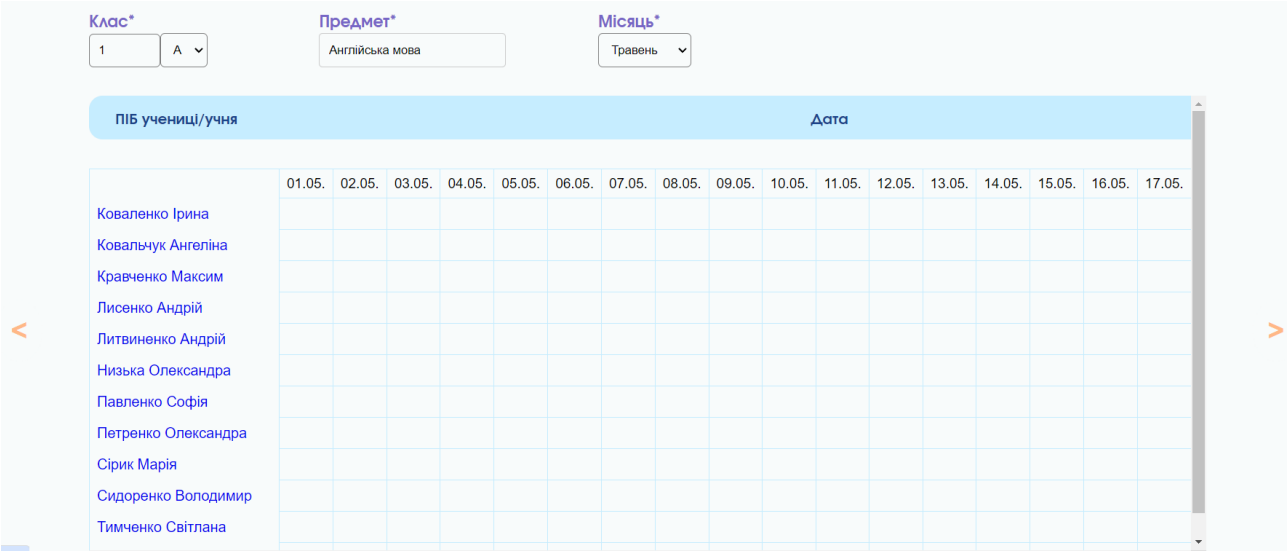


Рисунок 3.20 – Сторінка для відстеження успішності та присутності учнів

На даній сторінці можна переглядати, виставляти та редагувати оцінки будь-якого класу з будь-якого предмету та за будь-який місяць [21]. При кліканні двічі на одну з клітинок (навіть ту, де вже стоїть оцінка або “н-ка”), у клітинці з’являється поле вводу (рис. 3.21). Зберегти введену в нього оцінку можна натиснувши клавішу “Enter”, а скасувати зміни можна просто клікнувши де-інде.

Натиснувши на прізвище та ім’я учня можна перейти на його профіль. А якщо навести курсор на клітинку з “н-кою” чи оцінкою, можна побачити, хто виставив дану оцінку або “н-ку” та перейти на профіль вчителя (рис. 3.22).

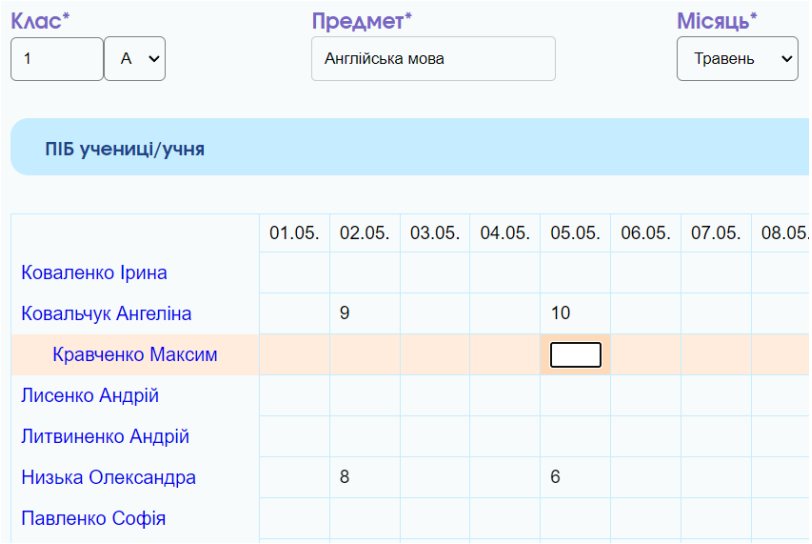


Рисунок 3.21 – Відкриття поля для введення оцінки

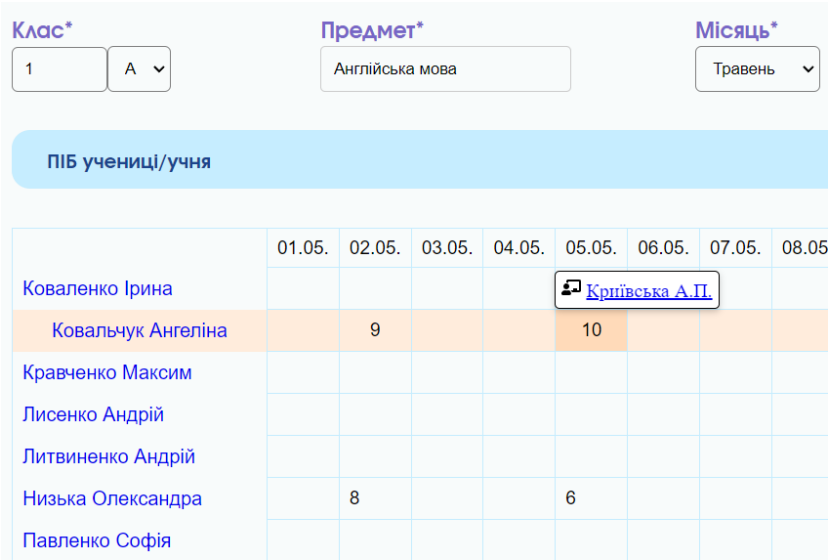


Рисунок 3.22 – Спливаюче віконечко з посиланням на вчителя, що виставляв оцінку або “н-ку”

Перейдемо до перегляду та додавання домашніх завдань. Додати нове завдання можна клікнувши на кнопку “Додати домашнє завдання”. Після цього відкриється модальне вікно, де можна буде внести всі необхідні дані про завдання додому та зберегти його (рис. 3.23). Поле введення предмету представлене у вигляді інтерактивного поля вводу з автодоповненням [25].

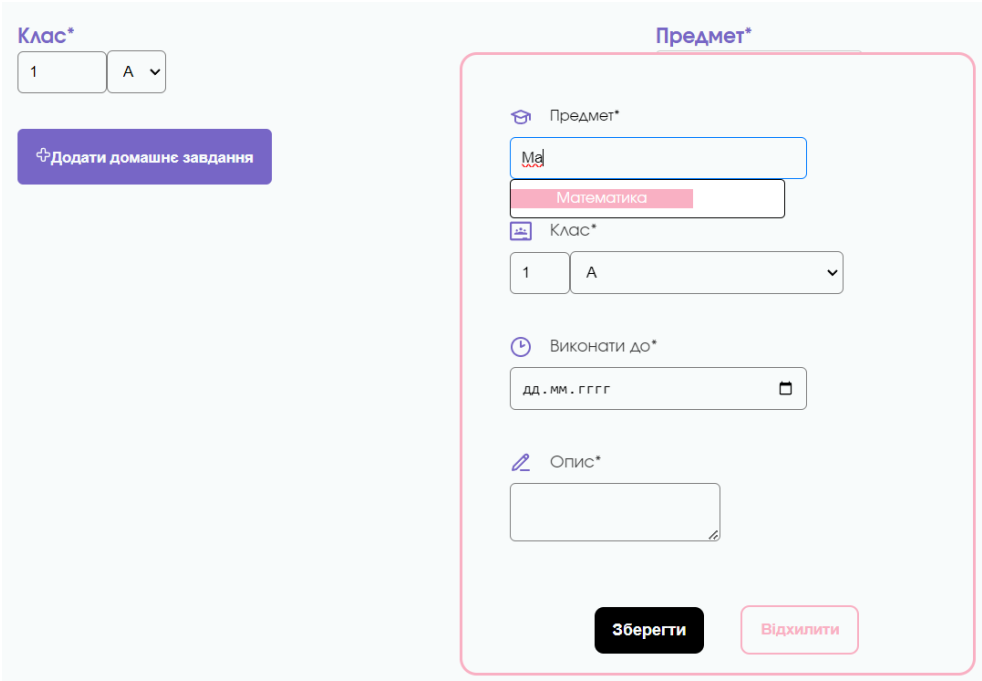


Рисунок 3.23 – Модальне вікно для додавання нового домашнього завдання

Після натиснення кнопки “Зберегти” додане завдання одразу з’являється на сторінці (рис. 3.24).

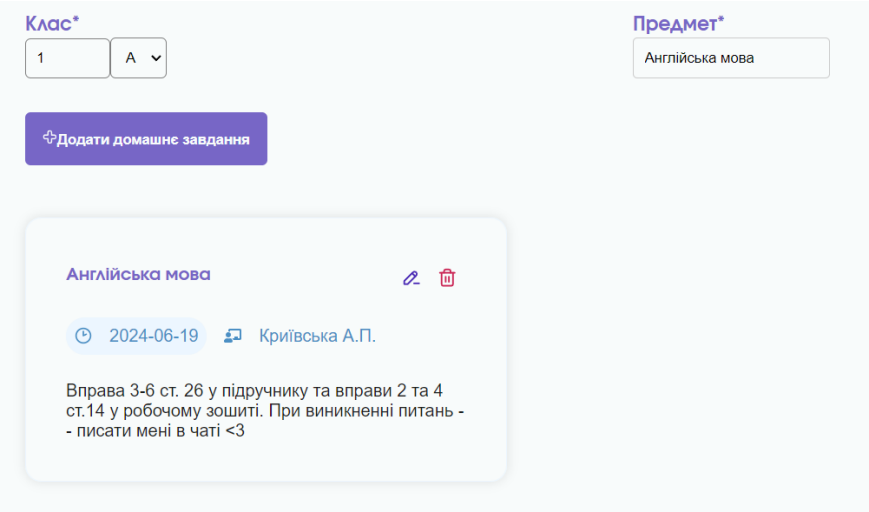


Рисунок 3.24 – Домашнє завдання

Учень також має доступ до сторінки з розкладом, оцінками та домашнім завданням. Проте для нього редагування, видалення та інші операції, доступні вчителю, невидимі та заборонені. Також є відмінність у відображенні оцінок зі сторони вчителя та зі сторони учня. Вчитель в першому стовпчику бачить імена всіх дітей у класі та їх оцінки. Для учня ж в лівому стовпчику було розміщено

назви предметів, а навпроти них вже показуються оцінки, якщо вони були виставлені.

3.3 Тестування платформи

Тестувати програмне забезпечення можна як вручну, так і за допомогою сторонніх програм, сайтів або написаних власноруч тестів. В моєму випадку було прийнято рішення здійснювати тестування вручну. Воно дозволить якісно перевірити валідацію полів у формах, протестувати граничні значення де це потрібно, та перевірити роботу платформи у випадку помилок чи виключних ситуацій.

Для початку було вирішено протестувати сторінку авторизації. Вона оснащена валідацією, щоб контролювати правильність введення даних користувачем (рис. 3.25). Також під час спроби авторизації виконується перевірка, чи користувач з такою поштою взагалі існує, і якщо так, чи правильно введено пароль (рис. 3.26 – 3.27) [26].

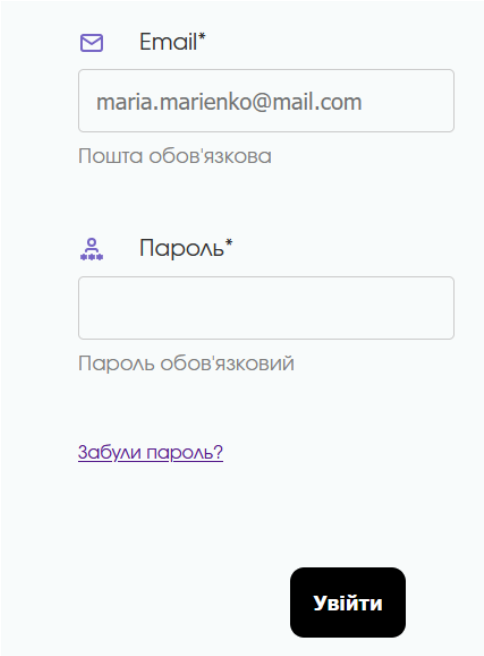


Рисунок 3.25 – Перевірка спроби увійти з пустими полями

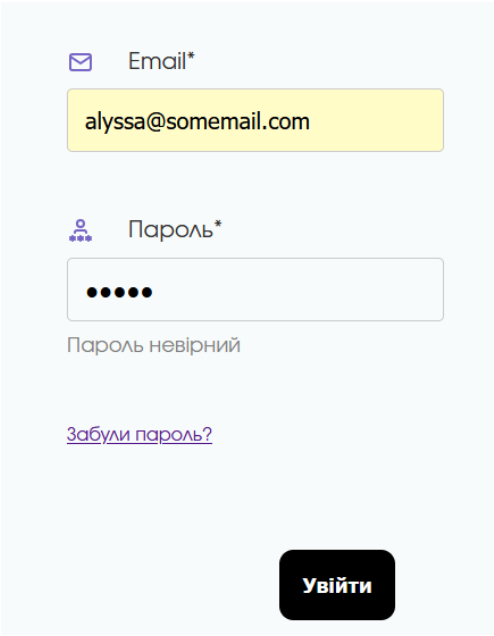


Рисунок 3.26 – Перевірка спроби увійти із неправильно вказаним паролем

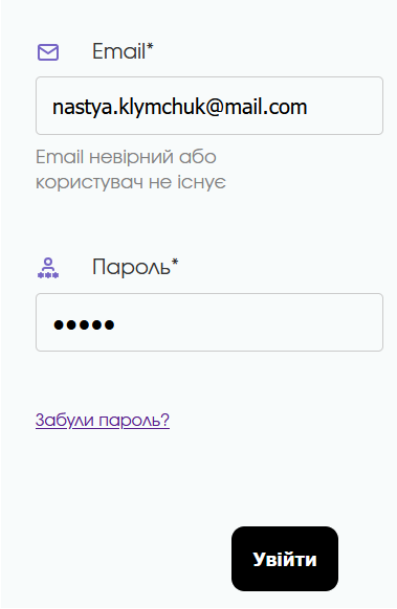


Рисунок 3.27 – Перевірка спроби увійти з використанням вигаданої пошти

Варто також протестувати виставлення оцінки та перевірити, чи все працює коректно та чи працює попередження про неправильно введені дані. В даному випадку корисно провести тестування граничних значень [27]. Оцінка не може бути меншою за 1 або більшою за 12, але що буде, якщо, наприклад, ввести оцінку рівно 1 (рис. 3.28)?

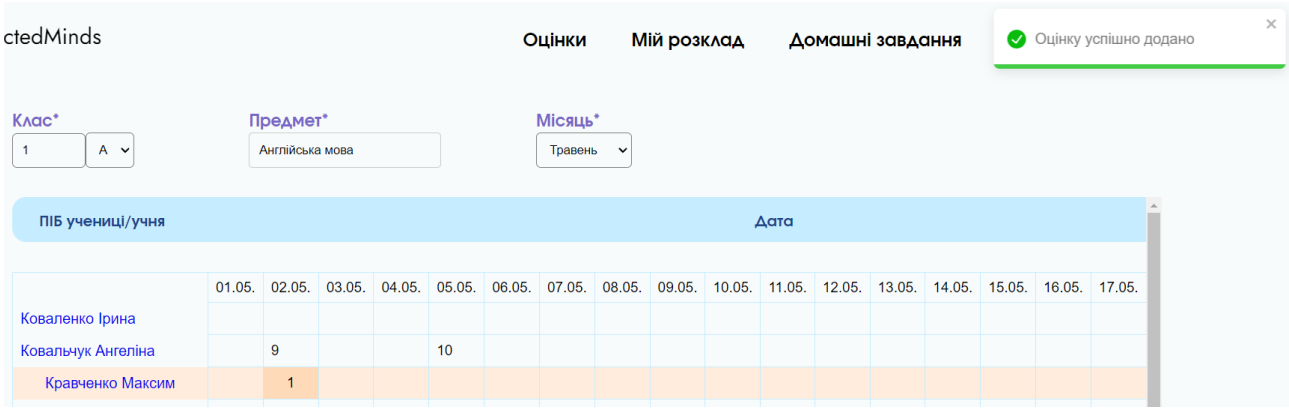


Рисунок 3.28 – Додавання оцінки, яка знаходиться на границі дозволених оцінок

Тестування на ввдення граничного значення успішно пройдене. Тепер варто перевірити введення неправильних даних. Наприклад, послідовності літер “abc” (рис. 3.29).

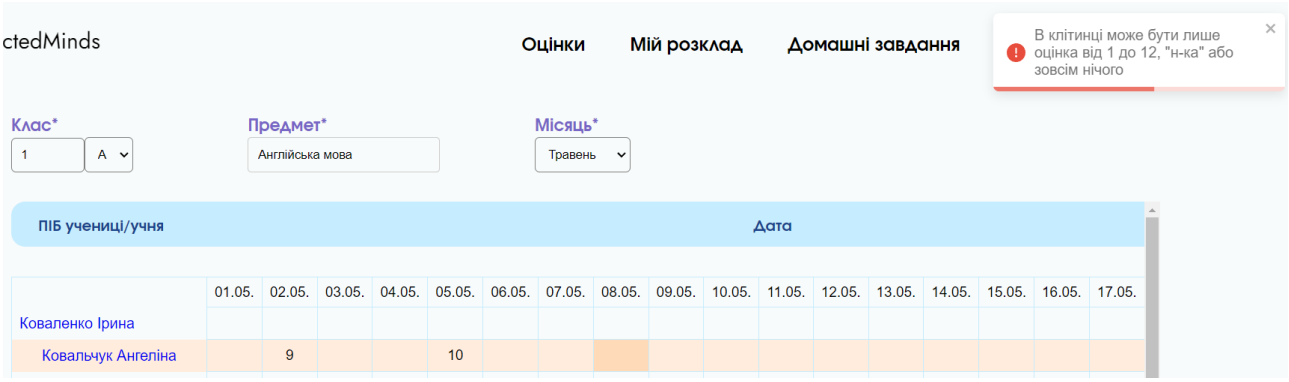


Рисунок 3.29 – Додавання некоректних даних в замість оцінки

Бачимо, що контроль правильності значень працює і в разі введення неправильних даних користувач одразу отримує сповіщення.

Тепер спробуємо додати в розклад занять урок з пустими даними (рис. 3.30).

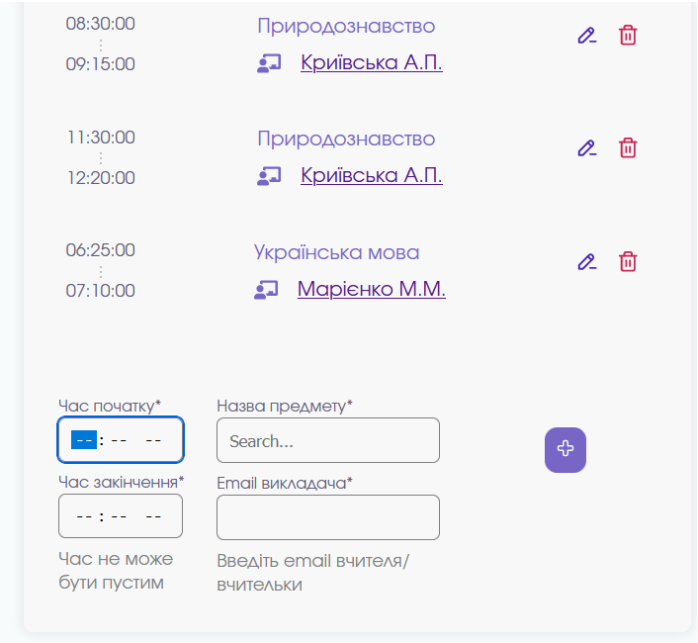


Рисунок 3.30 – Додавання нового уроку з пустими даними

Бачимо, що програма не пропускає пустий урок. Перевіримо чи можна зберегти відредагований урок з пустими або некоректними даними (рис. 3.31).

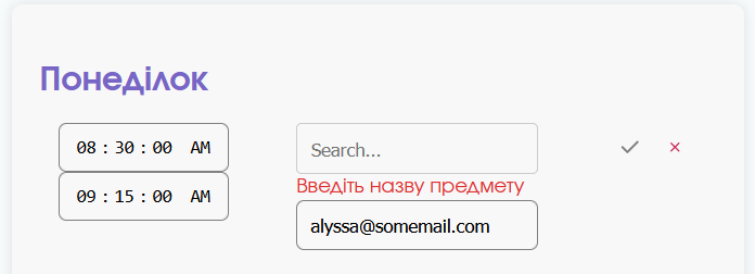


Рисунок 3.31 – Збереження відредагованого уроку з пустими або некоректними даними

Бачимо, що якщо присутні помилки в заповненні форми, урок не буде збережено в розкладі. Спробуємо тепер видалити урок української мови (рис. 3.32).

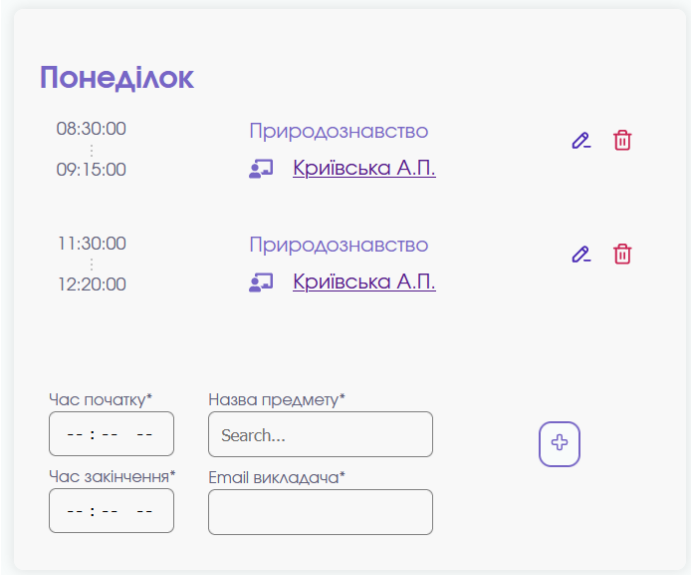


Рисунок 3.32 – Видалення уроку

Бачимо, що урок було успішно видалено.

Можна також виконати перевірку зміни паролю через власний профіль вчителя. Якщо старий пароль введено неправильно, програма не має змінити пароль в базі даних. Тестування пройшло успішно, користувача було повідомлено про те, що пароль введено невірно (рис. 3.33).

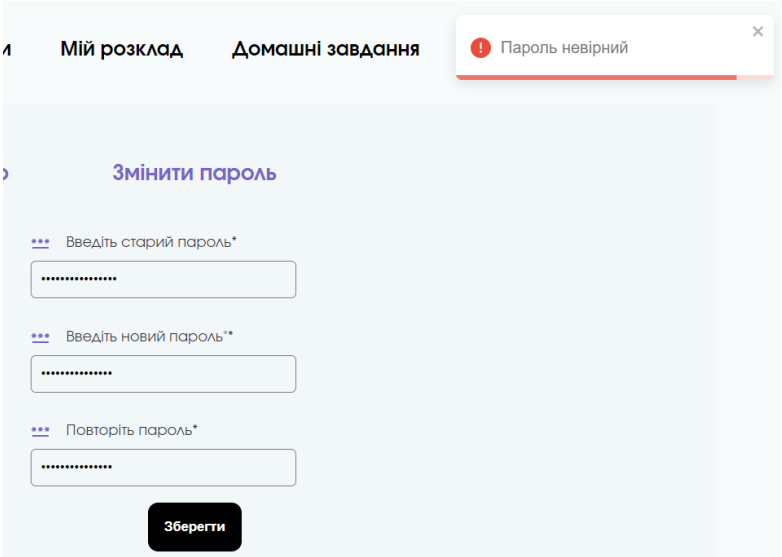


Рисунок 3.33 – Зміна пароля користувача

Наостанок можна протестувати редагування та видалення певних даних з адмін-панелі. Оскільки всі таблиці в адмін-панелі реалізовані аналогічно, було вирішено здійснити тестування авторизувавшись як головний адміністратор та

обрати запис для редагування з таблиці на сторінці навчальних закладів (рис. 3.34).

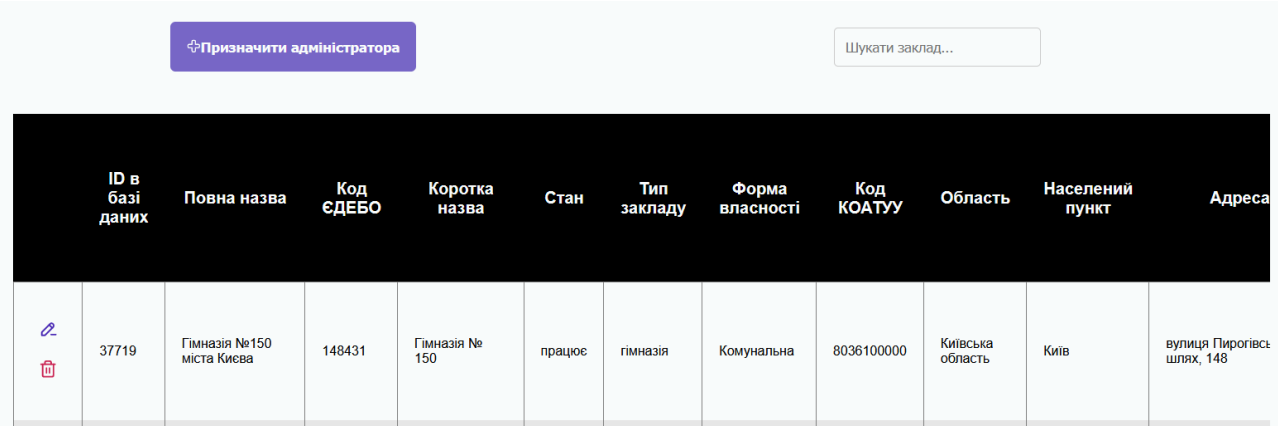


Рисунок 3.34 – Запис про навчальний заклад який буде відредаговано

Здійснюємо редагування шляхом натиснення на іконку-олівець та введення нових значень. Після цього можна натиснути на іконку-галочку для збереження змін. Бачимо, що запис успішно відредаговано (рис. 3.35).

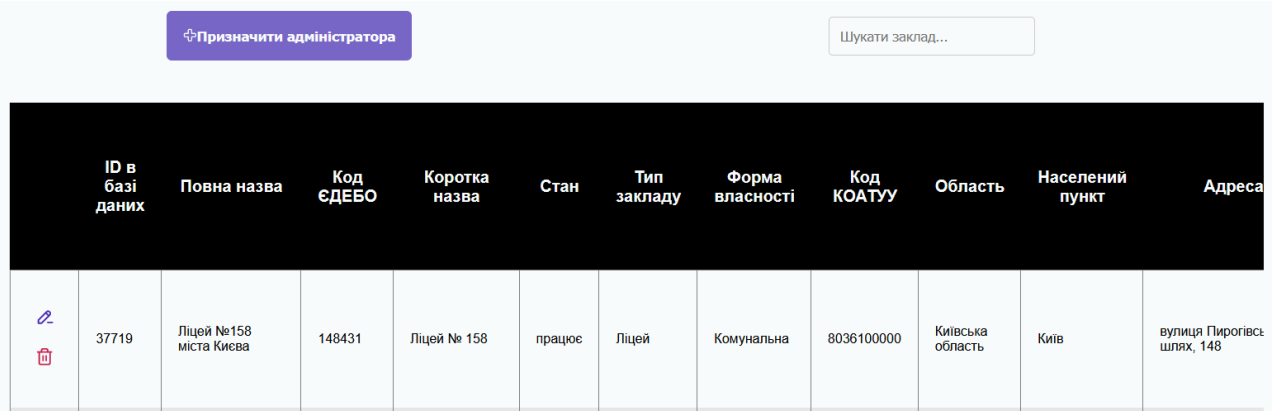


Рисунок 3.35 – Запис про навчальний заклад після редагування

Тепер можна протестувати видалення навчального закладу. Для цього варто натиснути на іконку-смітник. Запис про навчальний заклад видалено. При спробі знайти його через пошук не отримуємо ніяких результатів, що підтверджує успіх видалення (рис. 3.36).

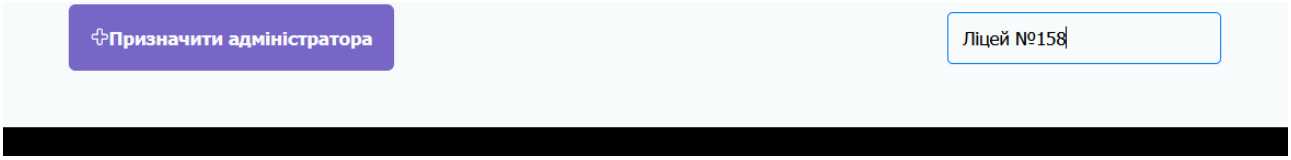


Рисунок 3.36 – Запис про навчальний заклад видалено (тому в пошуку він більше не відображається)

Висновки до третього розділу

В даному розділі було описано порядок встановлення та налаштування розробленої онлайн-платформи дистанційного навчання, та продемонстровано її інтерфейс та функціонал. Дана платформа відповідає викладеним на початку планування розробки вимогам та зможе покращити доступ до освіти навіть в умовах вимушеного дистанційного навчання.

Наприкінці розділу було виконано ручне тестування розробленої платформи. Воно дозволило упевнитись, що користувацький інтерфейс є простим та інтуїтивно зрозумілим, платформа працює коректно та правильно реагує на виключні ситуації, а весь необхідний функціонал працює як очікувано.

ВИСНОВКИ

Без ретельного планування неможливо створити якісний веб-додаток. Лише шляхом планування, аналізу існуючих на ринку конкурентних платформ, а також аналізу вимог до розробки сучасної онлайн-платформи дистанційного навчання можна створити дійсно якісну, зручну та надійну освітню платформу.

На першому етапі було важливо чітко визначити потреби та очікування користувачів платформи. Це включає в себе розуміння того, які фукнції знодобляться різним типам користувачів, яким має бути інтерфейс, які вимоги мають бути поставлені перш ніж почати розробку.

Коли вимоги чітко визначені, починається процес проектування та розробки платформи. У цей процес входить зокрема вибір технологій розробки, проектування бази даних, створення користувацького інтерфейсу та самої платформи. Важливо, щоб вона була зручною у використанні, інтуїтивно зрозумілою для пересічного користувача та не потребувала довгого навчання чи високого рівня технічної підготовки.

Завершальний етап включав в себе перевірку роботи платформи. Критично важливо упевнитись у тому, що платформа правильно реагує на виключні ситуації та що увесь її функціонал працює без нарікань. Для цього було використане ручне тестування

Отже, завдяки поетапному підходу до створення онлайн-платформи дистанційного навчання, було високоефективну, надійну та якісну освітню-платформу, яка відповідає очікуванням користувачів та забезпечує їх доступом до освіти в будь-якому місці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Всеосвіта [Електронний ресурс] – Режим доступу до ресурсу: https://vseosvita.ua/
2. Єдина Школа [Електронний ресурс] – Режим доступу до ресурсу: https://eschool-ua.com/#/
3. EdEra [Електронний ресурс] – Режим доступу до ресурсу: https://ed-era.com/
4. NextJs [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/docs
5. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en
6. Socket.io [Електронний ресурс] – Режим доступу до ресурсу: https://socket.io/
7. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/
8. SCSS [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/
9. MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/
10. Medium [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/
11. LogRocket [Електронний ресурс] – Режим доступу до ресурсу: https://logrocket.com/
12. MDN Web Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/
13. Stack Overflow [Електронний ресурс] – Режим доступу до ресурсу: https://stackoverflow.com/
14. Cloudinary [Електронний ресурс] – Режим доступу до ресурсу: https://cloudinary.com/

15. Dev To [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/

16. FreeCodeCamp [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/

17. GeeksforGeeks [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/

18. Usability Geek [Електронний ресурс] – Режим доступу до ресурсу: https://usabilitygeek.com/

19. Educative [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/

20. CodeAcademy [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/

21. MySQLTutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysqltutorial.org/

22. UXPin [Електронний ресурс] – Режим доступу до ресурсу: https://www.uxpin.com/studio/

23. FrontEndMasters [Електронний ресурс] – Режим доступу до ресурсу: https://frontendmasters.com/

24. GeeksForGeeks [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/

25. Egghead.io [Електронний ресурс] – Режим доступу до ресурсу: https://egghead.io/

26. Спільнота програмістів Dou.ua [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/

27. GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/

28. MySQL Crash Course: A Hands-on Introduction to Database Development ". No Starch Press, 2023. – 323 с.

29. Robin Wieruch. "The Road to React: Your journey to master plain yet pragmatic React.js". Independently published, 2018. – 286 с.

30. Marijn Haverbeke. "Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming". No Starch Press, 2018. – 472 с.

ДОДАТКИ

Лістинг AcademicRecords.jsx

```
"use client";

import standartStyles from "@/styles/Styles.module.scss";
import styles from "./AcademicRecords.module.scss";
import { useState, useEffect, useRef } from "react";
import AutocompleteInput from "../../UI/AutocompleteInput";
import { toast } from "react-toastify";
import { FaChalkboardTeacher } from "react-icons/fa";


export default function AcademicRecords({ userRole, user }) {
    const [className, setClassName] = useState({ number: 1, letter: "A" });
    const [classLetters, setClassLetters] = useState([]);
    const [subjects, setSubjects] = useState([]);
    const [subject, setSubject] = useState({});
    const [classObj, setClassObj] = useState({});
    const [month, setMonth] = useState(new Date().getMonth() + 1);
    const [year, setYear] = useState(new Date().getFullYear());
    const [selectedCell, setSelectedCell] = useState(null);
    const [showTinyPopup, setShowTinyPopup] = useState(false);
    const [popupData, setPopupData] = useState(null);
    const [academicRecords, setAcademicRecords] = useState([]);
    const [schoolchildren, setSchoolchildren] = useState([]);
    const [prevSelectedCell, setPrevSelectedCell] = useState({
        row: null,
        col: null,
    });
    const [isRequestInProgress, setIsRequestInProgress] = useState(false);


    const inputRef = useRef(null);
    const tableRef = useRef(null);
    const popupref = useRef(null);
    const tableContainerRef = useRef(null);


    const handleClassNumberChange = (e) => {
        setClassName((prev) => ({
            ...prev,
            number: +e.target.value,
        }));
    };


    const handleClassLetterChange = (e) => {
        setClassName((prev) => ({
            ...prev,
            letter: e.target.value,
        }));
    };


    const fetchClasses = async () => {
        const response = await fetch(`/api/institution/classes`);
        const result = await response.json();
        if (!result.success) {
            console.error(result.data);
        }
```

```
    const letters = result.data.map((el) => el.name.split("-")[1]);
    const uniqueLetters = [...new Set(letters)];
    setClassLetters(uniqueLetters);
};


const fetchSubjects = async () => {
    const response = await fetch(`/api/institution/subjects`);
    const result = await response.json();
    if (!result.success) {
        console.error(result.data);
    }


    const formatted = [];


    if (classObj) {
        result.data.forEach((el) => {
            if (el.classes_type === classObj.type) {
                formatted.push(el.name);
            }
        });
        setSubjects(formatted);
    } else {
        setSubjects(result.data);
    }
};


const fetchSchoolChildren = async () => {
    if (className) {
        const response = await fetch(
            `/api/institution/schoolchildren?class=${className.number}-${className.letter}`
        );
        const result = await response.json();
        if (!result.success) {
            console.error(result.data);
            return;
        }


        const sortByLastName = (a, b) => {
            const lastNameA = a.lastname.toUpperCase();
            const lastNameB = b.lastname.toUpperCase();


            if (lastNameA < lastNameB) {
                return -1;
            }
            if (lastNameA > lastNameB) {
                return 1;
            }
            return 0;
        };


        const sortedSchoolchildren = result.data.sort(sortByLastName);
```

```
        setSchoolchildren(sortedSchoolchildren);
    } else setSchoolchildren([]);
};


const fetchRecords = async () => {
    if (classObj && subject && subject.length > 0) {


        const response = await fetch(
            `/api/institution/academic-records?class=${classObj.name}&subject=${subject}`
        );
        const result = await response.json();
        if (!result.success) {
            console.error(result.data);
            return;
        }


        const filteredByMonth = [];
        result.data.academic_records.forEach((el) => {
            const date = new Date(el.date);
            const dateMonth = date.getMonth() + 1;
            const dateYear = date.getFullYear();
            if (dateMonth === month && dateYear === year) {
                filteredByMonth.push(el);
            }
        });


        const academicRecordsTransformed = filteredByMonth.reduce(
            (acc, obj) => {
                const { student_id, ...academicRecord } = obj;
                acc.push({ [student_id]: academicRecord });
                return acc;
            },
            []
        );

        setAcademicRecords(academicRecordsTransformed);
    }
};


useEffect(() => {
    fetchClasses();
}, []);


const fetchClass = async () => {
    if (className){
        const response = await fetch(
        `/api/institution/classes?class=${className.number}-${className.letter}`
    );
    const result = await response.json();
    if (!result.success) {
        console.error(result.data);
        return;
    }
    setClassObj(result.data);
    }
```

```
};
useEffect(() => {
    fetchClass();
}, [className.letter, className.number]);


useEffect(() => {
    fetchSubjects();
    fetchSchoolChildren();
}, [classObj]);
useEffect(() => {
    fetchRecords();
}, [month, year, classObj, subject]);
const handleOnChangeSubject = (value) => {
    if (value) {
        setSubject(value);
    }
};


const handlePrevMonth = () => {
    setMonth((prevMonth) => {
        if (prevMonth === 1) {
            setYear((prevYear) => prevYear - 1);
            return 12;
        } else {
            return prevMonth - 1;
        }
    });
};


const handleNextMonth = () => {
    setMonth((prevMonth) => {
        if (prevMonth === 12) {
            setYear((prevYear) => prevYear + 1);
            return 1;
        } else {
            return prevMonth + 1;
        }
    });
};


const handleMonthChange = (e) => {
    setMonth(parseInt(e.target.value));
};


useEffect(() => {
    if (selectedCell) {
        inputRef.current.focus();
    }
}, [selectedCell]);


const handleCellDoubleClick = (rowIndex, colIndex) => {
    if (!subject.length || subject.length === 0) {
        toast.error("Спочатку оберіть предмет");
        return;
    }
    setSelectedCell({ row: rowIndex, col: colIndex });
```

```
};

const handleKeyDown = async (event) => {
    if (event.key === "Enter") {
        const { relatedTarget } = event;
        if (
            selectedCell &&
            (!relatedTarget || !tableRef.current.contains(relatedTarget))
        ) {
            if (!isRequestInProgress) {
                const subjectName = subject;
                const classesType = classObj.type;
                const date = `${
                    selectedCell.col + 1 < 10
                        ? "0" + (selectedCell.col + 1)
                        : selectedCell.col + 1
                }.${month < 10 ? "0" + month : month}.`;
                setPrevSelectedCell((prev) => ({
                    ...prev,
                    row: selectedCell.row,
                    col: selectedCell.col,
                }));
                setSelectedCell(null);
                const data = {
                    grade: null,
                    present: null,
                    subjectName: subjectName,
                    classesType: classesType,
                    classId: classObj.id,
                    studentId: schoolchildren[selectedCell.row].id,
                    teacherUserId: user.id,
                    date: date,
                };
                if (
                    parseInt(event.target.value) &&
                    (parseInt(event.target.value) >= 1 &&
                        parseInt(event.target.value) <= 12)
                ) {
                    data.present = 1;
                    data.grade = parseInt(event.target.value);
                } else if (event.target.value.toLowerCase() === "н") {
                    data.present = 0;
                    data.grade = null;
                } else if (event.target.value === "") {
                    data.present = 1;
                    data.grade = null;
                } else {
                    toast.error(
                        'В клітинці може бути лише оцінка від 1 до 12, "н-ка" або зовсім нічого'
                    );
                    return;
                }
                setIsRequestInProgress(true);
                const response = await fetch(
                    `/api/institution/academic-records`,
                    {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(data),
                    }
                );
```

```
                const result = await response.json();
                if (!result.success) {
                    toast.error(result.data);
                    setIsRequestInProgress(false);
                } else {
                    toast.success(result.data);
                    setIsRequestInProgress(false);
                    fetchRecords();
                }
            }
        }
    }
};


const handleCellBlur = async (event) => {
    const { relatedTarget } = event;
    if (
        selectedCell &&
        (!relatedTarget || !tableRef.current.contains(relatedTarget))
    ) {
        setSelectedCell(null);
    }
};


useEffect(() => {
    document.body.addEventListener("keydown", handleKeyDown);
    document.body.addEventListener("focusout", handleCellBlur);


    return () => {
        document.body.removeEventListener("keydown", handleKeyDown);
        document.body.removeEventListener("focusout", handleCellBlur);
    };
}, [selectedCell]);



const getRecord = (rowIndex, colIndex) => {
    if (academicRecords && academicRecords.length > 0) {
        if (schoolchildren[rowIndex]) {
            let currentStudentRecords = [];
            const date = `${
                colIndex + 1 < 10 ? "0" + (colIndex + 1) : colIndex + 1
            }.${month < 10 ? "0" + month : month}.`;


            academicRecords.forEach((el) => {
                for (let key in el) {
                    if (key === schoolchildren[rowIndex].id) {
                        currentStudentRecords.push({
                            student_id: key,
                            ...el[key],
                        });
                    }
                }
            });


            if (currentStudentRecords && currentStudentRecords.length > 0) {
                for (let record of currentStudentRecords) {
```

```
                    const recordDateObj = new Date(record.date);
                    const recordMonth = recordDateObj.getMonth() + 1;
                    const recordDate = `${
                        recordDateObj.getDate() < 10
                            ? `0${recordDateObj.getDate()}`
                            : recordDateObj.getDate()
                    }.${
                        recordMonth < 10 ? "0" + recordMonth : recordMonth
                    }.`;


                    if (recordDate === date) {
                        return record;
                    }
                }
            }
        }
    }
};


const renderCellContent = (rowIndex, colIndex) => {
    const isEditable = true;


    if (
        selectedCell &&
        selectedCell.row === rowIndex &&
        selectedCell.col === colIndex &&
        isEditable
    ) {
        return (
            <input
                ref={inputRef}
                type="text"
                defaultValue={""}
                onBlur={handleCellBlur}
                onKeyDown={handleKeyDown}
                className="editable-cell"
            />
        );
    } else {
        if (academicRecords && academicRecords.length > 0) {
            if (schoolchildren[rowIndex]) {
                let currentStudentRecords = [];
                const date = `${
                    colIndex + 1 < 10 ? "0" + (colIndex + 1) : colIndex + 1
                }.${month < 10 ? "0" + month : month}.`;


                academicRecords.forEach((el) => {
                    for (let key in el) {
                        if (key === schoolchildren[rowIndex].id) {
                            currentStudentRecords.push({
                                student_id: key,
                                ...el[key],
                            });
                        }
                    }
                });
```

```
                    if (
                        currentStudentRecords &&
                        currentStudentRecords.length > 0
                    ) {
                        for (let record of currentStudentRecords) {
                            const recordDateObj = new Date(record.date);
                            const recordMonth = recordDateObj.getMonth() + 1;
                            const recordDate = `${
                                recordDateObj.getDate() < 10
                                    ? `0${recordDateObj.getDate()}`
                                    : recordDateObj.getDate()
                            }.${
                                recordMonth < 10
                                    ? "0" + recordMonth
                                    : recordMonth
                            }.`;


                            if (recordDate === date) {
                                return record.present
                                    ? (record.grade ? record.grade : "")
                                    : "Н";
                            }
                        }
                    }
                }
            }
        }
    };


    const renderTableCells = () => {
        const daysInMonth = new Date(year, month, 0).getDate();
        const rows = [];
        const dateCells = [];
        for (let i = 1; i <= daysInMonth; i++) {
            dateCells.push(
                <td key={i}>{`${i < 10 ? "0" + i : i}.${
                    month < 10 ? "0" + month : month
                }.`}</td>
            );
        }
        rows.push(
            <tr key={"datarow"}>
                <td key={"emptycell"}></td>
                {dateCells}
            </tr>
        );


        if (schoolchildren) {
            for (let i = 0; i < schoolchildren.length; i++) {
                const cells = [];
                cells.push(
                    <td key={`pib${i}`}>
                        <a
                            href={`/user/schoolchild/${schoolchildren[i].id}`}>{`${schoolchildren[i].
lastname} ${schoolchildren[i].firstname}`}</a>
                    </td>
                );
                for (let j = 0; j < daysInMonth; j++) {
                    cells.push(
```

```
                    <td
                        key={j}
                        onDoubleClick={() => handleCellDoubleClick(i, j)}
                        onMouseEnter={(e) => handleShowTinyPopup(e, i, j)}
                        onMouseLeave={(e) => setShowTinyPopup(false)}>
                        {isRequestInProgress &&
                        prevSelectedCell &&
                        prevSelectedCell.row === i &&
                        prevSelectedCell.col === j ? (
                            <span className={styles.loader}></span>
                        ) : (
                            renderCellContent(i, j)
                        )}
                    </td>
                );
            }
            rows.push(<tr key={`gradeRow${i}`}>{cells}</tr>);
        }
    }


    return rows;
};


const handleScroll = () => {
    setShowTinyPopup(false);
};


useEffect(() => {
    if (tableContainerRef.current) {
        tableContainerRef.current.addEventListener("scroll", handleScroll);
    }


    return () => {
        if (tableContainerRef.current) {
            tableContainerRef.current.removeEventListener(
                "scroll",
                handleScroll
            );
        }
    };
}, []);


const handleShowTinyPopup = (e, rowIndex, colIndex) => {
    const record = getRecord(rowIndex, colIndex);


    if (record) {
        setShowTinyPopup(true);
        setPopupData((prev) => ({
            ...prev,
            teacherId: record.teacher_id,
            teacherFirstname: record.teacher_firstname,
            teacherLastname: record.teacher_lastname,
            teacherAntroponym: record.teacher_antroponym,
        }));
```

```
                    setTimeout(() => {
                        if (record && popupref.current) {
                                popupref.current.style.left =
                                    e.target.offsetLeft + "px";
                                popupref.current.style.top =
                                    e.target.offsetTop + 48 + "px";
                        } else {
                            return;
                        }
                    }, 800);
                }
            };


            const renderPopup = () => {
                if (popupData) {
                    return (
                        <div
                            ref={popupref}
                            className={styles.popup}
                            onMouseEnter={(e) => setShowTinyPopup(true)}
                            onMouseLeave={(e) => setShowTinyPopup(false)}>
                            <FaChalkboardTeacher />{" "}
                            <a
                                href={`/user/teacher/${popupData.teacherId}`}>{`${popupData.teacherLastname}
${popupData.teacherFirstname[0]}.${popupData.teacherAntroponym[0]}.`}</a>
                        </div>
                    );
                }
            };


            return (
                <div className={styles.page}>
                    <div className={styles.header}>
                        {userRole !== 'SCHOOLCHILD' && <div>
                            <label>Клас*</label>
                            <div>
                                <input
                                    className={standartStyles.inputRegular}
                                    defaultValue="1"
                                    type="number"
                                    min="1"
                                    max="11"
                                    placeholder="1"
                                    onChange={handleClassNumberChange}
                                />
                                <select
                                    className={standartStyles.selectRegular}
                                    onChange={handleClassLetterChange}>
                                    {classLetters &&
                                        classLetters.map((el, index) => (
                                            <option
                                                key={index}
                                                value={el}>
                                                {el}
                                            </option>
                                        ))}
                                </select>
                            </div>
                        </div>}
                        <div>
```

```
                <label>Предмет*</label>
                <AutocompleteInput
                    dataToSearch={subjects}
                    onChange={handleOnChangeSubject}
                    defaultValue={subject ? subject.name : ""}
                />
            </div>
            <div>
                <label>Місяць*</label>
                <select
                    className={standartStyles.selectRegular}
                    onChange={handleMonthChange}
                    value={month}>
                    <option value="1">Січень</option>
                    <option value="2">Лютий</option>
                    <option value="3">Березень</option>
                    <option value="4">Квітень</option>
                    <option value="5">Травень</option>
                    <option value="6">Червень</option>
                    <option value="7">Липень</option>
                    <option value="8">Серпень</option>
                    <option value="9">Вересень</option>
                    <option value="10">Жовтень</option>
                    <option value="11">Листопад</option>
                    <option value="12">Грудень</option>
                </select>
            </div>
        </div>
        <div className={styles.tableContainerAndArrows}>
            <button
                className={styles.buttonArrow}
                onClick={handlePrevMonth}>
                &lt;
            </button>


            <div
                className={styles.tableContainer}
                ref={tableContainerRef}
                onBlur={handleCellBlur}>
                <div className={styles.tableHeaders}>
                    <div>
                        <span>ПІБ учениці/учня</span>
                    </div>
                    <div>
                        <span>Дата</span>
                    </div>
                </div>
                {subject && month && (
                    <table ref={tableRef}>
                        <tbody>{renderTableCells()}</tbody>
                    </table>
                )}


                {showTinyPopup && subject && renderPopup()}
            </div>
            <button
                className={styles.buttonArrow}
                onClick={handleNextMonth}>
                &gt;
            </button>
```

```
            </div>
        </div>
    );
}
```

Лістинг Chat.jsx

```
"use client";

import React, { useEffect, useState, useRef } from "react";
import styles from "./Chat.module.scss";
import { TbChecks, TbSend2 } from "react-icons/tb";
import { io } from "socket.io-client";


const socket = io("http://localhost:3001");


const Chat = ({ talker, room, currentUser }) => {
    const [currentMsg, setCurrentMsg] = useState("");
    const [messages, setMessages] = useState([]);
    const messagesEndRef = useRef(null);


    useEffect(() => {
        const fetchMessages = async () => {
            const response = await fetch(
                `/api/user/chats/messages?room-id=${room.room_id}`
            );
            const result = await response.json();


            if (result.success) {
                setMessages(result.data);
            } else {
                console.error(result.data);
            }
        };


        fetchMessages();
        handleJoinRoom();


        socket.on("receive_msg", (data) => {
            setMessages((prevChat) => [...prevChat, data]);
        });


        return () => {
            socket.off("receive_msg");
        };
    }, [room.room_id]);


    useEffect(() => {
        scrollToBottom();
    }, [messages]);


    const handleJoinRoom = () => {
        socket.emit("join_room", room.room_id);
```

```
};

const sendData = async (e) => {
    e.preventDefault();
    if (currentMsg !== "") {
        const msgData = {
            room_id: room.room_id,
            sender_id: currentUser.id,
            text: currentMsg,
            sending_datetime: new Date().toISOString(),
        };

        setMessages((prevMessages) => [...prevMessages, msgData]);

        const response = await fetch("/api/user/chats/messages", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify(msgData),
        });

        if (response.ok) {
            socket.emit("send_msg", msgData);
            setCurrentMsg("");
        } else {
            console.error("Failed to send message");
            setMessages((prevMessages) =>
                prevMessages.filter((msg) => msg !== msgData)
            );
            setCurrentMsg("");
        }
    }
};

const handleInputChange = (e) => {
    setCurrentMsg(e.target.value);
};

const scrollToBottom = () => {
    if (messagesEndRef.current) {
        messagesEndRef.current.scrollIntoView({
            behavior: "smooth",
            block: "end",
            inline: "nearest",
        });
    }
};

const formatDate = (date) => {
    const today = new Date();
    const messageDate = new Date(date);

    if (
```

```
            today.getDate() === messageDate.getDate() &&
            today.getMonth() === messageDate.getMonth() &&
            today.getFullYear() === messageDate.getFullYear()
        ) {
            return "Сьогодні";
        } else if (
            today.getDate() - messageDate.getDate() === 1 &&
            today.getMonth() === messageDate.getMonth() &&
            today.getFullYear() === messageDate.getFullYear()
        ) {
            return "Вчора";
        } else {
            const day = messageDate.getDate();
            const month = messageDate.getMonth() + 1;
            const year =
                today.getFullYear() === messageDate.getFullYear()
                    ? ""
                    : messageDate.getFullYear();
            return `${day < 10 ? "0" + day : day}.${
                month < 10 ? "0" + month : month
            }.${year}`;
        }
    };


    return (
        <div className={styles.page}>
            <div className={styles.container}>
                <div className={styles.header}>
                    <div className={styles.profilePic}>
                        <img
                            src={
                                talker.photo
                                    ? talker.photo
                                    : "/images/school-child1.png"
                            }
                            alt="Profile"
                        />
                    </div>
                    <span>{`${talker.firstname} ${talker.lastname}`}</span>
                </div>
                <div className={styles.user}>
                    {messages.length > 0 &&
                        messages.map(
                            (
                                { room_id, sender_id, text, sending_datetime },
                                key
                            ) => (
                                <div
                                    key={key}
                                    className={`${
                                        sender_id === talker.id
                                            ? styles.userRight
                                            : styles.userLeft
                                    } ${styles.userRow}`}>
                                    <div
                                        className={`${
                                            styles.profilePhotoCircle
                                        } ${
                                            sender_id === talker.id
                                                ? styles.profilePhotoCircleRight
                                                : styles.profilePhotoCircleLeft
```

```
                                        }`}>
                                        <img
                                            src={
                                                sender_id === talker.id
                                                    ? (talker.photo ? talker.photo :
'/images/teacher.png')
                                                    : (currentUser.photo ? currentUser.photo :
'/images/teacher.png' )
                                            }
                                            alt="Profile"
                                        />
                                    </div>
                                    <div className={styles.messageContainer}>
                                        <div
                                            className={`${styles.messageInfo} ${
                                                sender_id === talker.id
                                                    ? styles.messageInfoRight
                                                    : styles.messageInfoLeft
                                            }`}>
                                            <div
                                                className={`${
                                                    styles.messageText
                                                } ${
                                                    sender_id === talker.id
                                                        ? styles.messageTextRight
                                                        : styles.messageTextLeft
                                                }`}>
                                                {text}
                                            </div>
                                            <div
                                                className={`${
                                                    styles.messageStatus
                                                } ${
                                                    sender_id === talker.id
                                                        ? styles.messageStatusRight
                                                        : styles.messageStatusLeft
                                                }`}>
                                                <TbChecks />
                                            </div>
                                        </div>
                                        <div
                                            className={`${styles.messageTime} ${
                                                sender_id === talker.id
                                                    ? styles.messageTimeRight
                                                    : styles.messageTimeLeft
                                            }`}>
                                            <span>
                                                {formatDate(sending_datetime)}{" "}
                                                {new Date(
                                                    sending_datetime
                                                ).toLocaleTimeString([], {
                                                    hour: "2-digit",
                                                    minute: "2-digit",
                                                })}
                                            </span>
                                        </div>
                                    </div>
                                </div>
                            )
                        )}
                    <div ref={messagesEndRef} />
                </div>
```

```
                <div className={styles.inputContainer}>
                    <form onSubmit={sendData}>
                        <textarea
                            className={styles.messageInput}
                            value={currentMsg}
                            placeholder="Введіть повідомлення..."
                            onChange={handleInputChange}
                            onKeyDown={(e) => {
                                if (e.key === "Enter" && !e.shiftKey) {
                                    e.preventDefault();
                                    sendData(e);
                                }
                            }}
                        />
                        <button
                            type="submit"
                            className={styles.buttonIcon}>
                            <TbSend2 />
                        </button>
                    </form>
                </div>
            </div>
        </div>
    );
};


export default Chat;
```

Лістинг Classes.jsx

```
"use client";

import { useEffect, useState } from "react";
import styles from "./Classes.module.scss";
import { FaPlus, FaChalkboardTeacher } from "react-icons/fa";
import { toast } from "react-toastify";
import standartStyles from "@/styles/Styles.module.scss";
import { LuCross, LuTrash2 } from "react-icons/lu";
import AddClass from "@/components/modals/AddClass";


export default function InstitutionAdminClasses({ userRole }) {
    const [classes, setClasses] = useState([]);
    const [schoolchildren, setSchoolchildren] = useState({});
    const [expandedClasses, setExpandedClasses] = useState({});
    const [addClassModuleOpened, setAddClassModuleOpened] = useState(false);


    const fetchClassesAndChildren = async () => {
        const classesResponse = await fetch(`/api/institution/classes`);
        const classesResult = await classesResponse.json();
        if (!classesResult.success) {
            console.error(classesResult.data);
        }


        setClasses(classesResult.data);
```

```
        const schoolchildrenResponse = await fetch(
            `/api/institution/schoolchildren`
        );
        const schoolchildrenResult = await schoolchildrenResponse.json();
        if (!schoolchildrenResult.success) {
            console.error(schoolchildrenResult.data);
        }


        const childrenByClass = schoolchildrenResult.data.reduce(
            (acc, child) => {
                if (!acc[child.class_id]) {
                    acc[child.class_id] = [];
                }
                acc[child.class_id].push(child);
                return acc;
            },
            {}
        );


        setSchoolchildren(childrenByClass);
    };


    useEffect(() => {
        fetchClassesAndChildren();
    }, []);


    const toggleExpandedClass = (classId) => {
        setExpandedClasses((prevState) => ({
            ...prevState,
            [classId]: !prevState[classId],
        }));
    };


    const onDragStart = (e, classId, child) => {
        e.dataTransfer.setData("classId", classId);
        e.dataTransfer.setData("childId", child.id);
    };


    const onDrop = async (e, targetClassId) => {
        e.preventDefault();
        const childId = e.dataTransfer.getData("childId");


        const response = await fetch(`/api/institution/schoolchildren`, {
            method: "PATCH",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ id: childId, class_id: targetClassId }),
        });
        const result = await response.json();


        if (!result.success) {
            toast.error(result.data);
```

```
    }

    fetchClassesAndChildren();
};

const onDragOver = (e) => {
    e.preventDefault();
};

const handleAddClass = () => {
    setAddClassModuleOpened(true);
};

const handleOnCloseClassModule = () => {
    setAddClassModuleOpened(false);
    fetchClassesAndChildren();
};

const handleDelete = async (el) => {
    const response = await fetch(`/api/institution/classes?id=${el.id}`, {
        method: "DELETE",
    });
    const result = await response.json();
    if (!result.success) {
        toast.error(result.data);
        return;
    }
    toast.success("Клас видалено");
    fetchClassesAndChildren();
};

return (
    <div className={styles.classesPage}>
        {userRole === "INSTITUTION_ADMIN" && (
            <button
                onClick={handleAddClass}
                className={styles.addClassButton}>
                <LuCross />
                Додати клас
            </button>
        )}
        {userRole === "INSTITUTION_ADMIN" && addClassModuleOpened && (
            <AddClass
                isVisible={addClassModuleOpened}
                onClose={handleOnCloseClassModule}
            />
        )}
        <div className={styles.classesContainer}>
            <div className={styles.column0}>
                {classes && classes.map(
                    (el, index) =>
                        index % 2 !== 0 && (
                            <div
                                key={el.id}
                                className={`${styles.class}`}
                                onDrop={(e) => onDrop(e, el.id)}
```

```
                                                onDragOver={(e) => onDragOver(e)}>
                                                <div className={styles.header}>
                                                    <h3>{el.name}</h3>
                                                    {userRole === "INSTITUTION_ADMIN" && (
                                                        <button
                                                            className={`${standartStyles.buttonDelete}
${standartStyles.buttonIconNoText}`}
                                                            onClick={() =>
                                                                handleDelete(el)
                                                            }>
                                                            <LuTrash2 />
                                                        </button>
                                                    )}
                                                </div>


                                                <div className={styles.teacherLine}>
                                                    <FaChalkboardTeacher />
                                                    <span>
                                                        Класом керує:{" "}
                                                        <a
                                                            href={`/user/teacher/${el.teacher_id}`}>{`${el.lastna
me} ${el.firstname[0]}.${el.antroponym[0]}.`}</a>
                                                    </span>
                                                </div>


                                                <p>Учні:</p>
                                                <div
                                                    className={`${styles.members} ${
                                                        expandedClasses[el.id]
                                                            ? styles.membersExpanded
                                                            : ""
                                                    }`}
                                                    onClick={
                                                        expandedClasses[el.id]
                                                            ? null
                                                            : () =>
                                                                toggleExpandedClass(
                                                                    el.id
                                                                )
                                                    }>
                                                    <button
                                                        className={styles.addMemberButton}>
                                                        <FaPlus />
                                                    </button>
                                                    <div
                                                        className={`${
                                                            styles.membersCircles
                                                        } ${
                                                            expandedClasses[el.id]
                                                                ? styles.expanded
                                                                : ""
                                                        }`}>
                                                        {schoolchildren[el.id]?.length >
                                                        0 ? (
                                                            schoolchildren[el.id]
                                                                ?.slice(
                                                                    0,
                                                                    expandedClasses[el.id]
                                                                        ? schoolchildren[
                                                                            el.id
```

```
                                                                    ].length
                                                                : 2
                                                            )
                                                            .map((child, index) => (
                                                                <div
                                                                    key={index}
                                                                    className={`${
                                                                        styles.member
                                                                    } ${
                                                                        index > 1 &&
                                                                        !expandedClasses[
                                                                            el.id
                                                                        ]
                                                                            ? styles.hidden
                                                                            : ""
                                                                    }`}
                                                                    draggable="true"
                                                                    onDragStart={(e) =>
                                                                        onDragStart(
                                                                            e,
                                                                            el.id,
                                                                            child
                                                                        )
                                                                    }
                                                                    onClick={() =>
                                                                        (window.location.href =
`/user/schoolchild/${child.id}`)

                                                                    }>
                                                                    <img
                                                                        src={child && child.photo ?
child.photo : "/images/school-child1.png"}

                                                                        alt="Child"
                                                                    />
                                                                    {expandedClasses[
                                                                        el.id
                                                                    ] && (
                                                                        <span>
                                                                            {
                                                                                child.firstname
                                                                            }{" "}
                                                                            {
                                                                                child.lastname
                                                                            }
                                                                        </span>
                                                                    )}
                                                                </div>
                                                            ))
                                                        ) : (
                                                            <></>
                                                        )}
                                                    </div>
                                                </div>
                                                {expandedClasses[el.id] && (
                                                    <button
                                                        className={styles.hideButton}
                                                        onClick={() =>
                                                            toggleExpandedClass(el.id)
                                                        }>
                                                        Hide
                                                    </button>
                                                )}
                                            </div>
```

```
                                )
                            )}
                        </div>
                        <div className={styles.column1}>
                            {classes && classes.map(
                                (el, index) =>
                                    index % 2 === 0 && (
                                        <div
                                            key={el.id}
                                            className={`${styles.class}`}
                                            onDrop={(e) => onDrop(e, el.id)}
                                            onDragOver={(e) => onDragOver(e)}>
                                            <div className={styles.header}>
                                                <h3>{el.name}</h3>
                                                {userRole === "INSTITUTION_ADMIN" && (
                                                    <button
                                                        className={`${standartStyles.buttonDelete}
${standartStyles.buttonIconNoText}`}

                                                        onClick={() =>
                                                            handleDelete(el)
                                                        }>
                                                        <LuTrash2 />
                                                    </button>
                                                )}
                                            </div>


                                            <div className={styles.teacherLine}>
                                                <FaChalkboardTeacher />
                                                <span>
                                                    Класом керує:{" "}
                                                    <a
                                                        href={`/user/teacher/${el.teacher_id}`}>{`${el.lastna
me} ${el.firstname[0]}.${el.antroponym[0]}.`}</a>
                                                </span>
                                            </div>


                                            <p>Учні:</p>
                                            <div
                                                className={`${styles.members} ${
                                                    expandedClasses[el.id]
                                                        ? styles.membersExpanded
                                                        : ""
                                                }`}
                                                onClick={() =>
                                                    toggleExpandedClass(el.id)
                                                }>
                                                <button
                                                    className={styles.addMemberButton}>
                                                    <FaPlus />
                                                </button>
                                                <div
                                                    className={`${
                                                        styles.membersCircles
                                                    } ${
                                                        expandedClasses[el.id]
                                                            ? styles.expanded
                                                            : ""
                                                    }`}>
                                                    {schoolchildren[el.id]?.length >
                                                    0 ? (
```

```
                                                    schoolchildren[el.id]
                                                        ?.slice(
                                                            0,
                                                            expandedClasses[el.id]
                                                                ? schoolchildren[
                                                                        el.id
                                                                  ].length
                                                                : 2
                                                        )
                                                        .map((child, index) => (
                                                            <div
                                                                onClick={() =>
                                                                    (window.location.href =
`/user/schoolchild/${child.id}`)

                                                                }
                                                                key={index}
                                                                className={`${
                                                                    styles.member
                                                                } ${
                                                                    index > 1 &&
                                                                    !expandedClasses[
                                                                        el.id
                                                                    ]
                                                                        ? styles.hidden
                                                                        : ""
                                                                }`}
                                                                draggable="true"
                                                                onDragStart={(e) =>
                                                                    onDragStart(
                                                                        e,
                                                                        el.id,
                                                                        child
                                                                    )
                                                                }>
                                                                <img
                                                                    src={child && child.photo ?
child.photo : "/images/school-child1.png"}

                                                                    alt="Child"
                                                                />
                                                                {expandedClasses[
                                                                    el.id
                                                                ] && (
                                                                    <span>
                                                                        {
                                                                            child.firstname
                                                                        }{" "}
                                                                        {
                                                                            child.lastname
                                                                        }
                                                                    </span>
                                                                )}
                                                            </div>
                                                        ))
                                                ) : (
                                                    <></>
                                                )}
                                            </div>
                                        </div>
                                        {expandedClasses[el.id] && (
                                            <button
                                                className={styles.hideButton}
                                                onClick={() =>
```

```
                                                toggleExpandedClass(el.id)
                                            }>
                                            Сховати
                                        </button>
                                    )}
                                </div>
                            )
                        )}
                    </div>
                </div>
            </div>
        );
    }
```

Лістинг ScheduleTable.jsx

```
import React, { useState } from 'react';
import Lesson from './Lesson';
import styles from './ScheduleTable.module.scss';
import Loading from "@/components/modals/Loading";

const ScheduleTable = ({ day, lessons, onUpdateLesson, onDeleteLesson, userRole }) => {


  const hasNullOrUndefinedProperties = (obj) => {
    for (let key in obj) {
      if (obj[key] === null || obj[key] === undefined) {
        return true;
      }
    }
    return false;
  };

  return (
    <div className={styles.table}>
        {lessons && lessons.length > 0 ? lessons.map((lesson, index) =>
          !hasNullOrUndefinedProperties(lesson) ? (
            <Lesson
              key={index}
              lesson={lesson}
              onUpdate={onUpdateLesson}
              onDelete={() => onDeleteLesson(day, index)}
              userRole={userRole}
            />
          ) : null
        )
      :
      <Loading/>
      }
    </div>
  );
};


export default ScheduleTable;
```

Лістинг Lesson.jsx

```
import React, { useEffect, useState } from "react";
import styles from "./Lesson.module.scss";
```

```
import standartStyles from "@/styles/Styles.module.scss";
import { TbPencilMinus, TbCheck } from "react-icons/tb";
import { LuTrash2, LuCross } from "react-icons/lu";
import { IoIosClose } from "react-icons/io";
import { FaChalkboardTeacher } from "react-icons/fa";
import AutocompleteInput from "../../UI/AutocompleteInput";
import { useForm, Controller } from "react-hook-form";
import { toast } from "react-toastify";

export default function Lesson({ lesson, onUpdate, onDelete, userRole }) {
    const [isEditing, setIsEditing] = useState(false);
    const [editedLesson, setEditedLesson] = useState(lesson);
    const [subjects, setSubjects] = useState([]);


    const {
        register,
        handleSubmit,
        watch,
        setError,
        control,
        formState: { errors },
    } = useForm();


    const handleChange = (e) => {
        const { name, value } = e.target;
        setEditedLesson({ ...editedLesson, [name]: value });
    };


    const handleSave = () => {
        setIsEditing(false);
        onUpdate();
    };


    const handleEdit = () => {
        setIsEditing(true);
    };


    const fetchSubjects = async () => {
        const response = await fetch(`/api/institution/subjects`);
        const result = await response.json();
        if (!result.success) {
            console.error(result.data);
        }


        const formatted = [];
        result.data.forEach((el) => {
            if (el.classes_type === lesson.classes_type) {
                formatted.push(el.name);
            }
        });
        setSubjects(formatted);
    };


    useEffect(() => {
        fetchSubjects();
```

```
}, []);


const submitionHandler = async (data) => {
    if (!data.subject_name) {
        setError("subject_name", {
            type: "manual",
            message: "Введіть назву предмету",
        });
        return;
    }


    const response = await fetch(`/api/institution/schedule/lesson`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            ...data,
            lesson_id: lesson.lesson_id,
            schedule_id: lesson.schedule_id,
            classes_type: lesson.classes_type,
        }),
    });


    const result = await response.json();
    if (!result.success) {
        toast.error(result.data);
    }


    onUpdate();
};


const handleDelete = async () => {
    const response = await fetch(`/api/institution/schedule/lesson`, {
        method: "DELETE",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            id: lesson.lesson_id,
        }),
    });


    const result = await response.json();
    if (!result.success) {
        toast.error(result.data);
    }


    onUpdate();
}


return (
    <form
        onSubmit={handleSubmit(submitionHandler)}
        className={styles.lessonRow}>
        {isEditing && userRole=== 'INSTITUTION_ADMIN' ? (
            <>
                <fieldset className={styles.subjectTime}>
```

```
                <input
                    className={`${standartStyles.inputRegular}
${styles.subjectTime}`}
                    type="time"
                    name="start_time"
                    defaultValue={editedLesson.start_time}
                    onChange={handleChange}
                    {...register("start_time", {
                        required: "Час не може бути пустим",
                    })}
                />
                {errors.start_time && (
                    <span className={standartStyles.errorMessage}>
                        {errors.start_time.message}
                    </span>
                )}
                <input
                    className={`${standartStyles.inputRegular}
${styles.subjectTime}`}
                    type="time"
                    name="end_time"
                    defaultValue={editedLesson.end_time}
                    onChange={handleChange}
                    {...register("end_time", {
                        required: "Час не може бути пустим",
                    })}
                />
                {errors.end_time && (
                    <span className={standartStyles.errorMessage}>
                        {errors.end_time.message}
                    </span>
                )}
            </fieldset>
            <fieldset className={styles.subjectText}>
                <Controller
                    name="subject_name"
                    control={control}
                    render={({ field: { onChange, value } }) => (
                        <AutocompleteInput
                            dataToSearch={subjects}
                            value={value}
                            onChange={onChange}
                            defaultValue={
                                watch().subject_name
                                    ? watch().subject_name
                                    : null
                            }
                        />
                    )}
                />
                {errors.subject_name && (
                    <span className={standartStyles.errorMessage}>
                        {errors.subject_name.message}
                    </span>
                )}
                <input
                    className={standartStyles.inputRegular}
                    type="email"
                    name="teacher_email"
                    defaultValue={editedLesson.teacher_email}
                    onChange={handleChange}
                    {...register("teacher_email", {
```

```
                                        required: "Введіть email вчителя/вчительки",
                                        pattern: {
                                            value: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/,
                                            message: "Формат email невірний",
                                        },
                                    })}
                                />
                                {errors.teacher_email && (
                                    <span className={standartStyles.errorMessage}>
                                        {errors.teacher_email.message}
                                    </span>
                                )}
                            </fieldset>
                            <fieldset>
                                <button
                                    disabled={
                                        errors.subject_name ||
                                        errors.start_time ||
                                        errors.end_time ||
                                        errors.teacher_email
                                            ? true
                                            : false
                                    }
                                    type="submit"
                                    className={`${standartStyles.buttonSave}
${standartStyles.buttonIconNoText}`}
                                    onClick={handleSave}>
                                    <TbCheck />
                                </button>
                                <button
                                    className={`${standartStyles.buttonDelete}
${standartStyles.buttonIconNoText}`}
                                    onClick={() => setIsEditing(false)}>
                                    <IoIosClose />
                                </button>
                            </fieldset>
                        </>
                    ) : (
                        <>
                            <fieldset className={`${styles.subjectTime}`}>
                                <p>{lesson.start_time}</p>
                                <div className={styles.dash}></div>
                                <p>{lesson.end_time}</p>
                            </fieldset>
                            <fieldset className={styles.subjectText}>
                                <p>
                                    <a>{lesson.subject_name}</a>
                                </p>
                                {userRole !== 'TEACHER' && <div>
                                    <div className={styles.icon}>
                                        <FaChalkboardTeacher />
                                    </div>
                                    <a
                                        href={`/user/teacher/${lesson.teacher_id}`}>{`${lesson.lastname}
${lesson.firstname[0]}.${lesson.antroponym[0]}.`}</a>
                                </div>}
                                {userRole === 'TEACHER' && <div>
                                    <div className={styles.icon}>
                                        <FaChalkboardTeacher />
                                    </div>
                                    <span>{lesson.class_name}</span>
                                </div>}
```

```
                            </fieldset>
                            <fieldset>
                                {userRole=== 'INSTITUTION_ADMIN' && <button
                                    className={`${standartStyles.buttonEdit}
${standartStyles.buttonIconNoText}`}
                                    onClick={handleEdit}>
                                    <TbPencilMinus />
                                </button>}
                                {userRole=== 'INSTITUTION_ADMIN' && <button
                                    className={`${standartStyles.buttonDelete}
${standartStyles.buttonIconNoText}`}
                                    onClick={handleDelete}>
                                    <LuTrash2 />
                                </button>}
                            </fieldset>
                        </>
                    )}
                </form>
            );
        }
```

Лістинг session.js

```
import 'server-only';
import { SignJWT, jwtVerify } from 'jose';
import { cookies } from 'next/headers';

const secretKey = process.env.SESSION_SECRET
const encodedKey = new TextEncoder().encode(secretKey)

export async function encrypt(payload) {
  return new SignJWT(payload)
    .setProtectedHeader({ alg: 'HS256' })
    .setIssuedAt()
    .setExpirationTime('7d')
    .sign(encodedKey)
}


export async function createSession(userId) {
    const expiresAt = new Date(Date.now() + 7 * 24 * 60 * 60 * 1000)
    const session = await encrypt({ userId, expiresAt })

    cookies().set('session', session, {
      httpOnly: true,
      secure: true,
      expires: expiresAt,
      sameSite: 'lax',
      path: '/',
    })
}

export async function decrypt(session) {
  try {
    const { payload } = await jwtVerify(session, encodedKey, {
      algorithms: ['HS256'],
    })
    return payload
  } catch (error) {
    console.error('Failed to verify session')
  }
}
```

```
export async function updateSession() {
    const session = cookies().get('session').value
    const payload = await decrypt(session)

    if (!session || !payload) {
      return null
    }

    const expires = new Date(Date.now() + 7 * 24 * 60 * 60 * 1000)
    cookies().set('session', session, {
      httpOnly: true,
      secure: true,
      expires: expires,
      sameSite: 'lax',
      path: '/',
    })
}

export function deleteSession() {
  cookies().delete('session')
}
```

Лістинг dal.js

```
import "server-only";

import { cookies } from "next/headers";
import { decrypt } from "./session";
import User from "@/models/User";
import Role from "@/models/Role";


export const verifySession = async () => {
    if (cookies().get("session")?.value) {
        const cookie = cookies().get("session").value;
        const session = await decrypt(cookie);


        if (!session.userId) {
            redirect("/auth/sign-in");
        }


        return { isAuth: true, userId: session.userId };
    } else return null;
};


export const getUser = async () => {
    const session = await verifySession();


    if (!session) return null;


    try {
        const result = await User.findById(session.userId);
        const user = result.data;
```

```
        return user;
    } catch (error) {
        console.error("Failed to fetch user");
        return null;
    }
};


export const getUserRole = async () => {
    const user = await getUser();
    if (user) {
        const userRoles = await Role.findByUserId(user.id);
        const role = await Role.findById(userRoles.data.role_id);
        return role.data.role_name;
    } else return null;
};


export const getInstitutionUseedCode = async () => {
    const user = await getUser();
    if (user) {
        const userRoles = await Role.findByUserId(user.id);
        const useedCode = userRoles.data.institution_useed_code;
        return useedCode;
    } else return null;
};
```

Лістинг server.js

```
const http = require("http");
const { Server } = require("socket.io");
const cors = require("cors");

const httpServer = http.createServer();


const io = new Server(httpServer, {
    cors: {
        origin: "http://localhost:3000",
        methods: ["GET", "POST"],
        allowedHeaders: ["my-custom-header"],
        credentials: true,
    },
});


io.on("connection", (socket) => {
    console.info("A user connected:", socket.id);


    socket.on("join_room", (roomId) => {
        socket.join(roomId);
        console.info(`user with id-${socket.id} joined room - ${roomId}`);
    });


    socket.on("send_msg", (data) => {
        socket.to(data.room_id).emit("receive_msg", data);
    });
```

```
    socket.on("disconnect", () => {
        console.info("A user disconnected:", socket.id);
    });
});


const PORT = process.env.PORT || 3001;
httpServer.listen(PORT, () => {
    console.info(`Socket.io server is running on port ${PORT}`);
});
```