Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ШКІЛЬНИМ ЗАКЛАДОМ З ВИКОРИСТАННЯМ REACT, NEXT.JS ТА POSTGRESQL
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Зоан Суан Тунг
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Керівник доц. Руденко Д. О.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зоан Суан Тунг
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку для управління шкільним закладом з використанням React, Next.js, та PostgreSQL

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 31 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, мова програмування TypeScript, бібліотека React, фреймворк Next.js, платформа Node.js, програмне середовище розробки Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних шкільних застосунків.
2. Дослідити особливості та засоби реалізації.
3. Моделювання структури застосунку.
4. Реалізація інформаційної панелі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми шкільних застосунків, постановка задачі, зображення дизайну, блок-схеми архітектури.

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Руденко Д. О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 63 с., 1 табл., 19 рис., 18 джерел.

ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ШКІЛЬНИМ ЗАКЛАДОМ, БАЗА ДАНИХ, ІНТЕРФЕЙС КОРИСТУВАЧА, ІНФОРМАЦІЙНА ПАНЕЛЬ.

Об’єктом роботи є застосунок для управління шкільним закладом.

Метою роботи є розробка застосунку, який дозволяє адміністрації, вчителям, батькам та учням ефективно взаємодіяти в межах освітнього процесу.

У рамках роботи створено оптимізовану базу даних для зберігання інформації про користувачів, навчальні предмети, розклади, відвідуваність, оцінки та інші дані.

Розроблено зручний інтерфейс користувача для перегляду списків, реалізовано функції створення різних об’єктів, внесення оцінок, управління користувачами та пошук.

У результаті роботи створено сучасний вебзастосунок для адміністративного управління школою. Система поєднує зручний для користувача інтерфейс з функціональною серверною частиною. Застосунок полегшує комунікацію між усіма учасниками процесу.

SCHOOL MANAGEMENT APPLICATION, DATABASE, USER INTERFACE, DASHBOARD

The object of the work is an application for school management.

The aim of the work is to develop an application that allows administration, teachers, parents and students to interact effectively within the educational process.

The project includes an optimized database to store information about users, subjects, schedules, attendance, grades, and other data.

Was developed a convenient user interface for viewing lists and functions for creating various items, entering grades, managing users and searching system.

As a result, a modern web application for school administration was created. the system combines a user-friendly interface with a functional communication between all participants in the process.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД – система керування базами даних

ПК – персональний комп’ютер

UI – User Interface (призначений для користувача інтерфейс)

UX – User Experience (досвід користувача)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

API – Application Programming Interface (інтерфейс програмування додатків)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертекста)

SSR – Server-Side Rendering (серверний рендеринг)

ISR – Incremental Static Regeneration (інкрементна статична генерація)

CRUD – Create Read Update Delete (створення, читання, оновлення, видалення)

JWT – JSON Web Token (JSON-токен)

ORM – Object-Relational Mapping (об’єктно-реляційне відображення)

ACID – Atomicity Consistency Isolation Durability (атомарність, узгодженість, ізольованість, довговічність)

SPA – Single Page Application (односторінковий застосунок)

DOM – Document Object Model (об’єктна модель документа)

REST – Representational State Transfer (передача репрезентативного стану)

ВСТУП

У сучасному світі цифрові технології набули ключового значення в усіх сферах життя, зокрема в освіті. Завдяки розвитку Інтернету та вебтехнологій з’явилась можливість не лише отримувати доступ до інформації в будь-який час, а й ефективно керувати навчальним процесом через цифрові інструменти. Одним із важливих напрямів цифровізації шкільної освіти стало створення спеціалізованих інформаційних систем для адміністрування навчального середовища.

Інформаційні панелі як форма таких систем відіграють роль універсальних платформ для доступу до навчальних, адміністративних і організаційних даних [1]. Вони дозволяють користувачам – адміністрації, вчителям, учням і батькам – переглядати та обробляти інформацію, пов’язану з уроками, класами, оцінками, завданнями, екзаменами, відвідуванням, подіями тощо. Завдяки цьому забезпечується прозорість процесів, покращується комунікація між усіма учасниками освітнього процесу та оптимізується управління шкільною діяльністю.

У межах проєкту передбачається створення ефективної бази даних для зберігання основної інформації про школу, розробка функціонального вебінтерфейсу з використанням сучасних технологій (React, Next.js), а також реалізація захищеного доступу для різних категорій користувачів (адміністрації, вчителів, учнів, батьків).

Результатом розробки стане інформаційна система, яка сприятиме покращенню якості освітнього процесу, забезпечить швидкий доступ до актуальних даних, оптимізує взаємодію між учасниками навчального процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз вимог до застосунку

Початковим етапом у створенні інформаційної системи, зокрема у сфері освіти, є комплексний аналіз предметної області, формалізація робочих процесів та чітке визначення функціональних вимог. В умовах цифровізації освітніх закладів стає очевидною необхідність у створенні єдиного інформаційного простору, який забезпечував би зручний доступ до ключової інформації для адміністрації, викладачів, учнів та їхніх батьків. Саме тому особливу актуальність набувають інформаційні панелі як вебзастосунки для централізованого керування навчальними процесами та адміністративними задачами.

Інформаційна панель шкільного закладу має відповідати широкому спектру функціональних та нефункціональних вимог, які забезпечують її ефективність, масштабованість, зручність використання та безпеку. На етапі аналізу доцільно виокремити ключові аспекти, які впливають на архітектуру рішення, вибір технологій і реалізацію функціоналу:

– визначення функціональних вимог: система має забезпечувати доступ до різних інформаційних блоків: даних про викладачів, учнів, батьків, навчальні предмети, розклад занять, домашні завдання, оцінки, екзамени, відвідування, шкільні події тощо. Кожна роль користувача повинна мати відповідний рівень доступу та можливість здійснювати певні дії (перегляд, редагування, додавання, експортування інформації);

– створення гнучкої структури бази даних: ефективне зберігання інформації вимагає розробки продуманої реляційної моделі, що включає сутності «Учень», «Вчитель», «Клас», «Предмет», «Завдання», «Оцінка», «Відвідування» тощо. PostgreSQL як СКБД забезпечує масштабованість, підтримку транзакцій, складних запитів та збережених процедур, що ідеально підходить для освітнього контексту [2];

– безпека та захист даних: інформація про учнів, оцінки, персональні дані батьків та викладачів має оброблятись відповідно до норм конфіденційності. Аутентифікація, авторизація – обов'язкові аспекти системи;

– розробка зручного та доступного інтерфейсу: інтерфейс має бути інтуїтивно зрозумілим, з чіткою структурою навігації, візуальними підказками та адаптивним дизайном. Використання React і Next.js дозволяє реалізувати сучасний фронтенд із високою продуктивністю та швидким рендерингом [6];

– платформна незалежність та мобільність: система повинна працювати однаково коректно на різних пристроях: ПК, планшетах, смартфонах. У контексті зростаючої мобільності користувачів адаптивність інтерфейсу – ключова вимога до сучасного вебзастосунку.

Для реалізації проєкту за допомогою сучасних вебтехнологій важливо дотримуватись принципу «чистої архітектури», де фронтенд, бекенд та база даних мають чітко визначені межі відповідальності. React забезпечує високу динамічність клієнтського інтерфейсу, Next.js додає серверний рендеринг, а PostgreSQL гарантує надійність та продуктивність роботи з великим обсягом даних.

Окрім безпосередньо технічної реалізації, надзвичайно важливим аспектом у процесі створення ефективної та стійкої інформаційної системи для управління шкільним закладом є комплексне розуміння специфіки функціонування самої освітньої установи, її внутрішніх процесів, структури та вимог з боку усіх груп користувачів. На відміну від систем електронної комерції або платформ розважального характеру, освітні інформаційні системи мають справу з великою кількістю чутливої, структурованої та взаємозалежної інформації – персональні дані учнів, батьків, викладачів, розклади занять, успішність, медичні довідки, результати контрольних заходів, історія навчання, відвідуваність, адміністративні рішення, події тощо.

Управління шкільним закладом передбачає постійний обмін даними між усіма учасниками освітнього процесу. На практиці це означає, що система повинна враховувати одночасну діяльність кількох ролей: адміністратора, який формує розклади, додає або видаляє класи, призначає вчителів; вчителів, які вносять оцінки, перевіряють домашні завдання, відзначають відвідуваність; батьків, які хочуть оперативно дізнаватись про успішність своїх дітей; та самих учнів, які мають доступ до графіків занять, домашніх завдань і результатів. Усі ці дії мають відбуватись у межах єдиного цифрового середовища, що гарантує цілісність, актуальність та безпечність даних.

Важливо також врахувати потребу у масштабованості платформи. З плином часу школа може збільшувати кількість класів, учнів, викладачів; можливо – розширюватися до міжшкільної або регіональної системи, тому база даних, логіка авторизації, фільтрація даних за ролями й обмеження прав доступу повинні бути побудовані надійно та з урахуванням майбутнього розширення. Це означає, що ще на етапі проєктування слід закласти модульність, структурованість та принципи повторного використання компонентів.

Дизайн користувацького інтерфейсу має відповідати очікуванням різних груп користувачів. Адміністративний модуль може бути більш складним і функціональним, але інтерфейси для батьків чи учнів повинні бути максимально простими, адаптивними до мобільних пристроїв, з великими кнопками, чіткими повідомленнями та логічною навігацією. Особливої актуальності набуває адаптивність дизайну до смартфонів, з огляду на те, що більшість користувачів взаємодіють із системою саме з мобільних пристроїв.

1.2 Аналіз існуючих систем освітніх закладів

У сучасному освітньому середовищі спостерігається активне впровадження цифрових технологій, спрямованих на автоматизацію та оптимізацію процесів управління навчальними закладами [3]. Інформаційні системи для шкіл та освітніх установ відіграють ключову роль у забезпеченні ефективної організації навчального процесу, обліку результатів, адміністрування персоналу та підтримці взаємодії між учнями, вчителями, батьками та адміністрацією.

Такі системи, як правило, включають модулі електронного журналу, обліку відвідуваності, складання розкладу, управління навчальними планами, комунікаційні інструменти та бази даних з інформацією про учасників освітнього процесу. Їх основна мета – зробити управління школою прозорим, структурованим і доступним для всіх зацікавлених сторін. Крім того, сучасні вебзастосунки дозволяють працювати з освітньою інформацією в режимі реального часу, що особливо актуально в умовах дистанційного або змішаного навчання.

Сьогодні на українському ринку існує кілька популярних систем, які активно використовуються у загальноосвітніх школах. Найбільш поширеними серед них є платформи «Єдина школа» [4] та «Моя освіта» [5]. Обидва сервіси реалізують широкий спектр функціональних можливостей, включно з електронним журналом, розкладом занять, системою оцінювання, обліком відвідуваності, обміном повідомленнями тощо. Проте попри зовнішню подібність, кожна система має власні особливості реалізації, переваги й обмеження, які варто враховувати при розробці власного рішення.

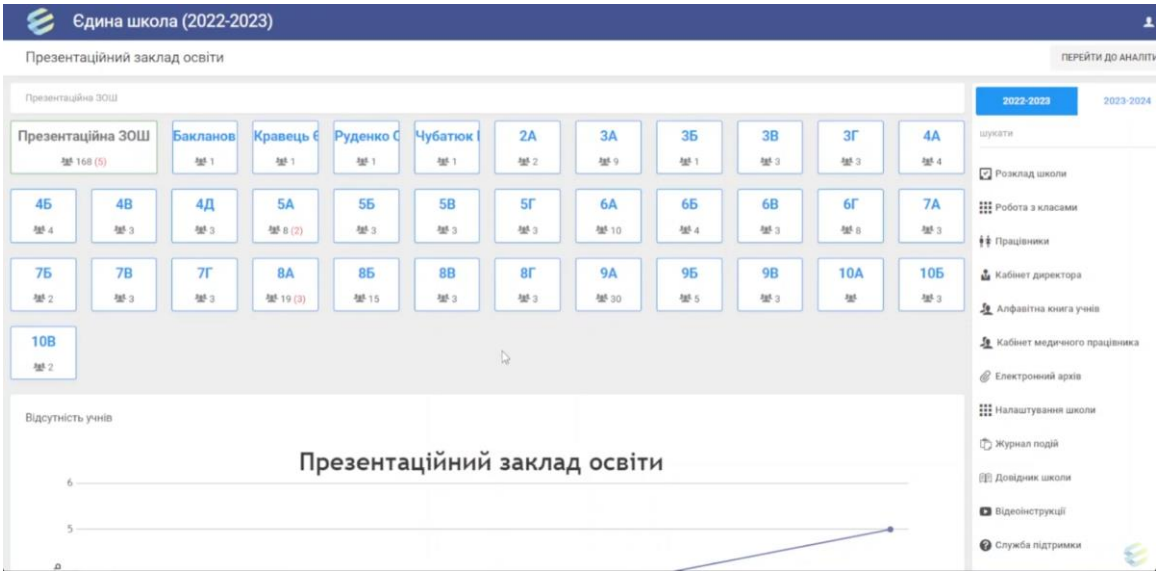
1.2.1 Аналіз системи «Єдина школа»

«Єдина школа» – це державна платформа, розроблена з метою цифрової трансформації української системи середньої освіти. Сервіс призначений для автоматизації ключових процесів у шкільному адмініструванні, забезпечуючи зручний онлайн-доступ до розкладів занять, журналів оцінювання, обліку відвідуваності, внутрішньої комунікації та звітності. Платформа створена у межах загальнодержавної ініціативи впровадження цифрової освіти, і вже впроваджена в багатьох школах по всій Україні.

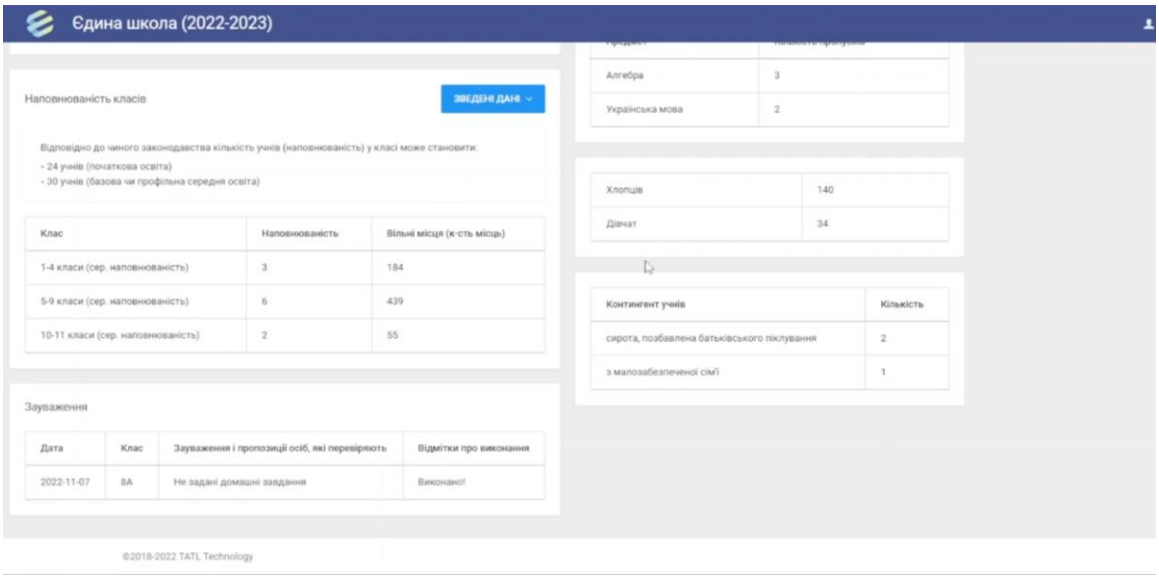
Система орієнтована на просту та інтуїтивну взаємодію з користувачами різних ролей – адміністраторами, вчителями, учнями та батьками. «Єдина школа» забезпечує єдиний доступ до важливої інформації для всіх учасників навчального процесу через вебінтерфейс та підтримку мобільних пристроїв. Однією з ключових переваг цієї системи є централізованість, що дозволяє органам освіти на державному рівні моніторити навчальні показники та адмініструвати навчальні заклади в єдиній базі даних.

Для початку роботи з платформою необхідно авторизуватися. На головній сторінці інтерфейсу зображено ключові функціональні елементи, серед яких:

- перелік класів зі вказаною кількістю учнів;
- графік відсутності учнів;
- інформація про успішність учнів;
- інформація про пропуски учнів;
- інформація по наповнюваності класів;
- інформація по зауваженням;
- інформація щодо гендерних показників та контингенту учнів;
- навігаційна панель.



а)



б)

Рисунок 1.1 – Інформаційна панель адміністратора системи

а) перелік класів, графік відсутності та навігаційна панель;

б) інформація про пропуски, наповнюваність, зауваження та контингент

На рисунку 1.2 представлено інтерфейс модуля керування класами. Даний розділ призначений для перегляду, редагування та адміністрування інформації про всі наявні класи навчального закладу.

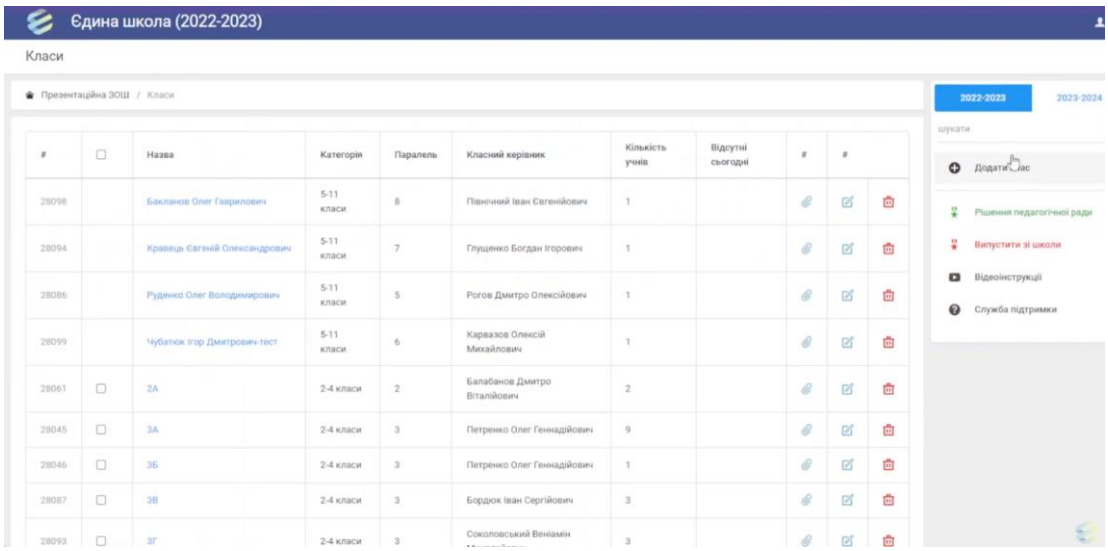


Рисунок 1.2 – Інтерфейс модуля керування класами

У центральній частині виведено таблицю з основними характеристиками кожного класу, зокрема:

– назва класу або відповідального викладача;

– категорія (наприклад, 5-11 класи або 2-4 класи);

– паралель – класний рівень;

– класний керівник – прізвище, ім’я та по батькові відповідального викладача;

– кількість учнів у класі;

– кількість відсутніх на поточний день.

У правій частині кожного рядка таблиці розташовані кнопки для перегляду, редагування або видалення запису. Це дозволяє адміністрації оперативно оновлювати або коригувати інформацію про конкретний клас. У верхній частині сторінки можна перемикатися між навчальними роками (наприклад, 2022–2023 або 2023–2024), що спрощує навігацію по архівних даних. Також реалізовано рядок пошуку, який дозволяє швидко знайти потрібний клас за іменем чи іншими критеріями.

На рисунку 1.3 зображено інтерфейс перегляду учнів у складі конкретного класу (в даному випадку – 7Б клас) у системі «Єдина школа».

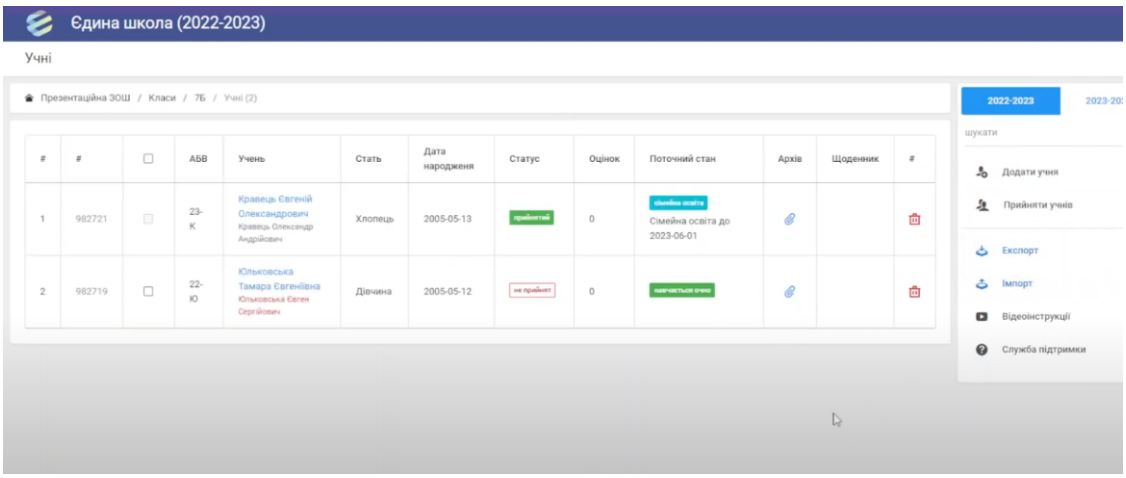


Рисунок 1.3 – Інтерфейс перегляду учнів

Даний розділ призначений для адміністрування списку учнів, перегляду їхніх особистих даних, статусів, навчальної форми та поточного стану.

Таблиця в центрі інтерфейсу містить такі ключові стовпці:

– номер запису та ID учня – унікальний ідентифікатор у базі системи;

– ПІБ учня – з посиланням на повну картку профілю;

– стать;

– дата народження;

– статус – індикатор, чи учень зарахований до класу;

– кількість оцінок у журналі;

– поточний стан – форма навчання (наприклад, сімейна освіта, навчання очно);

– архів – посилання на документи або архівні записи;

– щоденник – доступ до індивідуального щоденника учня.

Праворуч на сторінці розташоване контекстне меню, що містить корисні функції для адміністратора або класного керівника.

На рисунку 1.4 продемонстровано інтерфейс модуля «Розклад школи», що дозволяє адміністрації, вчителям та іншим користувачам переглядати структуру навчального тижня в межах поточного навчального року.

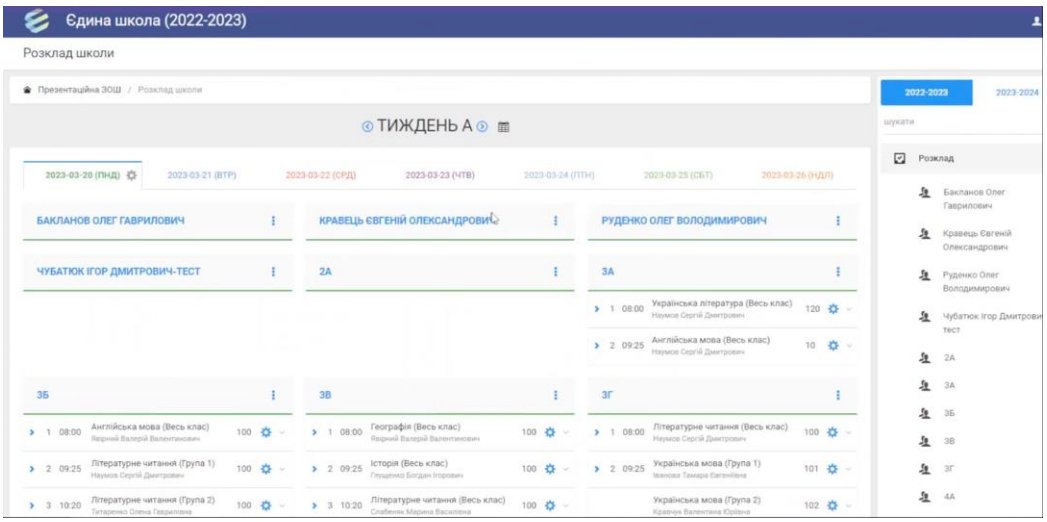


Рисунок 1.4 – Сторінка перегляду розкладу

У верхній частині екрана відображено панель навігації за датами та позначенням типу тижня (наприклад, тиждень А). За допомогою горизонтального скролу можна переміщуватись між днями тижня – понеділок, вівторок, середа тощо. Кожен день має власне кольорове маркування для швидкої візуальної орієнтації.

Основна частина інтерфейсу складається з табличного подання розкладу для класів і вчителів, згрупованих у стовпці:

– у заголовках кожного блоку зазначено ПІБ вчителя або назву класу;

– у кожному осередку показано назву предмету, форму проведення (весь клас або група), ПІБ викладача, аудиторію, а також час початку заняття.

Поруч з кожною парою знаходиться іконка налаштувань, що вказує на можливість редагування інформації про конкретне заняття або його параметри.

Функціонал модуля розкладу є одним із ключових в системі «Єдина школа», оскільки він забезпечує прозору організацію навчального процесу та зручний візуальний доступ до всієї структури занять протягом тижня.

Проаналізувавши ключові елементи інтерфейсу платформи «Єдина школа» та ознайомившись із її основними функціональними можливостями, можна виокремити головні переваги та недоліки даної освітньої інформаційної системи:

– безкоштовність використання для державних закладів освіти, що дозволяє впроваджувати систему без додаткового фінансового навантаження;

– наявність електронного журналу, розкладу, звітності – повний набір базових функцій для організації навчального процесу;

– ролі з різними правами доступу, що дозволяє адаптувати інтерфейс для адміністрації, учителів, учнів і батьків;

– підтримка архіву даних та можливість роботи з навчальними роками, що дозволяє зберігати історію навчання.

Недоліки:

– застарілий дизайн і недостатня адаптація до UX/UI стандартів, що може ускладнювати роботу користувачам з низькою цифровою грамотністю;

– відсутність повноцінної мобільної версії з усім функціоналом, що обмежує доступність із мобільних пристроїв;

З огляду на виявлені характеристики, система «Єдина школа» є ефективним, проте досить жорстко стандартизованим інструментом. Вона добре підходить для централізованого обліку та контролю, але має низку обмежень у гнучкості та адаптації, які варто враховувати при розробці власного, локалізованого застосунку для шкільного адміністрування.

1.2.2 Аналіз системи «Моя освіта»

Інформаційна система «Моя освіта» була створена для організації базового електронного супроводу навчального процесу в загальноосвітніх навчальних закладах. Платформа реалізує ключові функції, необхідні для роботи школи: ведення обліку успішності учнів, облік відвідуваності, доступ до розкладу занять, а також елементи внутрішньої комунікації між учасниками освітнього процесу.

Система функціонує як вебзастосунок, доступний через браузер, і передбачає розподіл ролей між адміністрацією, вчителями, учнями та

батьками. Кожен користувач має доступ до персоналізованої інформації відповідно до своєї ролі, що забезпечує структурованість та контроль за навчальним процесом.

Після авторизації користувача відображається основна сторінка персонального кабінету в інформаційній системі «Моя освіта» (рис. 1.5). Інтерфейс має просту структуру, розраховану на базову навігацію по функціях системи.

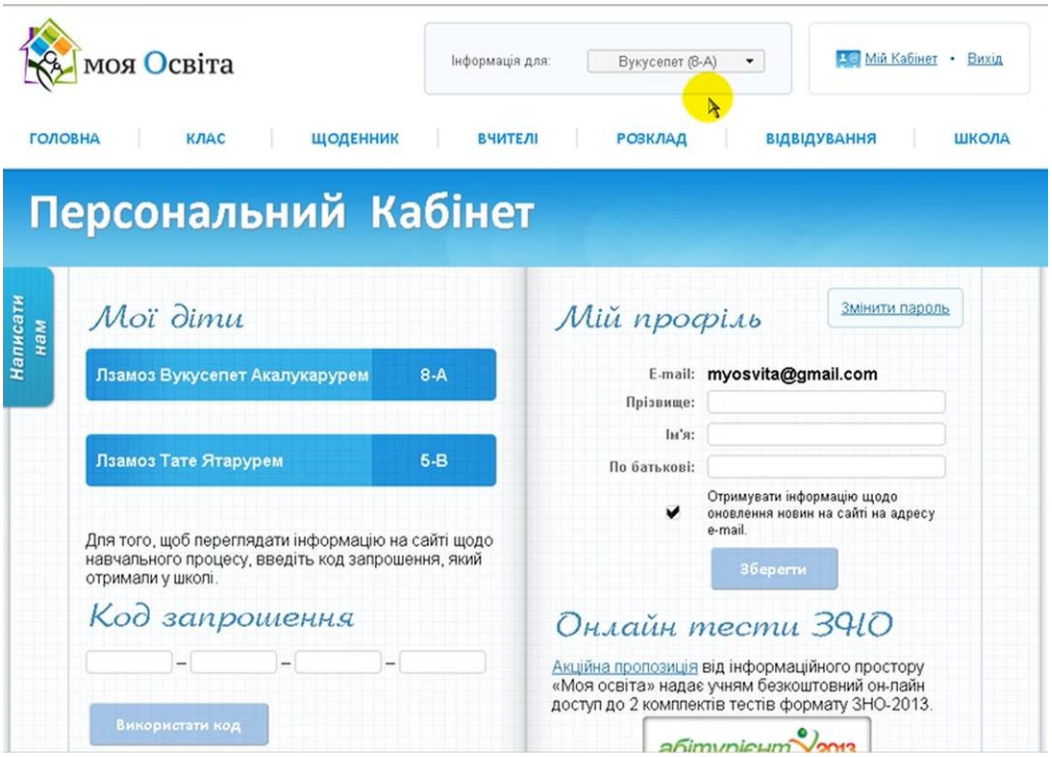


Рисунок 1.5 – Головна сторінка

У центральній частині екрана розміщено два ключові блоки:

– «Мої діти» – розділ, у якому користувач (батько або мати) бачить перелік зареєстрованих дітей, прив'язаних до облікового запису, із зазначенням класу, в якому навчається кожна дитина (наприклад, 8-А, 5-В). Вибір конкретної дитини дозволяє переглянути відповідну навчальну інформацію;

– «Мій профіль» – форма редагування контактних даних користувача (електронна пошта, ПІБ), а також налаштування розсилки новин на електронну пошту. Передбачено функцію зміни пароля.

У нижній частині лівого блоку присутнє поле для введення коду запрошення, який надається в школі для зв’язування облікового запису з конкретним класом або учнем. Цей механізм забезпечує базову верифікацію користувачів.

Угорі інтерфейсу розташоване меню навігації, яке містить посилання на основні розділи. Над меню – випадаючий список для вибору дитини, якщо до облікового запису прив’язано кількох учнів, а також посилання на «Мій кабінет» і кнопку виходу з системи.

На рисунку 1.6 зображено сторінку розділу «Клас», яка дозволяє учасникам освітнього процесу переглядати поточний розклад занять на конкретний день тижня та отримувати інформацію про домашні завдання і новини, пов’язані з навчальним процесом.

Рисунок 1.6 – Сторінка розділу «Клас»

Інтерфейс розділений на кілька логічних блоків:

– «Розклад на день» (лівий блок) – представлений у вигляді таблиці з трьома стовпцями: номер уроку, назва предмету та кабінет. Це дозволяє учням швидко орієнтуватися щодо послідовності занять та локацій. Присутність двох кабінетів для одного предмета (наприклад, геометрії) може свідчити про поділ класу на групи;

– «Сьогодні задали» – таблиця із домашніми завданнями, у якій вказано назву предмета, тип завдання (наприклад, реферат), тему або сторінки для опрацювання. Це дозволяє учням зручно відслідковувати, що потрібно виконати на наступні уроки;

– Новини (правий блок) – динамічна стрічка інформаційних повідомлень, яка містить анонси важливих подій, таких як графік проведення ЗНО, дати канікул, інформацію про пробне тестування тощо. Кожне повідомлення має заголовок, дату публікації та посилання на повний текст. Це дає змогу учням і батькам бути в курсі актуальних змін у навчальному процесі.

На рисунку 1.7 представлено інтерфейс розділу «Щоденник», який надає учням і батькам доступ до інформації щодо оцінок та домашніх завдань за конкретний період навчання.

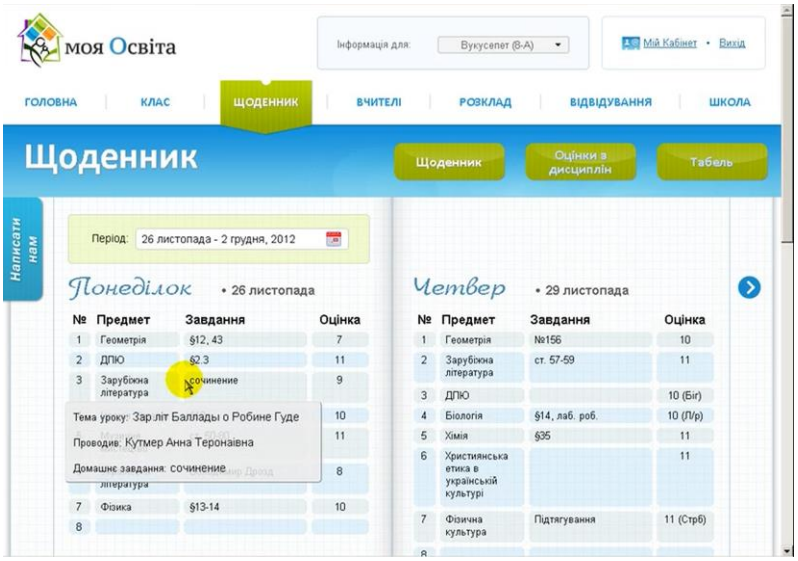


Рисунок 1.7 – Сторінка розділу «Щоденник»

У верхній частині інтерфейсу розміщено панель вибору періоду, що дозволяє користувачам обрати навчальний тиждень для перегляду даних. Це забезпечує зручну навігацію по хронології навчального процесу.

Основна частина сторінки поділена на два вертикальні блоки, які відображають інформацію щодо уроків у різні дні тижня. Кожен блок містить таблицю, що складається з таких стовпців:

- порядковий номер уроку в розкладі;
- назва навчальної дисципліни;
- короткий запис домашнього завдання;
- виставлена оцінка за урок.

Особливістю цього інтерфейсу є наявність інтерактивних підказок. При наведенні курсора на домашнє завдання з’являється спливаюче вікно, у якому зазначено тему уроку, ім’я та прізвище викладача, а також повний текст домашнього завдання. Це дає можливість користувачу отримати розширену інформацію без необхідності переходити на інші сторінки. Також є кнопки для зручного переходу до сторінок з оцінками дисциплін та табелем учня.

На рисунку 1.8 представлено інтерфейс розділу «Відвідування» в інформаційній системі «Моя освіта», що призначений для моніторингу часу перебування учня у навчальному закладі протягом обраного періоду.

У верхній частині екрана розташовано панель вибору дат, яка дозволяє задати період відображення. Вибір здійснюється за допомогою інтегрованого календаря. Інтерфейс поділено на дні тижня, кожен з яких містить відповідну дату та часові мітки: вхід – фіксується час входу учня до навчального закладу та вихід – фіксується час, коли учень залишив заклад.

Для деяких днів може бути зафіксовано декілька пар входу-виходу, що свідчить про можливість повторного відвідування (наприклад, вівторок, 19 лютого). Це може бути корисним для аналізу неповного навчального дня або запізнень.

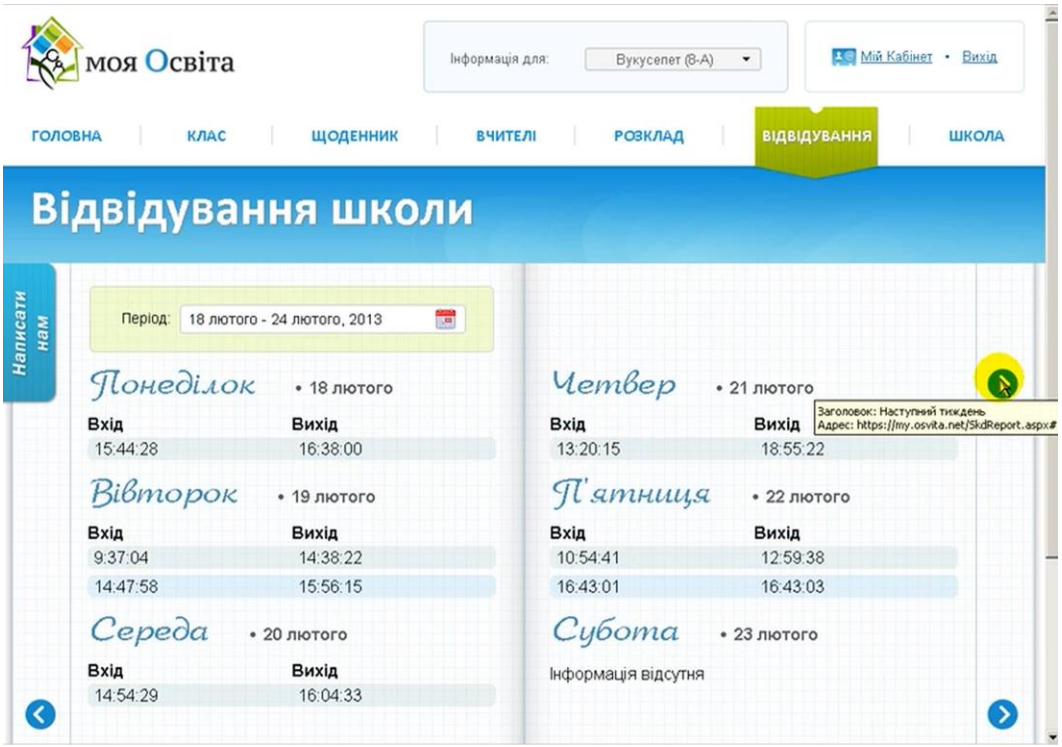


Рисунок 1.8 – Сторінка розділу «Відвідування школи»

Якщо учень не відвідував школу в конкретний день, система виводить повідомлення «Інформація відсутня». У правій частині інтерфейсу доступна кнопка переходу до наступного тижня, що дозволяє переглядати історію відвідуваності у динаміці.

Після детального ознайомлення з функціональністю платформи «Моя освіта» та аналізу її інтерфейсних рішень, можна визначити основні сильні сторони системи, а також виявити її недоліки, які впливають на зручність та ефективність використання у навчальному середовищі.

Переваги:

– наявність базових модулів забезпечує мінімально необхідний функціонал для організації навчального процесу;

– розподіл прав доступу між батьками, учнями та вчителями дозволяє адаптувати інтерфейс до ролі користувача;

– інтерактивні елементи (підказки з темами уроків і деталями домашніх завдань), покращують зручність використання;

– фіксація часу входу та виходу зі школи дає можливість точно аналізувати відвідуваність;

– простий інтерфейс полегшує взаємодію користувачам з базовими навичками володіння комп’ютером.

Недоліки:

– застарілий дизайн інтерфейсу та відсутність адаптації до сучасних UX/UI стандартів негативно впливають на зручність користування;

– обмежена функціональність;

– низький рівень адаптації під мобільні пристрої, що ускладнює роботу зі смартфонів та планшетів;

– незручна навігація та помітні затримки у відображенні інформації можуть викликати труднощі у користувачів.

Хоча система реалізує основні функції, необхідні для організації навчального процесу, її інтерфейс є досить застарілим, а функціональність – обмеженою. Дизайн платформи не відповідає сучасним стандартам користувацького досвіду, а швидкість роботи та логіка навігації часто викликають складнощі у користувачів.

Доступ до системи здійснюється через браузер, однак адаптація під мобільні пристрої є мінімальною. Платформа не підтримує повноцінне налаштування системи або розширення функцій, що суттєво обмежує її використання у більш складних сценаріях адміністрування навчального закладу.

Незважаючи на ці недоліки, система все ще залишається актуальною для частини закладів освіти, які не мають ресурсів для впровадження новіших або комерційних рішень.

1.3 Постановка задачі

Таким чином, створення сучасної та зручної інформаційної системи для організації та супроводу навчального процесу є актуальним завданням у сфері цифровізації освіти. Впровадження подібних рішень дозволяє підвищити ефективність управління навчальними закладами, забезпечити прозору взаємодію між учителями, учнями та батьками.

Об’єктом роботи є застосунок для управління шкільним закладом.

Метою роботи є розробка застосунку, який дозволяє адміністрації, вчителям, батькам та учням ефективно взаємодіяти в межах освітнього процесу. У рамках роботи створено оптимізовану базу даних для зберігання інформації про користувачів, навчальні предмети, розклади, відвідуваність, оцінки та інші дані.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати існуючі освітні системи, визначити їхні переваги та недоліки;

– сформувати вимоги до майбутньої системи з урахуванням результатів аналізу;

– реалізувати модуль реєстрації, автентифікації та розподілу ролей користувачів (адміністрація, вчителі, учні, батьки);

– створити інтерфейси для перегляду сторінок та редагування списків;

– реалізувати зберігання та обробку даних за допомогою бази даних;

– створити адаптивний, інтуїтивно зрозумілий інтерфейс із дотриманням сучасних стандартів UX/UI;

– здійснити тестування та демонстрацію функціоналу системи.

2 ВИКОРИСТАНІ ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОЗРОБКИ

2.1 Мови програмування

Розробка освітньої інформаційної системи – це складне й багатокомпонентне завдання, яке потребує грамотного підходу до вибору технологій та інструментів розробки. Важливо не лише забезпечити функціональність, а й створити рішення, яке буде зручним у використанні, простим у підтримці та здатним до масштабування. Тому вже на етапі проєктування було прийнято рішення про використання сучасних вебтехнологій, який охоплює компоненти для клієнтської логіки, серверної частини та системи зберігання даних.

До складу обраного технологічного рішення увійшли такі ключові мови та інструменти: CSS – мова стилів для оформлення й адаптивної верстки; JavaScript та TypeScript – мови для програмування логіки клієнта та сервера; а також PostgreSQL – система управління реляційними базами даних, що відповідає за збереження та обробку навчальної інформації. Кожна з цих технологій виконує окрему роль, і їх поєднання дозволило досягти цілісної, стабільної й функціонально повної системи.

Інтерфейс користувача системи розроблявся повністю у форматі TypeScript XML – розширенні мови TypeScript, яке дозволяє використовувати опис структури елементів інтерфейсу безпосередньо в коді [7]. Такий підхід поєднує логіку, представлення та поведінку елементів у межах одного компонента, що відповідає сучасним принципам компонентно-орієнтованої розробки вебзастосунків.

На відміну від класичного використання HTML-файлів, структура інтерфейсу описується у вигляді функціональних або класових компонентів із розміткою, що візуально нагадує HTML, але фактично є частиною TypeScript-коду. Це дозволяє створювати інтерфейси з високою гнучкістю,

глибокою типізацією та прямим контролем за передачею даних між компонентами.

Використання TypeScript XML у проєкті забезпечило:

– єдину мову програмування для фронтенду;

– гнучке компонування частин інтерфейсу з можливістю їх повторного використання;

– інтеграцію логіки керування інтерфейсом без окремих шаблонів або HTML-файлів.

Для того, щоб сторінка виглядала естетично, адаптувалась до розмірів екрана і зручно сприймалась користувачем, використовується CSS. Ця мова надає розробнику засоби для гнучкого керування зовнішнім виглядом усіх HTML-елементів. У реалізованій системі за допомогою CSS були створені стилі для таблиць, блоків розкладу, форм входу, панелей керування, повідомлень тощо. Крім того, особливу увагу приділено адаптивності: система повинна однаково добре працювати як на комп’ютерах, так і на мобільних пристроях, тому реалізовано гнучкі сітки, медіазапити та умовне форматування.

Щоб забезпечити динамічну поведінку системи, реакцію на дії користувача та взаємодію з сервером, було використано JavaScript – найпопулярнішу мову програмування у вебсередовищі (рис. 2.1). Її роль у проєкті полягає у створенні функцій, які дозволяють обробляти події (натискання кнопки, заповнення форми, зміна параметра), виконувати запити до серверної частини й оновлювати інтерфейс у режимі реального часу. Наприклад, під час перегляду розкладу класу користувач може обирати дату або день тижня, і JavaScript автоматично оновлює таблицю занять без повного перезавантаження сторінки. Завдяки цьому забезпечується плавний користувацький досвід і мінімізується затримка при взаємодії з даними.

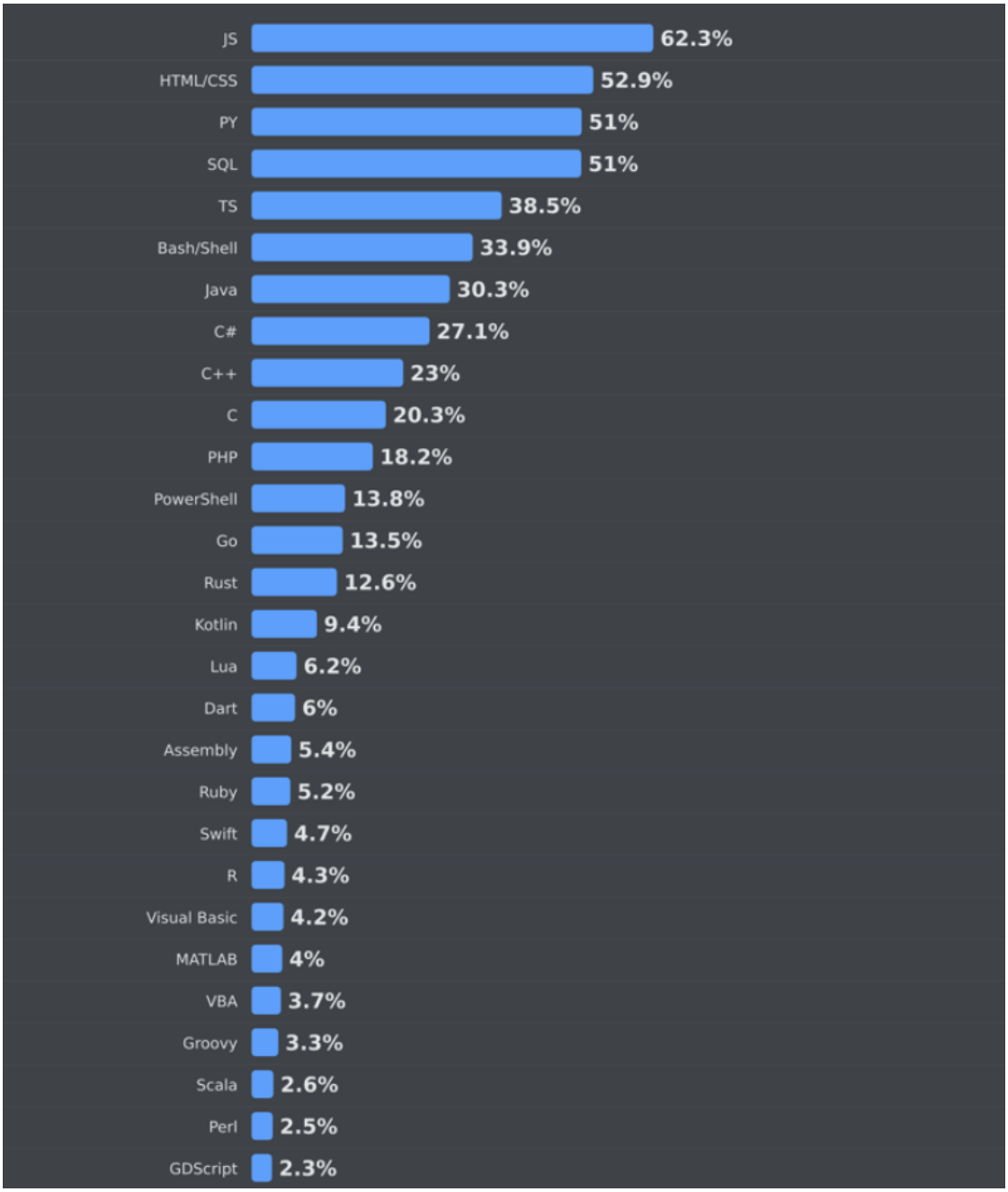


Рисунок 2.1 – Список найпопулярніших технологій для програмування, написання скриптів та розмітки

Однак для зручнішої та масштабованої роботи з кодом, особливо коли мова йде про великі проєкти, було вирішено використати TypeScript – надбудову над JavaScript, яка додає можливість статичної типізації. Це означає, що змінні, об’єкти, функції та інші елементи описуються з визначеними типами, які перевіряються ще до запуску програми. Завдяки цьому розробник може виявити й виправити помилки ще на етапі написання

коду. У рамках даного проєкту TypeScript застосовувався як для створення інтерфейсу користувача, так і для формування серверної логіки та обробки запитів. Особливо корисним виявилось спільне використання інтерфейсів для моделювання даних: один і той самий опис типів міг використовуватись і в коді клієнта, і на сервері, що дозволило уникнути дублювання та забезпечити точність передачі інформації між компонентами.

2.2 Архітектура застосунку

При створенні сучасних інформаційних систем, особливо тих, що орієнтовані на управління організаційними процесами у навчальних закладах, критично важливо з самого початку закласти коректну архітектуру програмного забезпечення. Від обраного архітектурного підходу безпосередньо залежать не лише технічні характеристики проєкту — такі як масштабованість, продуктивність і підтримуваність — але й швидкість розробки, гнучкість у впровадженні нових функцій, а також легкість адаптації до майбутніх змін у вимогах чи середовищі використання.

У межах розробки системи для управління шкільним закладом були сформульовані основні вимоги до архітектури:

– можливість швидкої побудови мінімально життєздатного продукту із подальшим масштабуванням функціоналу;

– інтеграція з зовнішніми сервісами, зокрема для автентифікації, зберігання медіа;

– простота супроводу та тестування;

– чітка модульність та логічне розділення обов’язків;

– гнучка база даних, яка дозволяє змінювати модель з мінімальними ризиками;

– підтримка сучасних технологій фронтенду та бекенду, із можливістю гібридного рендерингу сторінок.

Для реалізації цих вимог було обрано монолітну архітектуру з розділенням на логічні шари та використанням сучасних вебтехнологій. Застосунок створено з використанням фреймворку Next.js, який забезпечує одночасну роботу як на сервері, так і на клієнті, та є надбудовою над React. Такий підхід дозволяє уникнути складної побудови багатокомпонентної інфраструктури, зберігаючи при цьому можливість поділу логіки на ізольовані частини. У майбутньому це дозволить виділити окремі функціональні блоки в сервіси або мікросервіси при потребі.

Ключові компоненти архітектури:

– фронтенд реалізовано на React, який взаємодіє з серверними API;

– серверна частина функціонує на базі серверних можливостей Next.js. API-обробники відповідають за обробку HTTP-запитів, взаємодію з базою даних, валідацію та авторизацію;

– база даних – PostgreSQL, яка взаємодіє із застосунком;

– аутентифікація та авторизація реалізована через Clerk – сервіс, який забезпечує повноцінну роботу з користувачами, включаючи реєстрацію, вхід, захист маршрутів, рольову модель доступу;

– зберігання зображень здійснюється через інтеграцію з Cloudinary [10], що дозволяє завантажувати файли з форм та отримувати оптимізовані посилання для відображення.

Монолітна архітектура виявилася найбільш доцільною на етапі розробки першої версії застосунку. Це пояснюється такими перевагами:

– оперативна розробка – вся логіка доступна в єдиній кодовій базі, що дозволяє швидко переключатись між модулями та повторно використовувати спільні компоненти;

– зменшена інфраструктурна складність – не потребує налаштування окремих сервісів, брокерів повідомлень чи балансувальників навантаження;

– цілісність даних – робота з однією базою даних спрощує підтримку цілісності та узгодженості записів;

– узгоджене управління станом – React-контекст, форми, сповіщення та обробка помилок реалізовані в єдиній системі, що знижує ризик неузгоджених помилок.

Разом із тим, як і будь-який архітектурний підхід, моноліт має свої недоліки:

– зростання складності з часом – при додаванні нових модулів (наприклад, електронний щоденник, журнал оцінок, повідомлення, аналітика тощо) ускладнюється підтримка загального коду;

– проблеми масштабування – при високому навантаженні на окремі частини (наприклад, розклад або вхід користувачів) складно масштабувати їх вибірково;

– ризик перехресних залежностей – недотримання чіткої модульності може призвести до сплутування логіки.

Проте ці недоліки є типовими для будь-якого проекту, що інтенсивно зростає. На етапі реалізації застосунку для внутрішнього використання у школі вони не створюють критичних обмежень.

Закладена структура дозволяє в перспективі перейти до модульної або мікросервісної архітектури. Наприклад, такі функціональні блоки як «управління користувачами», «зберігання медіа», «аналітика відвідуваності» можуть бути виділені в окремі сервіси. Це стане можливим завдяки чіткому поділу відповідальностей у поточному проєкті: базова структура коду побудована на логічних модулях (сторінки, API, форми, компоненти, моделі), що спрощує рефакторинг і винесення функцій в окремі сервіси при потребі.

Побудова застосунку за принципами інтегрованої монолітної архітектури на основі сучасних інструментів (Next.js, Prisma, PostgreSQL, Clerk) дозволила реалізувати швидкий, зручний та технічно гнучкий вебпродукт для управління освітнім процесом. Така архітектура добре підходить для початкового розгортання та демонстрації основного функціоналу. У майбутньому вона може бути поступово трансформована в

більш децентралізовану, сервіс-орієнтовану систему, без необхідності радикальних змін у логіці.

2.3 Інструменти для створення застосунку

Під час розробки серверної частини вебзастосунку, який виконує функції системи управління шкільним закладом, надзвичайно важливим є вибір фреймворків, здатних забезпечити не лише швидкість розробки, а й надійність, безпеку, зручність масштабування та підтримку сучасних підходів до побудови вебінтерфейсів і взаємодії з базами даних. У межах цього проєкту серверна логіка реалізована з використанням Next.js – фреймворку на базі React із підтримкою серверного рендерингу, побудови API, динамічних маршрутів та інфраструктурної оптимізації (рис. 2.2). Додатково для збереження та обробки даних використано Prisma [8], що виступає сучасною ORM для роботи з PostgreSQL, а автентифікацію реалізовано через сервіс Clerk [9], який забезпечує надійний механізм керування користувачами.

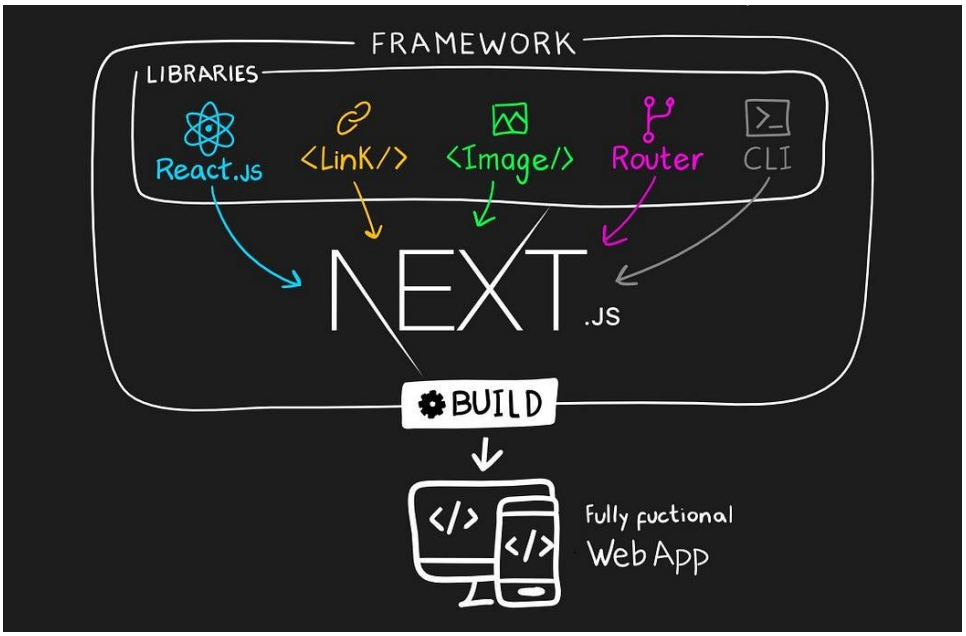


Рисунок 2.2 – Схема застосунку з використанням Next.js

На відміну від традиційних фреймворків, таких як Express або Koa, у даному проєкті серверна логіка тісно інтегрована з клієнтською через Next.js. Це сучасний фреймворк, який надає можливість створювати повноцінні застосунки з підтримкою SSR, ISR, API-роутів та повного контролю над маршрутизацією.

Next.js дозволяє реалізовувати серверну логіку через окремі API-ендпоїнти прямо всередині структури застосунку, без потреби створення окремого бекенд-сервера. Такий підхід значно спрощує архітектуру системи, дозволяє швидше налагодити інтеграцію з клієнтською частиною, а також легко розгортати проєкт на таких платформах як Vercel чи Railway.

Серверна логіка в цьому застосунку охоплює:

– Обробку запитів на створення, оновлення та видалення студентів, класів, предметів;

– Взаємодію з базою даних через Prisma;

– Інтеграцію з Cloudinary для завантаження та зберігання зображень;

– Перевірку автентичності користувача через Clerk API;

– Обробку помилок і відповіді у форматі JSON.

Таким чином, Next.js виконує не лише роль фреймворка для інтерфейсу, а й ефективно замінює класичну серверну платформу.

Взаємодія із базою даних PostgreSQL у проєкті реалізована через Prisma ORM. Це сучасний інструмент, який дозволяє працювати з даними типобезпечно та ефективно, використовуючи власну декларативну мову для опису моделей та зв’язків між ними.

Prisma генерує типізований клієнт для доступу до бази даних, що спрощує розробку й зменшує кількість помилок під час виконання запитів. Окрім CRUD-операцій, Prisma також підтримує фільтрацію, сортування, зв’язки між таблицями, транзакції та міграції бази даних.

Крім того, Prisma інтегрується з Next.js безпосередньо у серверних обробниках запитів, дозволяючи централізовано керувати запитами до бази даних та виконувати логіку бізнес-рівня в одному середовищі.

Для реалізації системи автентифікації в проєкті використано сервіс Clerk, який надає повноцінну інфраструктуру для обліку користувачів. Clerk дозволяє швидко реалізувати:

- реєстрацію та вхід користувачів;
- захист маршрутів;
- доступ до JWT-токенів;
- витяг профілю користувача на сервері або клієнті;
- керування сесіями.

Інтеграція Clerk із Next.js здійснюється як на клієнтському рівні, так і на серверному, що дозволяє обмежувати доступ до API-ендпоїнтів, забезпечуючи безпечну роботу з даними.

У контексті системи керування шкільним закладом Clerk відіграє ключову роль в ідентифікації адміністрації закладу та забезпеченні захищеного доступу до інтерфейсів додавання, редагування чи видалення учнів, викладачів та інших сутностей.

У межах реалізації проєкту було також використано низку допоміжних інструментів, які істотно вплинули на зручність розробки, якість користувацького досвіду та підтримку чистоти коду. Одним із таких є react-hook-form – бібліотека, яка забезпечує ефективне управління станом форм у React-додатку [11]. Вона дозволила значно скоротити обсяг повторюваного коду при створенні форм, таких як форма додавання або редагування інформації про учнів. Завдяки реактивній системі обробки введених даних, ця бібліотека забезпечує високу продуктивність навіть при складних і багатоступеневих формах.

Для валідації даних, що надходять з форм, використано бібліотеку zod [12], яка інтегрується з react-hook-form та дозволяє описувати схеми перевірки даних у декларативному стилі. Це забезпечує типобезпечність, контроль на стороні клієнта й сервера, а також уніфіковану логіку перевірки, що особливо важливо для забезпечення цілісності інформації у системі управління школою.

У частині завантаження файлів, зокрема фотографій учнів, була застосована бібліотека @uploadthing/react [13], яка значно спрощує реалізацію інтерфейсів для передачі файлів у зв’язці з Cloudinary. Вона дозволяє не лише швидко реалізувати логіку drag-and-drop, а й інтегрується з серверними обробниками, забезпечуючи безпечну передачу файлів та обмеження за типом і розміром. Це виявилося особливо корисним при роботі зі шкільною базою фотографій, де важливо не тільки отримати файл, але й обробити його відповідно до встановлених вимог до зображень.

Ще одним важливим елементом взаємодії з користувачем стала бібліотека react-hot-toast [14], яка використовується для виводу повідомлень про статус виконання дій. У випадку успішного створення, оновлення чи видалення запису, а також при виникненні помилок (наприклад, при недопустимому форматі зображення чи помилці валідації), система одразу інформує користувача за допомогою спливаючих повідомлень. Це сприяє прозорій взаємодії з інтерфейсом і зменшує кількість непорозумінь під час користування системою.

Усі вищезгадані інструменти органічно доповнюють основні фреймворки проєкту, формуючи з ними єдине середовище розробки. Завдяки цьому вдалось реалізувати не лише базову функціональність, але й створити інтуїтивно зрозумілий, безпечний та адаптивний вебзастосунок, що відповідає сучасним вимогам до систем адміністративного управління у сфері освіти.

2.4 База даних

У сучасну епоху цифрової трансформації та стрімкого розвитку інформаційних технологій зберігання, обробка та ефективне управління

даними стали одними з головних завдань для будь-якої інформаційної системи. Практично кожна сучасна програма, вебзастосунок чи мобільний сервіс взаємодіє з певною базою даних. Це стосується як невеликих локальних проєктів, так і масштабних корпоративних систем. В умовах збільшення обсягів інформації, що обробляється щодня, бази даних забезпечують упорядковане зберігання, високу швидкодію, безпечний доступ і можливість масштабування застосунків.

База даних – це організований набір структурованих відомостей, які зберігаються в електронному вигляді та керуються системою керування базами даних (СКБД). Вона дозволяє легко додавати, змінювати, видаляти та шукати необхідні дані, а також гарантує їх цілісність, консистентність і збереження при виникненні помилок або збоїв у системі. У світі СКБД існує велика кількість рішень, які орієнтовані на різні моделі зберігання даних, але серед них особливе місце займають реляційні системи.

Однією з найпотужніших і найгнучкіших реляційних СКБД, яка користується широким попитом серед розробників та аналітиків даних, є PostgreSQL. Це вільна об’єктно-реляційна система з відкритим вихідним кодом, яка поєднує класичні принципи реляційної моделі з можливістю зберігання складних об’єктних структур. Розробка PostgreSQL почалася ще у 1986 році в Каліфорнійському університеті Берклі, а з 1996 року проєкт розвивається як повноцінна СКБД з активною міжнародною спільнотою.

Однією з головних переваг PostgreSQL є розширюваність та гнучкість архітектури. Це означає, що користувачі можуть створювати власні функції, типи даних, агрегатні функції, оператори та навіть мови запитів. Завдяки цьому PostgreSQL адаптується під будь-які вимоги, починаючи від простих CRUD-операцій у вебдодатках, і завершуючи побудовою складних аналітичних систем із великою кількістю взаємозв’язаних таблиць, процедур і тригерів.

Система підтримує різні типи даних, включаючи стандартні (рядки, числа, дати), складні (масиви, множини), та сучасні формати, зокрема JSON і

JSONB. Останній формат особливо корисний для зберігання напівструктурованих даних, які притаманні сучасним вебзастосункам. У поєднанні з потужною мовою запитів SQL PostgreSQL дозволяє виконувати гнучкі фільтрації, агрегації, обробку вкладених структур та перетворення даних безпосередньо на рівні бази.

Підтримка транзакційності на основі ACID-принципів гарантує, що всі операції з даними виконуються коректно та надійно, навіть у випадках збоїв, обривів з'єднання або паралельного доступу з багатьох джерел. PostgreSQL використовує мультиверсійний контроль конкурентності, що дозволяє уникнути блокувань між транзакціями, підвищуючи продуктивність і забезпечуючи кращу масштабованість для багатокористувацьких систем.

PostgreSQL також має вбудовані засоби реплікації (як синхронної, так і асинхронної), що дозволяють створювати резервні копії баз у реальному часі, організовувати відмовостійкі кластери та підвищувати надійність системи. У великих застосунках застосовуються рішення для горизонтального масштабування, зокрема Citus, який розширює можливості PostgreSQL шляхом розподілу даних по кількох вузлах, перетворюючи її на розподілену базу даних.

Ще одним важливим компонентом є безпека. PostgreSQL пропонує широкі можливості з контролю доступу до об'єктів бази даних, підтримує аутентифікацію через SSL, Kerberos, LDAP, GSSAPI, SCRAM-SHA-256 та інші механізми. Адміністратори можуть налаштовувати права доступу на рівні окремих таблиць, стовпців, функцій, а також визначати політики доступу за допомогою системи row-level security.

Не менш важливою є інтеграція PostgreSQL з сучасними мовами програмування і фреймворками. Завдяки великій кількості клієнтських бібліотек ця система ідеально підходить для проєктів на будь-яких технологіях. Також PostgreSQL підтримується великою кількістю ORM бібліотек – наприклад, Prisma, TypeORM, Sequelize, Django ORM, що

дозволяє розробникам працювати з базою даних на високому рівні абстракції, не вдаючись до написання SQL-запитів вручну.

У поєднанні з відкритим кодом, регулярними оновленнями, активною підтримкою спільноти та великою кількістю готових розширень (наприклад, для повнотекстового пошуку, роботи з геоданими, аналітики, машинного навчання тощо), PostgreSQL є універсальним вибором як для традиційних реляційних завдань, так і для сучасних сценаріїв, де потрібна висока продуктивність, масштабованість та надійність.

PostgreSQL вирізняється суворою відповідністю ACID-вимогам, гнучкою блокувальною моделлю MVCC та багатою екосистемою розширень, що дає змогу одночасно обслуговувати великий обсяг транзакцій і виконувати складну аналітику без зовнішніх модулів. На відміну від нього, MySQL традиційно фокусується на швидкому OLTP — завдяки простішій архітектурі воно демонструє нижчі накладні витрати, однак за складних багатокористувацьких сценаріїв відчувається обмеження гранулярності блокувань і менш гнучкий механізм тригерів та перевірок цілісності. MariaDB, як форк MySQL, успадковує його переваги швидкого читання й легкості розгортання, але додає власні рушії зберігання, наприклад Aria та ColumnStore, і відкритішу модель розвитку спільнотою; попри це, сумісність інтерфейсів лишає систему прив’язаною до історичних компромісів MySQL. MongoDB представляє документно-орієнтований підхід із горизонтальною масштабованістю завдяки шардованню та гнучкою схемою BSON-документів, що спрощує швидкі зміни структури даних, проте вносить додаткові складності у транзакційну цілісність, побудову складних відношень і забезпечення строгих зовнішніх ключів.

Для панелі керування школою потрібні суворі гарантії збереження оцінок, розкладів і фінансових записів, а також можливість легко моделювати складні зв’язки «учень – клас – предмет – викладач». Реляційна природа PostgreSQL з повноцінними зовнішніми ключами, перевірками та каскадними діями гарантує цілісність цих зв’язків; одночасно розширена

підтримка JSONB дозволяє зберігати напівструктуровані дані, наприклад індивідуальні налаштування користувачів, без відмови від транзакцій. Завдяки плагінам логічної реплікації та вбудованим механізмам шифрування PostgreSQL спрощує реалізацію резервування й відповідність вимогам захисту персональних даних, що критично для освітньої сфери. Таким чином, у порівнянні з MySQL та MariaDB, де частина функціоналу реалізується поза ядром або має меншу гнучкість, та з MongoDB, яке робить акцент на швидкі зміни схеми ціною складності транзакцій, PostgreSQL пропонує найкращий баланс надійності, виразності моделі даних і можливості розширення, необхідний для довготривалого проекту управління школою [15].

Таким чином, PostgreSQL виступає не лише як інструмент для зберігання даних, а як цілісна екосистема для створення масштабованих, безпечних і гнучких інформаційних рішень, здатних ефективно працювати в умовах високих навантажень і динамічного зростання вимог до обробки інформації. Завдяки своїм характеристикам, вона посідає одне з провідних місць серед усіх СКБД, поступаючись лише дорогим комерційним рішенням і навіть у деяких випадках перевершуючи їх за гнучкістю та функціональністю. Саме тому PostgreSQL є одним із найкращих варіантів для створення сучасних вебзастосунків, корпоративних платформ та масштабованих онлайн-сервісів.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ ПАНЕЛІ ДЛЯ ШКІЛЬНОГО ЗАКЛАДУ

3.1 Обґрунтування вибору середовища програмної реалізації

Розробка сучасного вебзастосунку для управління освітнім закладом вимагає ретельного підбору технологій, які забезпечують не лише функціональність і продуктивність, але й гнучкість, масштабованість та простоту подальшого супроводу. На етапі планування програмного забезпечення основною метою було побудувати універсальний, надійний, безпечний і водночас зручний у користуванні застосунок, здатний працювати з великим обсягом даних, підтримувати різні ролі користувачів (учень, вчитель, батьки, адміністратор) та забезпечувати високу доступність через браузер.

Одним із ключових чинників успішної реалізації цього завдання є правильний вибір середовища розробки. У даному проєкті було вирішено застосувати Visual Studio Code – один із найпопулярніших та найпотужніших редакторів коду [16], що забезпечує гнучку екосистему розширень, підтримку TypeScript, інтеграцію з Git, налагодження коду, автодоповнення та миттєву перевірку помилок. Завдяки плагінам, таким як TailwindCSS IntelliSense, ESLint, Prettier, Prisma, Clerk та ін., розробка стала максимально ефективною та комфортною. VS Code також добре адаптований до роботи в команді, дозволяє швидко перемикатись між проектами та легко інтегрується з GitHub або іншими системами контролю версій.

Для реалізації клієнтської частини застосунку обрано Next.js – фреймворк для React, який надає розширені можливості для побудови серверно-рендерених додатків, генерації статичних сторінок, використання гібридного рендерингу та оптимізації ресурсів. Остання версія Next.js включає підтримку App Router, що дозволяє реалізовувати маршрутизацію на

основі вкладеної файлової структури, використовуючи сучасні можливості React, зокрема серверні компоненти та асинхронні компоненти.

React, як основна бібліотека для побудови інтерфейсу користувача, був обраний завдяки своїй гнучкості, поширеності та активній підтримці спільноти. Його компонентна архітектура дозволяє легко масштабувати застосунок, розділяючи інтерфейс на незалежні частини, які можна перевикористовувати. У поєднанні з TypeScript це дозволило зменшити кількість помилок на етапі компіляції, краще документувати структуру компонентів і підвищити стабільність програми.

Для стилізації інтерфейсу було використано TailwindCSS – CSS-фреймворк, що працює на утилітарному підході. Tailwind дозволив гнучко та швидко верстати адаптивний інтерфейс без потреби у створенні великих CSS-файлів. Завдяки наявності змінних тем, розширень для анімації та підтримки темної теми, Tailwind значно покращив візуальне сприйняття інтерфейсу без втрати продуктивності.

Обробка форм реалізована через React Hook Form у поєднанні з Zod – це дало можливість контролювати стани форм, проводити надійну типізовану валідацію введених даних на клієнтському рівні та швидко реагувати на помилки. Також ця пара інтегрується з серверною логікою, що дозволило реалізувати повторне використання схем валідації.

Компонентна бібліотека Radix UI надала базові інтерфейсні елементи, які відповідають вимогам доступності та легко стилізуються через TailwindCSS. З її допомогою реалізовано випадаючі меню, модальні вікна, таби та інші компоненти з передбачуваною поведінкою.

Для клієнт-серверної взаємодії застосовано axios – зручну бібліотеку для HTTP-запитів, яка підтримує інтерсептори, обробку помилок, конфігурацію заголовків, зокрема передачу токенів авторизації. Це забезпечило стабільну та передбачувану роботу з API застосунку.

Окрема увага була приділена організації автентифікації та керування користувачами. Для цього використано Clerk – хмарне рішення, яке дозволяє

реалізувати реєстрацію, вхід, авторизацію, підтримку соціальних провайдерів, захист сесій та ролей. Clerk інтегрується з Next.js, надає UI-компоненти та REST API для повного контролю над життєвим циклом користувача. Це дозволило уникнути необхідності самостійного написання системи автентифікації та сконцентруватися на бізнес-логіці застосунку.

На стороні бекенду використано Prisma ORM для взаємодії з базою даних PostgreSQL. Prisma автоматично генерує типізовані клієнти, дозволяє створювати та відслідковувати міграції, легко масштабувати схеми та інтегрується з TypeScript. Це зменшило кількість помилок при роботі з базою, спростило налагодження та пришвидшило розробку.

Для зберігання зображень (наприклад, фотографій учнів) було впроваджено інтеграцію з Cloudinary – хмарним сервісом для обробки мультимедіа, який підтримує завантаження, зміну розміру, оптимізацію та захищений доступ до зображень. Через підписані запити забезпечено безпечне завантаження фото без зловживання ресурсами системи.

Додаткові бібліотеки, що були використані:

– clsx – для зручного керування класами в JSX;

– lucide-react – набір іконок з підтримкою кастомізації;

– recharts – бібліотека для побудови графіків і візуалізації статистичних даних (наприклад, відвідуваності, оцінювання тощо).

Вся система побудована за принципами чистої архітектури з чітким поділом на компоненти, що відповідають за логіку, валідацію, відображення даних і зберігання. Усі частини застосунку взаємодіють через чітко визначені API та мають підтримку типів, що забезпечує злагоджену роботу клієнта й сервера.

Обраний технологічний стек дозволив створити гнучкий, модульний, безпечний та масштабований застосунок, який легко розширюється й підтримується. Всі інструменти підтримують форматування коду, тести й мають відповідну документацію.

Такий підхід забезпечив технічну надійність системи, зручність для кінцевих користувачів та відкриває можливість для подальшого вдосконалення проєкту, включаючи адаптацію під мобільні пристрої, інтеграцію з іншими сервісами та аналітику користувацьких дій.

3.2 Програмна реалізація

Розробка сучасного вебзастосунку для управління освітнім процесом є складним і багатоступеневим завданням, що охоплює різноманітні аспекти – від створення архітектури до розгортання продукту. Щоб досягти ефективної, масштабованої та безпечної системи, яка відповідатиме реальним потребам навчального закладу, важливо ретельно спланувати й реалізувати кожен етап програмного циклу.

Були сформовані ключові етапи розробки програмного забезпечення. Кожен із них був спрямований на побудову стабільної, гнучкої та зручної у використанні системи, що охоплює керування учнями, розкладом, класами, викладачами та адміністративним персоналом.

До основних етапів реалізації належали:

– проєктування загальної архітектури системи, що визначає її структуру, взаємодію компонентів і загальну логіку роботи;

– розробка клієнтської частини із застосуванням сучасних технологій, зокрема React, Next.js, TailwindCSS;

– побудова серверної логіки з використанням Prisma для взаємодії з базою даних, та обробки запитів через API;

– інтеграція автентифікації та безпеки на основі платформи Clerk;

– впровадження форм, валідації та обробки файлів через react-hook-form, Zod, Cloudinary;

– тестування функціональності і оптимізація клієнтської та серверної частин;

– розгортання з урахуванням майбутнього масштабування.

3.2.1 Проєктування архітектури застосунку

Важливим початковим кроком у реалізації будь-якого повнофункціонального застосунку є створення правильної архітектури, яка не лише задовольнить поточні потреби користувачів, а й дозволить у майбутньому легко додавати нову функціональність, масштабувати систему та підтримувати її стабільність у виробничому середовищі.

Було обрано сучасну клієнт-серверну архітектуру, яка забезпечує розподілення відповідальностей між фронтендом та бекендом. Такий підхід дозволяє:

– ізолювати бізнес-логіку від інтерфейсу;

– покращити безпеку за рахунок обробки конфіденційної інформації виключно на сервері;

– оптимізувати продуктивність кожного шару окремо;

– підтримувати масштабування інфраструктури без повного переписування коду.

Архітектура системи базується на принципі розділення відповідальностей. Це означає, що кожен функціональний блок системи – автентифікація, управління учнями, завантаження фотографій, редагування профілів тощо – винесений в окремі модулі. Це полегшує розробку, підтримку та повторне використання коду.

На рівні даних використовувалась реляційна база PostgreSQL, з якою взаємодіє Prisma – ORM-інструмент для TypeScript. Модель даних побудована так, щоб відображати реальну структуру освітнього процесу: класи, учні, вчителі, ролі, дисципліни тощо.

Узаємодія між клієнтською частиною та сервером відбувається через HTTP-запити. RESTful API реалізовані через маршрути в Next.js API Route-

обробниках, із підтримкою типів, обробки помилок та валідації даних за допомогою бібліотеки Zod. Взаємодія між компонентами схематично зображена на рисунку 3.1

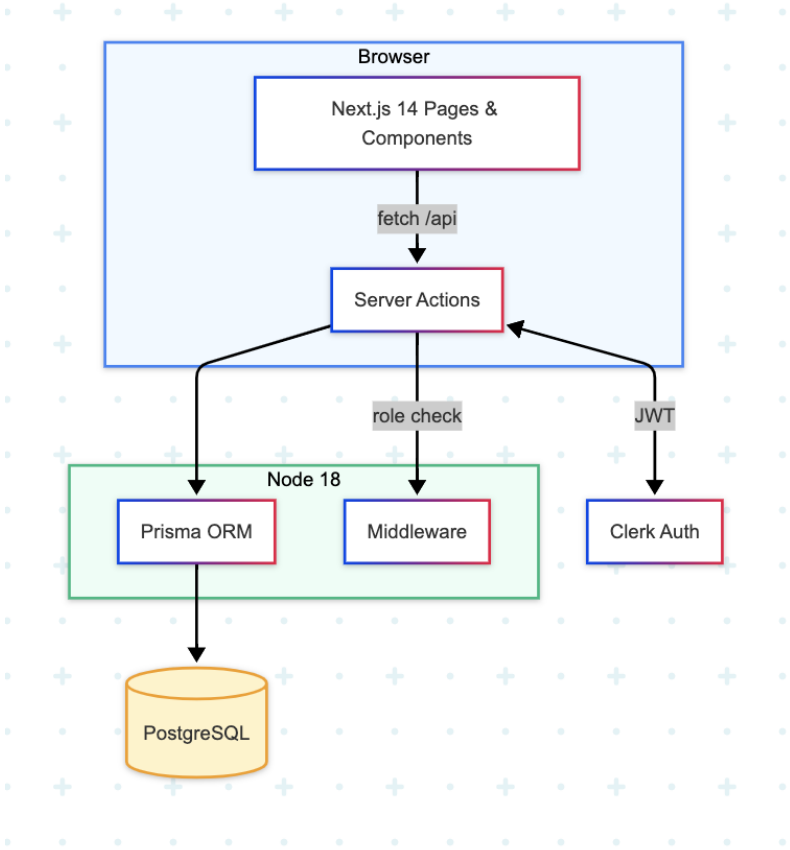


Рисунок 3.1 – Діаграма архітектурної схеми застосунку

Ключовими характеристиками архітектури є:

– модульність. Компоненти чітко розділені за призначенням: компоненти для роботи з формами, API, валідацією, інтерфейсом тощо. Це спрощує тестування і підтримку;

– гнучкість і масштабованість. Завдяки динамічному маршрутизатору Next.js та Prisma ORM, система легко адаптується під зміну вимог: додавання нових таблиць, ролей, сторінок або API-ендпоінтів;

– безпечна автентифікація. Clerk забезпечує повноцінну автентифікацію користувачів з підтримкою ролей, обмеження доступу до маршрутів, токенів і OAuth;

– оптимізація запитів. Prisma дозволяє писати складні запити до бази даних з використанням автогенерованих типів і зменшує кількість помилок завдяки типобезпечності;

– продуктивність. Завдяки SSR, кешуванню, оптимізації рендерингу та вибірковому завантаженню ресурсів інтерфейс лишається швидким навіть при великій кількості користувачів.

Вся комунікація між компонентами виконується через захищені канали (HTTPS), а дані перевіряються на рівні як клієнта, так і сервера.

Для кращого розуміння наведено приклади запитів у таблиці 3.1, які відправляються між частинами системи.

Таблиця 3.1 – HTTP-запити

Дія	**Користувач**	**Запит**	**Результат**
Додавання учня	Адміністратор	POST /api/students	Додавання до бази учня
Редагування профілю	Вчитель	PATCH /api/students/:id	Оновлення даних учня
Завантаження фото	Учень	POST /api/upload-image → Cloudinary	Отримання URL зображення
Авторизація	Будь-який	Clerk API	Отримання токена, login state
Перевірка доступу	Сервер	Middleware /api/*	Перевірка ролі та прав доступу

Таким чином реалізовано з урахуванням найкращих практик веброзробки – як з погляду структури, так і з боку практичної реалізації. Обрана архітектура дає змогу розвивати систему в майбутньому без радикальних змін, додаючи нові модулі, покращуючи UX або інтегруючи сторонні сервіси без шкоди для існуючої структури.

3.2.2 Проєктування структури бази даних

Проєктування структури бази даних є одним із ключових етапів у створенні інформаційної системи, що забезпечує стабільне, ефективне та логічне зберігання й обробку всіх необхідних даних. Для застосунку, який охоплює управління освітнім процесом – включно з обліком учнів, викладачів, класів, предметів, розкладу, документації та ролей користувачів – критично важливо мати добре продуману модель даних.

Для застосунку було використано реляційну базу даних PostgreSQL, яка забезпечує високу продуктивність, надійність, транзакційність і підтримку складних запитів. Роботу з базою даних реалізовано за допомогою ORM Prisma, яка дозволяє використовувати типобезпечні запити, автоматичну генерацію міграцій, а також синхронізацію схеми з кодом.

Під час проєктування структури даних було дотримано принципів нормалізації для усунення надлишковості інформації, забезпечення узгодженості та цілісності. Було змодельовано основні сутності, що відображають реальні об’єкти предметної області навчального закладу. Усі сутності мають чіткі зв’язки між собою – як один-до-багатьох, так і багато-до-багатьох, де це необхідно.

Основні таблиці, реалізовані в системі:

– User – базова таблиця користувачів, яка містить інформацію про обліковий запис: унікальний ідентифікатор, електронну пошту, ім’я, прізвище, роль (адміністратор, викладач, учень), зображення профілю, дату реєстрації. Ролі визначають рівень доступу до функцій системи;

– Student – таблиця, що зберігає розширену інформацію про учнів: дата народження, стать, адреса проживання, контактні дані батьків, клас, фотографія. Зв’язана з таблицею User (через зовнішній ключ) та Class;

– Teacher – модель для викладачів, що містить посаду, предмети, стаж, контактні дані. Також пов’язана з User та має зв’язок із предметами, які викладає;

– Class – структура класів (1-А, 2-Б тощо), де вказано назву класу, кількість учнів, класного керівника. Зв’язки: один клас має багато учнів (Student), і одного відповідального вчителя (Teacher);

– Subject – таблиця предметів (математика, біологія тощо). Кожен предмет може викладатися у різних класах і різними викладачами, тому існує додаткова таблиця-зв’язка;

– Schedule – розклад уроків: клас, предмет, день тижня, час початку та завершення уроку, відповідальний вчитель. Це дозволяє зберігати структурований розклад для кожного класу;

– Grade – модель для зберігання оцінок: учень, предмет, оцінка, дата, коментар. Дає змогу створювати журнал оцінювання;

– Attendance – відвідуваність учнів: дата, тип відсутності, коментар, пов’язана з учнем і класом;

– ParentContact – контактна інформація батьків або опікунів, прив’язана до Student. Це полегшує комунікацію з родичами учня у разі потреби;

– Document – навчальні або адміністративні документи (довідки, заяви, плани тощо), які можуть бути прив’язані до конкретного учня, класу чи викладача;

– ImageUpload – таблиця для зберігання посилань на зображення учнів, які завантажуються на зовнішній сервіс (Cloudinary). Зв’язана з Student.

Приклад основних зв’язків між сутностями:

– один клас має багато учнів, але кожен учень належить лише одному класу;

– один викладач може викладати кілька предметів, і кожен предмет може викладатись різними викладачами (багато-до-багатьох);

– кожен учень може мати багато оцінок, які пов’язані з конкретним предметом і викладачем;

– кожен розклад містить інформацію про клас, предмет, учителя, час;

– батьківський контакт пов’язаний із конкретним учнем (один-до-одного).

Візуальна структура таблиць та зв’язків представлена на рисунку 3.2

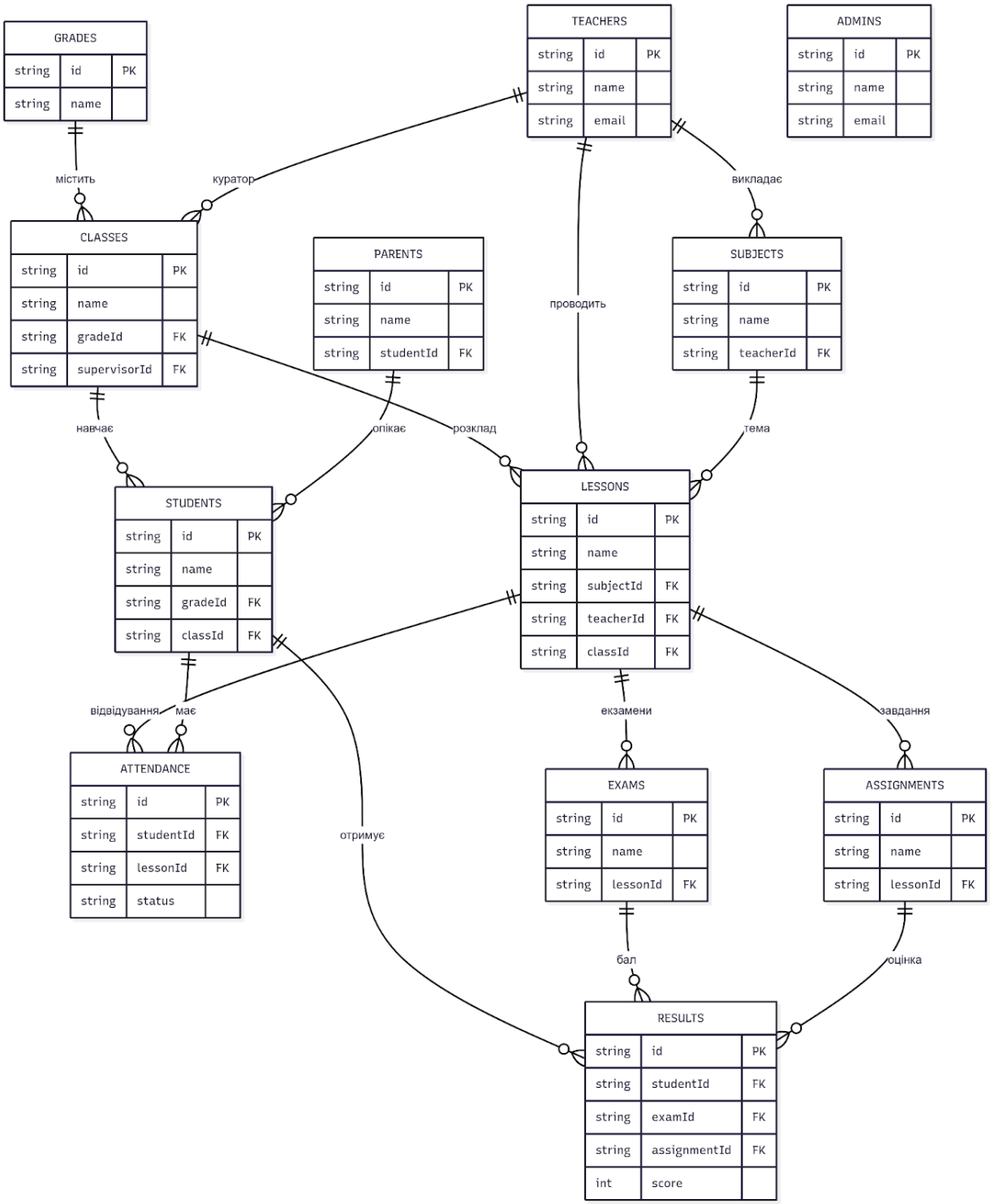


Рисунок 3.2 – Схематичне зображення структури бази даних застосунку

Проєктована база даних охоплює всі функціональні потреби системи: вона підтримує динамічні зміни, легко масштабується, дозволяє гнучко фільтрувати й сортувати інформацію, а також гарантує узгодженість даних завдяки використанню зовнішніх ключів, унікальних обмежень і типізації.

Також, завдяки інтеграції з Prisma, модель бази даних повністю відображена в коді – що дозволяє швидко генерувати запити, створювати міграції при зміні схеми та уникати помилок, пов’язаних з ручною роботою з SQL.

Таким чином, база даних стала надійною та гнучкою основою для всієї серверної логіки застосунку. Вона забезпечує правильне зберігання всіх сутностей освітнього процесу та сприяє ефективній реалізації бізнес-логіки системи.

3.2.3 Реалізація серверної частини

Серверна частина застосунку виступає ключовою ланкою між інтерфейсом користувача та базою даних, забезпечуючи обробку запитів, управління автентифікацією та виконання бізнес-логіки. Її головне завдання – гарантувати стабільну, безпечну й масштабовану роботу всієї системи.

Сервер реалізовано з використанням Next.js API Routes, що дозволило поєднати фронтенд і бекенд в одному середовищі. Це значно спрощує взаємодію між частинами застосунку та дозволяє обійтися без окремого сервера. Усі запити обробляються через API-ендпоїнти, які взаємодіють із базою даних за допомогою Prisma ORM.

Prisma використовується як основний інструмент для доступу до бази PostgreSQL. Його генерація типів на основі схеми бази дозволяє уникнути помилок типізації, пришвидшити розробку та гарантує передбачуваність взаємодії з даними. Кожна сутність – учень, клас, вчитель, розклад,

відвідуваність – має відповідну модель у схемі Prisma, що дозволяє ефективно реалізовувати CRUD-функціональність.

Аутентифікацію реалізовано через Clerk – сучасний сервіс керування користувачами. Clerk забезпечує повний спектр функцій: реєстрацію, вхід, керування сесіями та обмеження доступу до захищених маршрутів. На сервері реалізовано механізм перевірки токенів, що додаються до кожного запиту, – це дозволяє впевнено ідентифікувати користувача та контролювати його права доступу.

Дані, що надходять до API, проходять попередню перевірку: валідація виконана за допомогою бібліотеки Zod. Це забезпечує чітке визначення очікуваної структури об’єктів, зменшує кількість помилок при обробці запитів та унеможливлює обробку некоректних даних.

Загальна структура API дотримується REST-підходу.

- GET /api/students – отримання списку учнів;
- POST /api/students – створення нового учня;
- PATCH /api/students/:id – редагування даних;
- DELETE /api/students/:id – видалення з бази.

Усі серверні обробники мають логіку обробки помилок і відповідають з відповідними HTTP-статусами, що дозволяє клієнтській частині адекватно реагувати на події. Передбачено централізовану обробку помилок, зокрема для проблем доступу або неправильного введення.

Реалізація серверної частини побудована з урахуванням принципів розділення відповідальності, що дозволяє легко масштабувати та розширювати функціональність у майбутньому.

3.2.4 Реалізація клієнтської частини

Інтерфейс користувача є найбільш видимим компонентом системи, що забезпечує керування основними функціями: реєстрація учнів, перегляд

розкладу, адміністрування класів, управління відвідуваністю тощо. При розробці інтерфейсу основну увагу зосереджено на інтуїтивності, адаптивності та стабільності взаємодії.

Клієнт реалізовано на базі React у поєднанні з Next.js, що дозволило забезпечити серверний рендеринг, оптимізовану маршрутизацію та покращену продуктивність. TypeScript забезпечує типобезпеку в усіх компонентах, зменшуючи ризик помилок.

Для стилізації використано TailwindCSS, що дозволяє швидко створювати адаптивні компоненти та підтримувати єдиний стиль інтерфейсу. Анімації реалізовано за допомогою плагіну tailwindcss-animate, що покращує сприйняття динамічних елементів – наприклад, при додаванні чи видаленні учнів або повідомлень.

Компонентний підхід забезпечує незалежність кожної функціональної частини: сторінки створення, редагування, перегляду учнів, а також форми входу, профілю, списку класів і розкладів розділено на логічні блоки з окремими відповідальностями.

Управління формами виконано через бібліотеку React Hook Form, яка дозволяє уникнути зайвих ререндерів і спрощує роботу з валідацією. Перевірка введених даних реалізована через Zod – на кожне поле застосовується типобезпечна схема.

Маршрутизація заснована на файловій системі Next.js. Захищені сторінки перевіряють статус автентифікації через Clerk, автоматично обмежуючи доступ для неавторизованих користувачів.

Виклики API організовано через окремий модуль services, у якому кожна бізнес-сутність має свій файл із методами для взаємодії з сервером. Axios виступає головним HTTP-клієнтом, а інтеграція з TanStack React Query забезпечує автоматичне кешування, оновлення даних і повторну синхронізацію.

Для глобального стану використовується Redux Toolkit – його інтеграція з redux-persist дозволяє зберігати обрані дані між сесіями

користувача (наприклад, останній переглянутий клас або фільтр по учнях) [18].

Інтерфейс також включає додаткові інтерактивні можливості:

– drag-and-drop через @dnd-kit – для сортування чи зміни порядку класів;

– react-hot-toast – для інформування про успішні чи помилкові дії;

– ucide-react – для графічних позначень;

– react-colorful – вибір кольорів для профілю;

– cmdk – реалізація швидкого пошуку та навігації.

Весь інтерфейс адаптовано під мобільні пристрої, враховано доступність.

3.3 Тестування застосунку

Одним із ключових елементів у системі управління шкільним закладом є головна сторінка, яка виступає в ролі інформаційної панелі для користувача. Вона надає швидкий доступ до основних модулів системи та відображає зведену інформацію про поточний стан закладу. Інтерфейс реалізовано таким чином, щоб адміністратори, викладачі та інші користувачі могли зручно взаємодіяти з системою, оперативно отримувати актуальні дані про навчальний процес, відвідуваність, фінансові показники та майбутні події. Інтерфейс реалізований з урахуванням принципів зручності, наочності та сучасного дизайну (рис. 3.3).

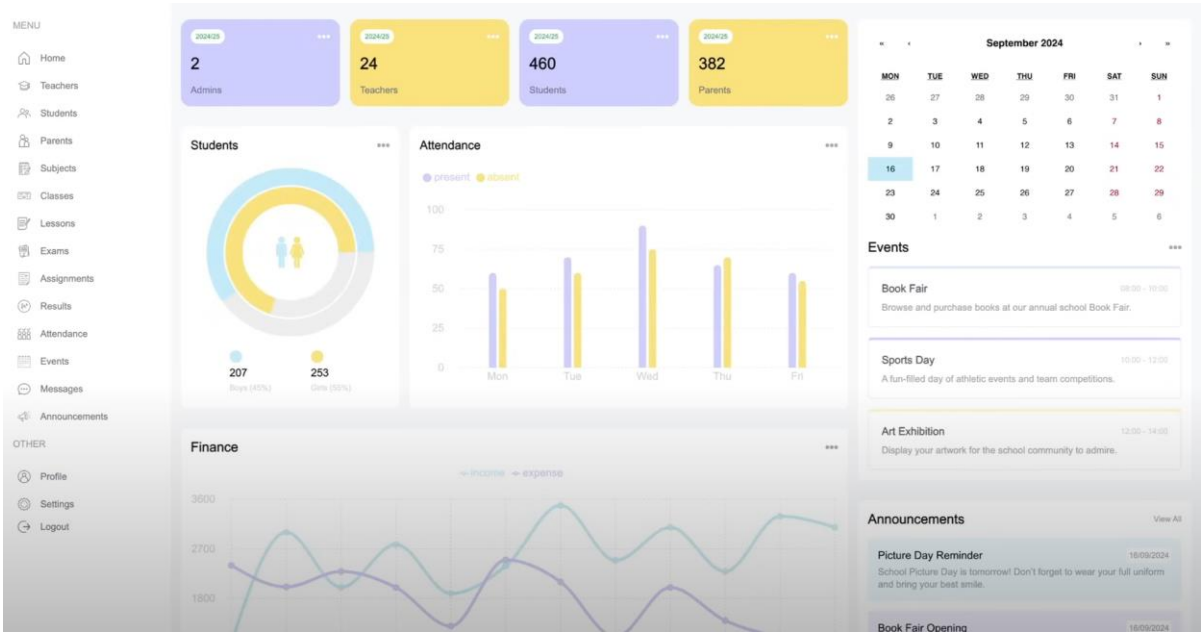


Рисунок 3.3 – Головна сторінка

У лівій частині сторінки розташовано вертикальне меню, що містить пункти навігації до всіх основних модулів: Home, Teachers, Students, Parents, Subjects, Classes, Lessons, Exams, Assignments, Results, Attendance, Events, Messages, Announcements, а також розділ Other з пунктами Profile, Settings та Logout. Іконки поруч із назвами сприяють швидкій ідентифікації функціоналу.

У верхній частині робочої області відображаються статистичні картки з основною інформацією: кількість адміністраторів, вчителів, учнів та батьків. Кожна картка має кольорове кодування, що дозволяє легко сприймати дані візуально.

Нижче розташовані візуалізовані аналітичні дані:

– блок Students відображає кількість хлопців і дівчат у вигляді кругової діаграми;

– блок Attendance містить гістограму присутності та відсутності учнів за днями тижня, де використано кольорове розмежування: фіолетовий для присутніх та жовтий для відсутніх.

Праворуч реалізовано інтерактивний календар на поточний місяць, який дає змогу швидко орієнтуватися у датах. Під календарем розміщено

список найближчих подій: «Book Fair», «Sports Day», «Art Exhibition», кожна з яких має опис і час проведення.

У нижній частині сторінки розміщені два додаткові блоки:

– Finance – лінійна діаграма, яка демонструє зміну доходів та витрат закладу в динаміці;

– Announcements – список важливих оголошень для користувачів, наприклад, нагадування про «Picture Day».

Головна сторінка забезпечує зручний доступ до ключових даних у реальному часі, сприяє прийняттю рішень адміністрацією закладу та інформуванню користувачів системи.

Для забезпечення безпеки та контролю доступу до функціоналу системи управління шкільним закладом реалізовано механізм авторизації користувачів. Вхід у систему здійснюється через спеціальну форму, яка передбачає введення імені користувача та пароля (рис. 3.4). Це дозволяє ідентифікувати користувача, обмежити доступ до конфіденційної інформації та розмежувати права доступу в залежності від ролі (адміністратор, учитель, учень). Проста та інтуїтивно зрозуміла структура інтерфейсу форми входу сприяє зручності користування та підвищує загальну ефективність системи.

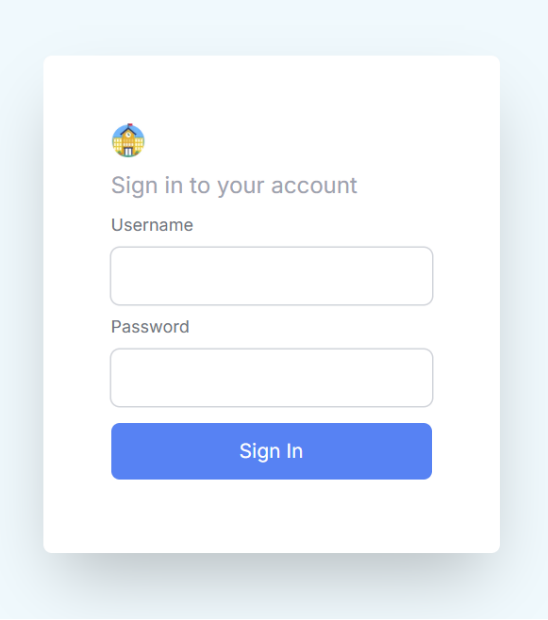


Рисунок 3.4 – Форма авторизації

Центральним елементом є компактна форма входу, розміщена на світлому фоні з м’якою тінню, що візуально виділяє її на сторінці.

У верхній частині форми розташовано іконку школи, що виконує роль впізнаваного візуального елементу. Під нею знаходиться заголовок «Sign in to your account», що інформує користувача про призначення даного інтерфейсу.

Форма складається з двох полів для введення даних:

– Username – поле для введення імені користувача;

– Password – поле для введення пароля.

На рисунку 3.5 продемонстровано інтерфейс сторінки «Усі вчителі» у вебзастосунку.

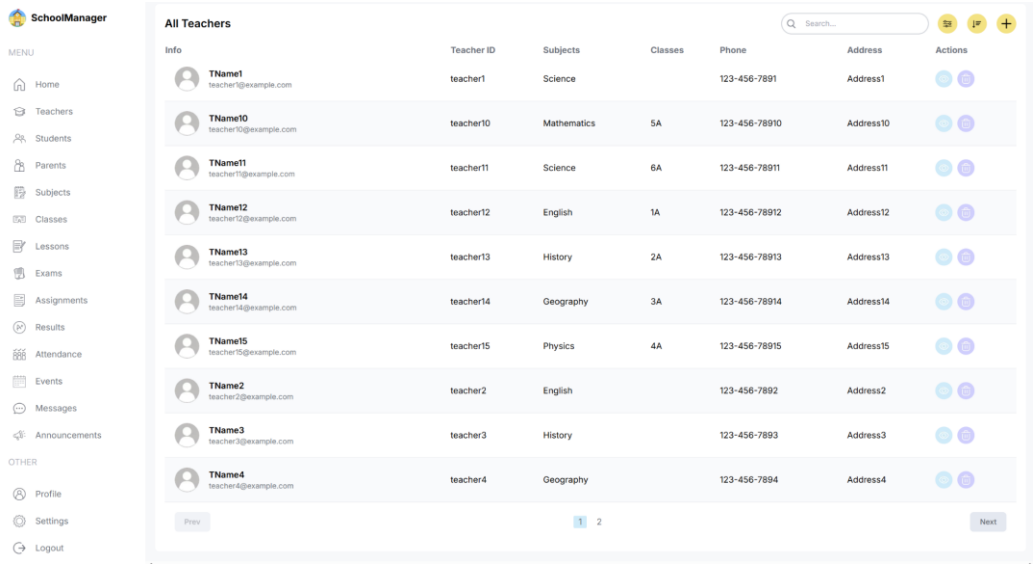


Рисунок 3.5 – Сторінка «Усі вчителі»

Основна частина інтерфейсу присвячена таблиці з інформацією про вчителів. Вона складається з кількох стовпців:

– Інфо – містить аватар, ім’я та email вчителя;

– Teacher ID – унікальний ідентифікатор;

– Subjects – предмет викладання;

– Classes – клас, закріплений за вчителем;

– Phone – контактний номер телефону;

– Address – адреса проживання;

– Actions – кнопки для перегляду, редагування або видалення запису (позначені піктограмами ока, олівця та смітника відповідно).

У верхній частині таблиці розміщене поле пошуку та кнопки для фільтрації і додавання нового викладача. У нижній частині таблиці реалізована пагінація, яка дозволяє переходити між сторінками списку викладачів. Дизайн інтерфейсу витриманий у світлих тонах, із чіткою візуальною ієрархією, що сприяє зручності навігації та сприйняттю інформації.

Сторінки, присвячені учням, батькам, предметам, класам, урокам, іспитам, домашнім завданням, результатам та відвідуваності, мають єдиний уніфікований дизайн, аналогічний до сторінки вчителів. Кожна з них представлена у вигляді зручного списку, що дозволяє переглядати основну інформацію в табличному форматі. Для кожного запису передбачені основні дії: перегляд деталей, редагування та видалення, які реалізовані за допомогою відповідних піктограм.

Далі розглянемо детальну сторінку профілю викладача в системі управління шкільним закладом(рис 3.6). У верхній частині інтерфейсу розміщено особисту інформацію про викладача, включаючи фотографію, ім’я, контактні дані, електронну пошту, дату приєднання, а також оцінку викладача у вигляді позначки.

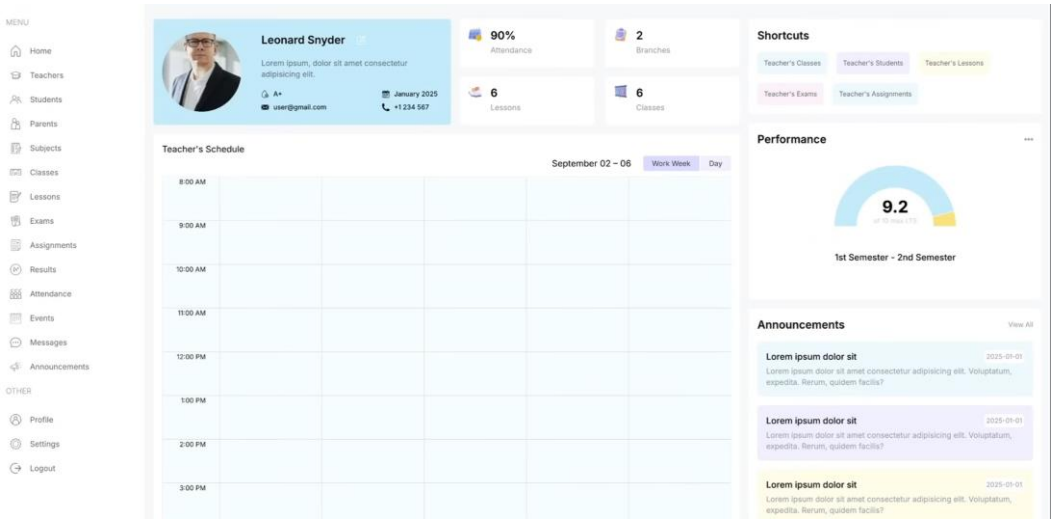


Рисунок 3.6 – Сторінка профілю вчителя

Праворуч від профілю відображаються основні показники діяльності: відвідуваність, кількість філій, уроків і класів. Нижче розташовано розклад занять викладача, представлений у вигляді таблиці з поділом за годинами та днями тижня, що дозволяє легко планувати навчальну діяльність. Також тут є індикатор успішності викладача (Performance), що відображає середню оцінку за семестри, і блок оголошень, який містить важливу інформацію або оновлення, що стосуються освітнього процесу.

У системі передбачена форма для створення профілю вчителя(рис. 3.7). Дизайн вікна виконано у світлих тонах з чіткою структурою, що поділяє поля введення даних на дві основні групи: Authentication Information (Інформація для автентифікації) та Personal Information (Особиста інформація), що сприяє логічному та зручному заповненню форми.

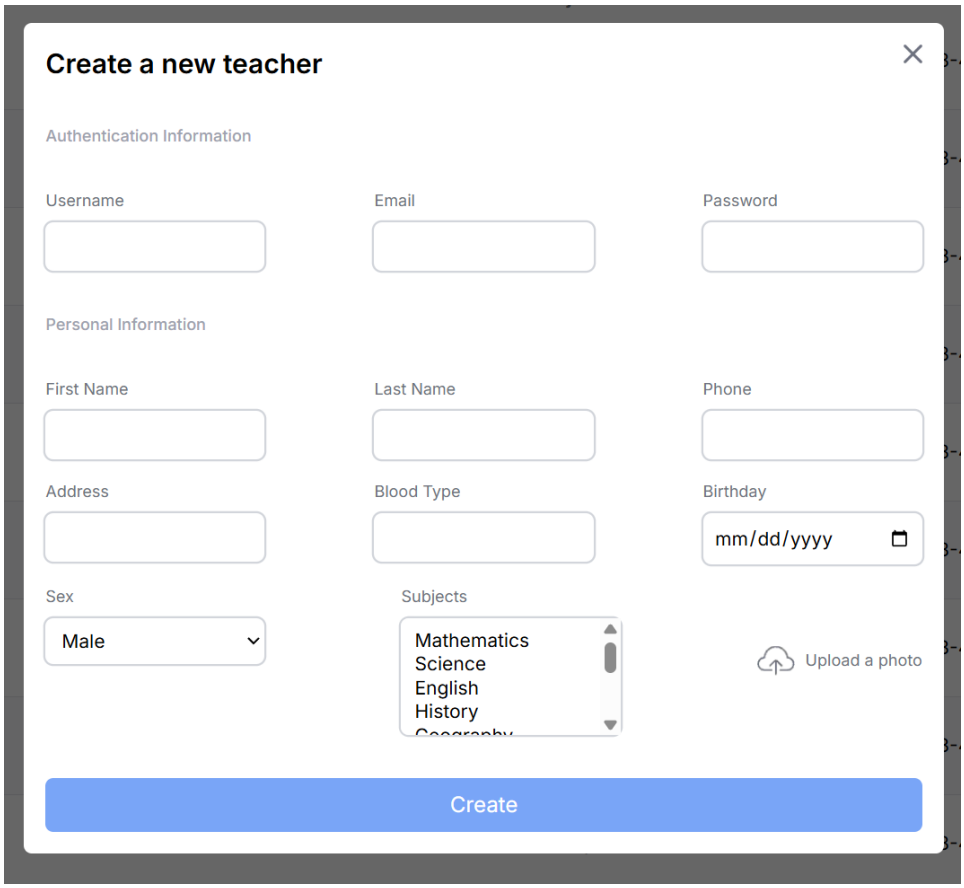


Рисунок 3.7 – Форма для створення профілю вчителя

У першому блоці, що відповідає за автентифікаційні дані, знаходяться три текстові поля:

– Username – для введення унікального імені користувача;

– Email – для електронної пошти;

– Password – для задання пароля доступу до системи.

Ці поля є обов’язковими для створення облікового запису викладача, оскільки вони забезпечують ідентифікацію та доступ до системи.

У другій секції – Personal Information – розміщено розширений набір полів, необхідних для збору персональних даних викладача. Вона містить:

– First Name і Last Name – для імені та прізвища;

– Phone – для номера телефону;

– Address – для зазначення домашньої адреси;

– Blood Type – поле для введення групи крові (може бути корисним у разі надзвичайних ситуацій);

– Birthday – поле з підтримкою вибору дати, що дозволяє вказати дату народження через зручний календар.

Далі користувач може обрати стать викладача за допомогою випадаючого списку Sex, а також зазначити один або декілька навчальних предметів у полі Subjects, яке реалізовано у вигляді мультиселекту (список з можливістю вибору кількох пунктів). Це дозволяє системі коректно відобразити належність викладача до відповідних дисциплін.

Також у правій частині форми розташована іконка у вигляді хмари зі стрілкою, яка дозволяє завантажити фотографію викладача – важлива функція для подальшої візуальної ідентифікації у списках або на персональній сторінці.

В нижній частині модального вікна знаходиться велика кнопка «Create», яка підтверджує введення всіх даних та ініціює процес створення нового запису в базі даних. Весь інтерфейс побудований таким чином, щоб полегшити введення інформації для адміністратора та зменшити ймовірність

помилок, забезпечуючи високу зручність використання та ефективність роботи з системою.

При вході в систему під обліковим записом учня, інтерфейс автоматично адаптується під роль користувача, відображаючи лише ті функціональні модулі, які безпосередньо стосуються освітнього процесу (рис. 3.8). В результаті навігаційне меню зліва стає більш компактним і містить лише необхідні пункти: Exams, Assignments, Results, Attendance, Events, Messages та Announcements.

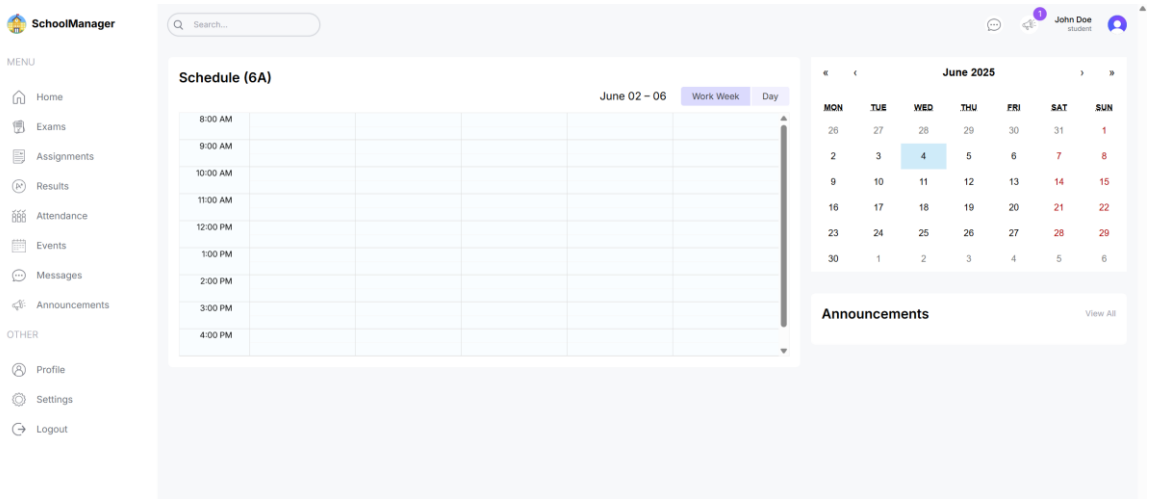


Рисунок 3.8 – Головна сторінка учня

Центральну частину екрана займає розклад занять, відображений у форматі табличної сітки, яка чітко структурує інформацію за днями тижня та годинами. Кожен блок у розкладі відповідає окремому уроку та містить назву предмета, а також, у деяких випадках, ім’я викладача або номер кабінету. Такий формат дозволяє учневі швидко зорієнтуватися у своєму навчальному дні та планувати час.

Праворуч від розкладу розміщено календар, який показує поточний місяць із можливістю перемикання між місяцями. У цьому блоці виділено дні, коли заплановано події, уроки або важливі дати.

Загальний вигляд сторінки побудований відповідно до принципів мінімалізму та функціональної доступності. Усі елементи логічно згруповані,

інтерфейс лишається зрозумілим навіть для молодших користувачів, а візуальні акценти допомагають швидко зорієнтуватися у поточних завданнях та подіях навчального процесу.

3.4 Перспективи подальшого розвитку та розширення системи управління шкільним закладом

Під час створення платформи для управління шкільним середовищем було реалізовано ключові модулі, необхідні для ефективного функціонування навчального процесу. Проте розроблена система має значний потенціал для подальшого розширення та вдосконалення. Насамперед, одним із пріоритетних напрямків розвитку є впровадження аналітичного модуля, який дозволятиме адміністрації та вчителям отримувати детальну статистику щодо успішності учнів, відвідуваності, навантаження вчителів і ефективності навчальних програм. Це сприятиме прийняттю обґрунтованих управлінських рішень і підвищенню якості освіти.

Доцільним також є розширення функціоналу особистого кабінету учня та батьків. Наприклад, можна додати систему нагадувань про майбутні події, домашні завдання та контрольні роботи, інтегрувати щоденник з оцінками, забезпечити доступ до навчальних матеріалів і методичних рекомендацій. Також корисним стане відображення динаміки навчання у вигляді графіків чи індикаторів прогресу, що надасть учням і батькам зворотний зв’язок про результати навчального процесу.

Окрему увагу варто приділити оптимізації системи обліку відвідуваності. Перспективним рішенням буде інтеграція з технологіями QR-кодів, що дозволить автоматизувати фіксацію присутності на уроках, зменшити кількість ручного введення та забезпечити точнішу статистику.

З метою розширення можливостей платформи варто реалізувати механізми сповіщень, зокрема електронну розсилку новин, сповіщень про зміни у розкладі, нагадування про події, повідомлення про отримані оцінки.

Ці сповіщення можуть бути інтегровані з email-сервісами або месенджерами, що зробить систему більш зручною та сучасною.

У контексті підвищення безпеки та захисту персональних даних, важливим кроком стане впровадження двохфакторної автентифікації, контроль сесій, обмеження доступу за ролями, шифрування критично важливих даних. Також рекомендується розробити механізм журналювання дій користувачів, що дозволить відстежувати активність у системі для запобігання зловживанням.

Ще одним перспективним етапом є мобільна адаптація платформи або створення окремого мобільного застосунку для Android та iOS. Це надасть можливість користувачам взаємодіяти з системою у зручному форматі незалежно від пристрою, переглядати розклад, отримувати сповіщення, надсилати повідомлення чи переглядати результати у будь-який час.

Для розширення функціоналу адміністративної частини варто реалізувати повноцінну систему управління користувачами, яка включатиме можливості створення нових акаунтів, редагування ролей, призначення прав доступу, перегляду активності. Важливим буде також впровадження механізмів резервного копіювання та відновлення даних, що підвищить надійність системи в умовах збоїв або втрати інформації.

З точки зору інтерфейсу, перспективним є впровадження адаптивного дизайну та нічної теми, що дозволить зробити використання системи більш комфортним для користувачів з різними потребами. Крім того, варто розглянути проведення UX-досліджень та тестування для виявлення слабких місць у взаємодії користувачів із платформою та подальшої її оптимізації.

Платформа має всі технічні передумови для масштабування, у тому числі можливість поступового переходу до мікросервісної архітектури, що дозволить краще ізолювати функціональні блоки, забезпечити стійкість до навантажень і спростити підтримку системи. Таким чином, подальший розвиток проєкту орієнтований на розширення функціоналу, підвищення стабільності, доступності та користувацької цінності освітньої платформи.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований повнофункціональний вебзастосунок для управління шкільним закладом. Реалізована платформа охоплює управління користувачами (учні, вчителі, батьки, адміністрація), ведення розкладу занять, реєстрацію відвідуваності, організацію навчальних подій (уроки, іспити, домашні завдання), облік результатів і формування аналітики.

Під час реалізації проєкту було проведено аналіз актуальних вимог до сучасних інформаційних систем в освіті, обрано оптимальні технології – серед них фреймворки Next.js для клієнтської частини та Prisma з базою даних PostgreSQL у зв’язці з серверною логікою на Node.js. Застосунок має адаптивний дизайн, зручну навігацію та підтримує взаємодію з системою через REST API.

Особливу увагу приділено забезпеченню зручності взаємодії користувачів із платформою: вчителі мають доступ до свого розкладу, списків учнів та завдань; учні – до результатів, розкладу, повідомлень; батьки – до даних дітей і навчального прогресу. Вбудована система аутентифікації та контролю доступу гарантує безпеку даних. Для тестування були перевірені ключові сценарії роботи користувачів, забезпечено правильну роботу в різних браузерах та на різних розширеннях екрана.

У перспективі система може бути доповнена мобільним застосунком, глибшою аналітикою, двофакторною автентифікацією, мультимовністю та інтеграцією з державними освітніми платформами. Розроблений застосунок має реальну практичну цінність і може стати основою для цифрової трансформації навчального процесу в українських школах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Литвинова С. Г.. Цифрова трансформація науково-освітніх середовищ в умовах воєнного стану: збірник матеріалів звітної науково-практичної конференції. Київ: ІЦО НАПН України, 2025.

2. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/ (дата звернення 15.05.2025).

3. Міністерство освіти і науки України. Стан і розвиток цифровізації освіти в Україні. URL: https://info.osvita.te.ua/stan-i-rozvytok-dydzhytalizatsii-osvity-v-ukraini/ (дата звернення 15.05.2025).

4. AIN.ua. Школа у смартфоні. Як працює платформа «Єдина школа». URL: https://ain.ua/special/meet-eschool (дата звернення 15.05.2025).

5. Моя Освіта. Офіційний сайт освітньої платформи «Моя Освіта». URL: https://my.osvita.net/ (дата звернення 15.06.2025).

6. Next.js. App Router – інтерактивний курс. URL: https://nextjs.org/learn/dashboard-app (дата звернення 19.05.2025).

7. JSX — TypeScript Handbook. URL: https://www.typescriptlang.org/docs/handbook/jsx.html (дата звернення 19.05.2025).

8. Prisma Labs. Prisma ORM Documentation. URL: https://www.prisma.io/docs (дата звернення 19.05.2025).

9. Clerk. Next.js Quickstart. URL: https://clerk.com/docs/quickstarts/nextjs (дата звернення 19.05.2025).

10. Cloudinary. Programmatically Uploading Images, Videos, and Other Files. URL: https://cloudinary.com/documentation/upload_images (дата звернення 19.05.2025).

11. React Hook Form Team. API Documentation. URL: https://react-hook-form.com/api (дата звернення 20.05.2025).

12. Zod Authors. Zod: TypeScript-first schema validation. URL: https://zod.dev/ (дата звернення 21.05.2025).

13. UploadThing Team. UploadThing React API Reference. URL: https://docs.uploadthing.com/api-reference/react (дата звернення 21.05.2025).

14. react-hot-toast Documentation. URL: https://react-hot-toast.com/docs (дата звернення 22.05.2025).

15. DB-Engines Ranking: popularity of database management systems, June 2025. URL: https://db-engines.com/en/ranking (дата звернення 22.05.2025).

16. Visual Studio Magazine. (2024). Stack Overflow Dev Survey: VS Code, Visual Studio and .NET Shine. Visual Studio Magazine. URL: https://visualstudiomagazine.com/articles/2024/07/26/so-dev-survey.aspx (дата звернення 22.05.2025).

17. DevTools Academy. (2024). Supabase vs Clerk: детальний огляд систем автентифікації. URL: https://www.devtoolsacademy.com/blog/supabase-vs-clerk/ (дата звернення 22.05.2025).

18. Erikson M. (2024). Why You Should Use Redux in 2024. GitNation Talk. URL: https://gitnation.com/contents/why-you-should-use-redux-in-2024 (дата звернення 22.05.2025).