НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ)

Інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
Інформаційних технологій	Комп'ютерних наук
(назва факультету (ННІ))	(назва кафедри)
_____ Ігор БОЛБОТ	_____ Белла ЛЬВІВНА
(підпис) (ім'я ПРІЗВИЩЕ)	(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 20__ р.	"___" _____ 20__ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему «Аналітична платформа визначення досягнень здобувачів на базі WazerCode»

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Інформаційні управляючі системи та технології»

Орієнтація освітньої програми Освітньо професійна

Гарант освітньої програми

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент _____ Віталій СВАТКО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ Богдан КІЧАК
(підпис) (ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
к.т.н, доцент _____ Белла ГОЛУБ
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"____" _____20____року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Кічак Богдан Володимирович
(прізвище, ім'я, по батькові)

Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: Аналітична платформа визначення досягнень здобувачів на базі WazerCode

затверджена наказом від "1" листопада 2024р. №1964 С

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи

1. *Платформа WazerCode з використанням засобів аналізу освітніх досягнень;*
2. *Оптимізація алгоритмів збору та опрацювання даних на основі аналізу освітньої діяльності здобувачів;*
3. *Розробка гнучкого модуля налаштувань інтерфейсу та покращення адаптивної аналітичної системи для різних рівнів користувачів;*
4. *Тестування платформи на різних освітніх сценаріях та Web-інтерфейсу.*

Перелік питань, що підлягають дослідженню:

1. Аналіз існуючих методів створення аналітичних платформ для навчання;
2. Огляд існуючих рішень та потенційних конкурентів;
3. Проведення досліджень на створеній аналітичній платформі;
4. Висновки.

Перелік графічного матеріалу (за потреби) постер, презентація

Дата видачі завдання "1" Листопад 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Віталій СВАТКО
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Богдан КІЧАК
підпис (ім'я ПРІЗВИЩЕ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API — Application Programming Interface (програмний інтерфейс додатку)

CRUD — Create, Read, Update, Delete (базові операції для роботи з даними)

HTTP — HyperText Transfer Protocol (протокол передачі гіпертексту)

JSON — JavaScript Object Notation (формат обміну даними)

KPI — Key Performance Indicators (ключові показники ефективності)

LMS — Learning Management System (система управління навчанням)

Next.js — фреймворк для React з підтримкою серверного рендерингу

ORM — Object-Relational Mapping (об'єктно-реляційне відображення)

React — бібліотека JavaScript для створення користувацьких інтерфейсів

REST — Representational State Transfer (архітектурний стиль для веб-сервісів)

UI — User Interface (користувацький інтерфейс)

Аналітична платформа — комплекс інструментів для збору, обробки та візуалізації даних про навчальну діяльність

Метод Наївного Байєса — ймовірнісний алгоритм класифікації даних

SDK — Software Development Kit (набір інструментів для розробки програмного забезпечення).

SMTP — Simple Mail Transfer Protocol (протокол передачі електронної пошти).

SQL — Structured Query Language (мова структурованих запитів до баз даних).

SSL/TLS — Secure Sockets Layer / Transport Layer Security (протоколи шифрування для забезпечення безпечного з'єднання).

Tailwind CSS — utility-first CSS фреймворк для швидкої розробки інтерфейсів.

Turbopack — інструмент для збірки веб-додатків, наступник Webpack.

TypeScript — типізована надбудова над JavaScript.

ВСТУП

Актуальність теми. Онлайн-платформи для навчання дозволяють оптимізувати основні процеси, дізнаватись про нові інноваційні рішення, швидко навчати персонал, без потреби фізичного відвідування курсів.

В сучасному світі тенденції розвитку потребують використання цифрових ресурсів, які допомагають значно підвищити ефективність навчання, його темп, та звичайно – прозорість навчання. Під час роботи з аналітичними даними здобувач освіти може переглянути свої досягнення, отримувати персоналізовані рекомендації та мати доступ до адаптивного навчання.

Розповсюдження дистанційної та гібридної форми навчання потребує зберігання, аналіз, передачу навчальних даних для подальшої оптимізації навчальних процесів, які позитивно впливають на темп, метод навчання.

В своїй роботі інформаційно-аналітичне забезпечення навчального процесу є основним фактором для підвищення якості освіти, адже аналітика інформації, яка отримується під час опрацювання курсу дає можливість в швидкому режимі реагувати на різні освітні виклики, дозволяє проводити різні моніторинги активності студентів, впроваджувати та контролювати за результатом дій нових начальних методик, програм розвитку, тощо.

Мета і завдання дослідження. Метою цієї роботи є розробка та покращення системи аналітики на навчальній платформі WazerCode, що забезпечить можливість отримати автоматизований облік, проводити аналіз курсів, отримувати візуалізацію освітніх результатів, що вимагається в навчальному процесі.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Проаналізувати сучасні теоретичні й практичні підходи до моніторингу освітніх досягнень із використанням цифрових платформ.
- Вивчити ключові функціональні та технічні можливості WazerCode для побудови аналітичної системи.

- Розробити структуру і алгоритми збору, обробки й візуалізації даних щодо досягнень здобувачів на платформі WazerCode.
- Реалізувати гнучкий інтерфейс користувача із можливістю налаштування аналітичних модулів.
- Провести тестування платформи у освітньому середовищі, оцінити її функціональність, надійність та перспективи впровадження.

Об’єкт дослідження — процес розробки та впровадження аналітичної платформи визначення досягнень здобувачів на базі WazerCode.

Предмет дослідження — методи, моделі та засоби для побудови аналітичної платформи визначення досягнень здобувачів на базі WazerCode.

Методи дослідження. Для вирішення поставлених завдань використовувалися методи системного аналізу, об'єктно-орієнтованого проєктування. Практична частина реалізована з використанням мови програмування TypeScript.

Наукова новизна одержаних результатів полягає у дослідженні, використанні та вдосконаленні інформаційної системи з онлайн курсами. Система інтегрує технології опрацювання інформації, це надає можливість персоналізувати аналіз навчальної діяльності, підвищити точність оцінювання та забезпечити візуалізацію освітніх результатів у реальному часі.

Практичне значення роботи полягає у розробці та застосуванні аналітичної платформи на базі WazerCode для автоматизованого моніторингу, обліку та аналізу освітніх досягнень здобувачів у реальному освітньому процесі.

Структура та обсяг роботи. Пояснювальна записка складається з чотирьох основних розділів.

У першому розділі проведено аналіз предметної області, визначено проблеми моніторингу освітніх досягнень у системах управління навчанням, охарактеризовано поняття аналітичної платформи та аспекти розробки аналітичної системи для WazerCode LMS, а також сформульовано завдання дослідження.

У другому розділі розглядаються головні методи створення аналітичних платформ, включаючи архітектурну модульну інтеграцію, оптимізацію обчислювальних ресурсів, порівняльний аналіз існуючих рішень у React (React Admin, Tremor, Ant Design Pro), а також визначено функціональні та нефункціональні вимоги до системи.

Третій розділ присвячений методам та технологіям аналізу освітніх даних, зокрема моделюванню джерел даних, застосуванню методів 1-Rule та Наївного Байеса, виявленню асоціативних правил, кластеризації користувачів, а також розрахунку ключових показників ефективності (KPI) навчального процесу.

У четвертому розділі описано інтеграцію та вдосконалення платформи WazerCode, включаючи вибір та впровадження сучасних інструментів веб-розробки (Next.js 15, React 19, Tailwind CSS, Better-Auth, Neon Postgres, Prisma ORM), архітектуру DevOps та інфраструктури на базі AWS і Vercel.

Основна частина пояснювальної записки займає 59 сторінок, із них 55 сторінок основного тексту. Додатки мають обсяг в 18 сторінок. Робота містить 21 рисунок. Для виконання дослідження та підготовки матеріалів використано 29 джерел інформації.

Апробація. За результатами роботи були опубліковані тези під назвою «Аналітична платформа визначення досягнень здобувачів на базі WazerCode» в збірнику наукових праць за матеріалами XV Міжнародної науково-практичної конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта» за 28-29 жовтня 2025 року та тези під назвою «Інтеграція аналітичних модулів у платформу WazerCode з метою визначення освітніх досягнень» в збірнику наукових праць за матеріалами XVIII міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2025»

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальний опис платформ для навчання

Навчальна платформа - це онлайн сервіс, який пов’язує в собі декілька функцій, а саме: відео-хостинг з навчальними матеріалами, та інформаційну систему, в якій зберігаються дані про успішність, проходження матеріалу, та інші статистичні дані для роботи з платформою. Навчальні платформи використовуються по всьому світу, та об’єднують студентів з різних країн.[1].

Історичний початок LMS платформ можна датувати з 1990-х років, коли на ринку з’явились такі платформи як WebCT, Blackboard, Moodle, які використовуються і сьогодні. Перші версії мали лімітований функціонал, але з часом розвиток цих платформ сформував поточні правила в цій галузі.

На початку 2000-х з’явилися платформи масових відкритих онлайн-курсів (MOOC) — Coursera, edX, Udacity, Khan Academy. Вони зробили високоякісну освіту доступною для мільйонів користувачів у світі.

У 2010-х і 2020-х освіта перейшла на хмарні і мобільні технології, інтегрувала аналітику, штучний інтелект, персоналізацію навчання та елементи гейміфікації.

В наш час без онлайн платформ для навчання ми не можемо уявити процес отримання знань, адже багатьом набагато зручніше пройти матеріал у вільний час, ніж турбуватись за багато обставин, які в певній мірі не залежать від користувача.

За рахунок своєї доступності, зручності та відкритості для нових користувачів – онлайн LMS платформи отримали великий попит у більшості країнах світу.

Завдяки певним представникам ми можемо отримати сертифікації, які впливають на працевлаштування, або пройти курс підвищення кваліфікації, який збільшить заробітню плату фахівця, та підтвердить його знання. І все це – не виходячи з дому, лише за допомогою комп’ютера та інтернет-з’єднання.

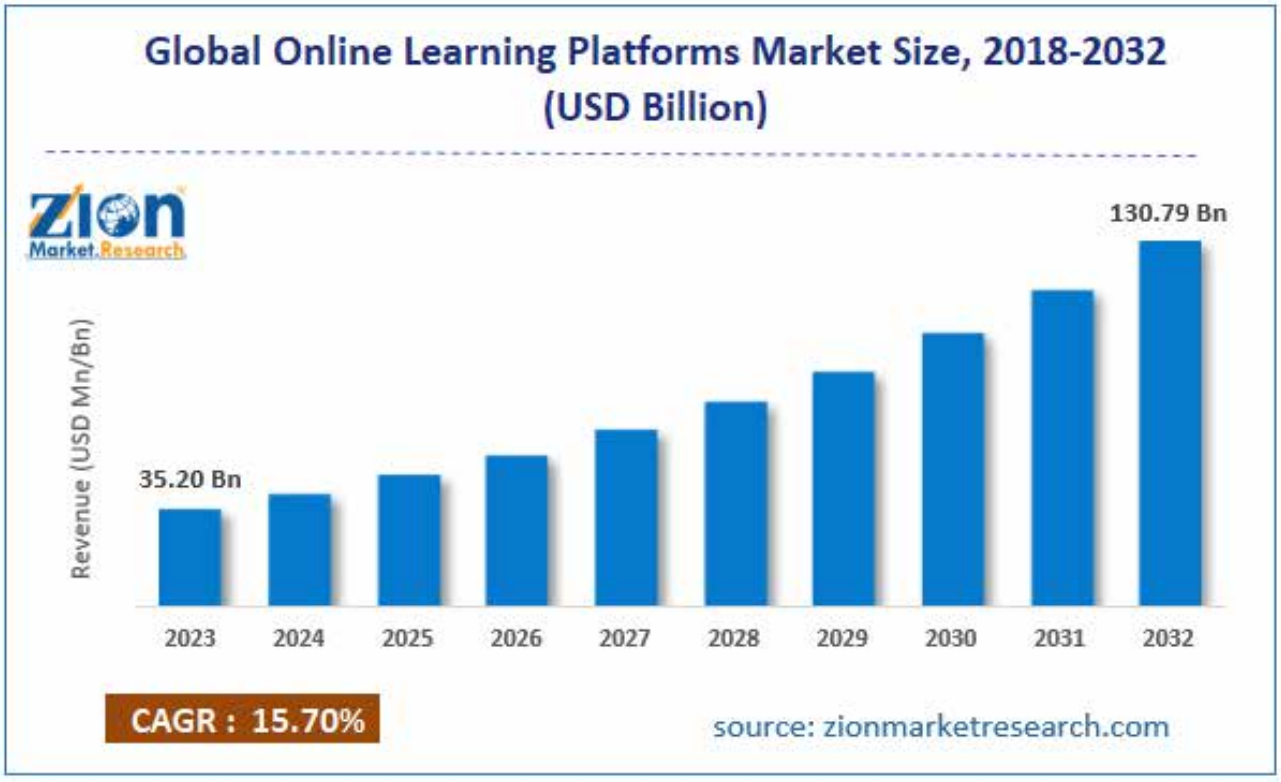


Рис. 1 Прогнозована глобальна ринкова капіталізація LMS платформ

На рис.1 зображений графік глобального ринку навчальних онлайн-платформ демонструє стабільно високі темпи приросту: лише за 2023 рік його обсяг досягнув 35,2 млрд доларів США, а до 2032 року прогнозується збільшення до 130,79 млрд доларів із середньорічним темпом зростання (CAGR) 15,7%. Ці дані свідчать про фундаментальні зміни у структурі освітніх послуг, які відбуваються під впливом новітніх технологічних та соціальних чинників.

Онлайн-освіта перетворюється зі спеціалізованого інструменту на інтегральний елемент освітньої системи, що забезпечує доступ до знань без географічних та часових обмежень. Основними драйверами такого розвитку стали розширення доступу до інтернету, масове використання мобільних і комп’ютерних пристроїв, а також значний попит на гнучкі формати навчання як серед студентів, так і серед працюючих професіоналів. Так, дистанційне навчання у період пандемії COVID-19 стало необхідністю, а університети, корпорації та незалежні експерти почали активно інвестувати у цифрові платформи для створення та поширення якісного навчального контенту.

1.2 Поняття «Аналітична платформа»

Аналітична платформа — це комплексний набір інструментів, що дозволяють працювати з інформацією, яка отримується в процесі роботи програми. Аналітичні платформи найбільш потрібні для адміністрації та менеджменту освітньої платформи для навчання.

У сучасних навчальних платформах користувачі мають високий рівень долученості до процесу навчання, що може бути в нагоді під час формування аналітичних звітів. В залежності від побудови курсу ми, як викладачі, або менеджери маємо розуміти – які маємо метрики стосовно долученості до цього курсу, його процесу проходження нашими студентами.

Наявність аналітичної платформи є невід'ємною частиною будь якої production-ready LMS платформи. Не зважаючи на те, що платформа може мати сучасний дизайн, яскравий маркетинг та високу якість виконання та інноваційним підходом до навчання, її успішність на відкритому для різних розробників ринку може бути не завжди виграшною, тим паче якщо ми будемо бачити різний відтік користувачів, який не має логічного пояснення (як то час доби, день тижня і т.п).

Обслуговування аналітичної платформи для визначення досягнень здобувачів на базі WazerCode включає кілька ключових аспектів, кожен із яких потребує спеціалізованого підходу для забезпечення максимальної продуктивності та надійності системи:

- **Обробка даних**. Оптимізація алгоритмів збору, валідації та обробки освітніх показників з метою мінімізації часу відповіді та збільшення пропускної здатності платформи без втрати точності результатів.
- **Обчислювальні ресурси**. Раціональне використання серверних і клієнтських обчислювальних потужностей для виконання аналітичних завдань, таких як побудова динамічних звітів, обчислення показників успішності, прогнозування результатів та адаптивна оцінка.

- **Управління пам’яттю**. Ефективне керування оперативною пам’яттю і кешуванням даних для запобігання перевантаження системи, що здатне викликати затримки у відображенні інформації або навіть збої у роботі платформи.

1.3 Аспекти розробки аналітичної системи для WazerCode LMS

Продуктивність є критично важливим аспектом функціонування аналітичної платформи у вигляді веб-додатку, розробленого на React, у системі управління навчанням (LMS) WazerCode. Від рівня продуктивності залежить швидкість відображення аналітичних даних, інтерактивність інтерфейсу та загальний користувацький досвід. Низька продуктивність може призводити до затримок у рендерингу компонентів, уповільнення часу завантаження сторінок, зависань при оновленні дашбордів та збоїв у роботі додатку.

Однією з основних проблем є надмірне навантаження на клієнтський процесор через складні обчислення та візуалізації, що виконуються безпосередньо у браузері. Це може спричинити "гальмування" інтерфейсу, особливо при обробці великих обсягів даних або у випадку відсутності оптимізації React-компонентів. Неправильне управління станом додатку або неефективне повторне рендерення компонентів також суттєво впливає на продуктивність.

Ще одним важливим аспектом є використання пам’яті браузера. Неоптимізоване кешування даних або надмірне зберігання в оперативній пам’яті може призводити до перевантаження системи, що ускладнює роботу з аналітичними звітами, викликаючи затримки або навіть аварійне завершення сесії користувача.

Довгі часи завантаження веб-додатку можуть бути пов’язані з масштабом і кількістю завантажуваних ресурсів, таких як JavaScript-бандли, стилі, графічні елементи та дані для аналітики. Якщо застосовані методи стиснення та кешування не налаштовані належним чином, це призводить до погіршення досвіду користування.

Для подолання цих проблем застосовуються наступні підходи. Оптимізація React-компонентів і використання memoization дозволяють зменшити кількість непотрібних повторних рендерів, покращуючи швидкодію інтерфейсу. Розподіл обчислень між сервером і клієнтом із застосуванням серверного рендерингу або легких клієнтських компонентів допомагає зменшити навантаження на браузер.

Управління станом додатку реалізується за допомогою ефективних бібліотек, що мінімізують зайві оновлення UI та оптимізують роботу з кешем. Також застосовується динамічне підвантаження (code splitting) JavaScript-бандлів і оптимізація завантаження даних для мінімізації часу старту додатку.

Комплексне застосування цих методів забезпечує стабільну, швидку та ефективну роботу аналітичної платформи у веб-додатку LMS WazerCode на базі React, що сприяє якісному аналізу освітніх даних та комфортній взаємодії користувачів з системою.

1.4 Постановка завдання

У сучасній системі управління навчанням (LMS) WazerCode на базі веб-додатку React, де користувачі мають доступ до великих обсягів освітніх даних на різних пристроях, питання продуктивності аналітичної платформи набуває особливої ваги. Платформа повинна стабільно працювати як на потужних робочих станціях, так і на пристроях із обмеженими ресурсами, при цьому забезпечуючи швидкий доступ до аналітичних звітів і зручність взаємодії з інтерфейсом. Тому завдання оптимізації функціональності є одним із найважливіших для розробників.

Основна мета цього дослідження — зробити аналіз існуючих підходів до створення веб-аналітики у LMS, впровадити у платформу WazerCode інструменти аналітичної платформи, а також удосконалити систему для забезпечення швидкої та ефективної роботи на різних типах пристроїв, зокрема мобільних.

Для досягнення цієї мети необхідно вирішити такі завдання:

- Провести аналіз предметної області, зокрема існуючих методів оптимізації продуктивності у веб-додатках для освітньої аналітики;
- Розглянути ключові аспекти продуктивності, такі як обробка даних, обчислювальна потужність клієнтських і серверних компонентів, а також ефективне управління пам’яттю;
- Дослідити можливості та інструменти React для оптимізації рендерингу, управління станом і завантаження ресурсів;
- Впровадити механізми адаптивного налаштування відображення аналітичних інтерфейсів залежно від можливостей пристрою користувача;
- Покращити систему кешування та розподілення навантаження для зменшення часу відповіді платформи;

Провести тестування оновленої платформи на різних типах пристроїв і порівняти результати з початковими показниками ефективності. Спираючись на сказане, завдання даного дослідження полягає у створенні та впровадженні системи збору та обробки інформації, яка забезпечить доступ до інформації на різних пристроях, мінімізуючи затримки та зберігаючи доступ до інформації.

2 ДОСЛІДЖЕННЯ МЕТОДІВ ВПРОВАДЖЕННЯ

2.1 Головні методи створення аналітичних платформ

При створенні аналітичної платформи, розробники стикаються з основними можливостями впровадження, завдяки BackEnd стороні додатку. Отримання інформації з BackEnd надасть доступ до актуальних даних, які потім побачить користувач

2.2 Архітектурна модульна інтеграція

Модуль - це основний блок, на якому будується система. Це незалежна одиниця, що містить певну інкапсульовану поведінку, але і може існувати без візуальної презентації. Це дозволяє нам розгорнути стандартні залежності застосунку та зробити так, щоби **View** не керував усім застосунком, а переорієнтовувався в залежності від необхідної поведінки. Склад модуля для системи аналітики можна побачити на рис.2.

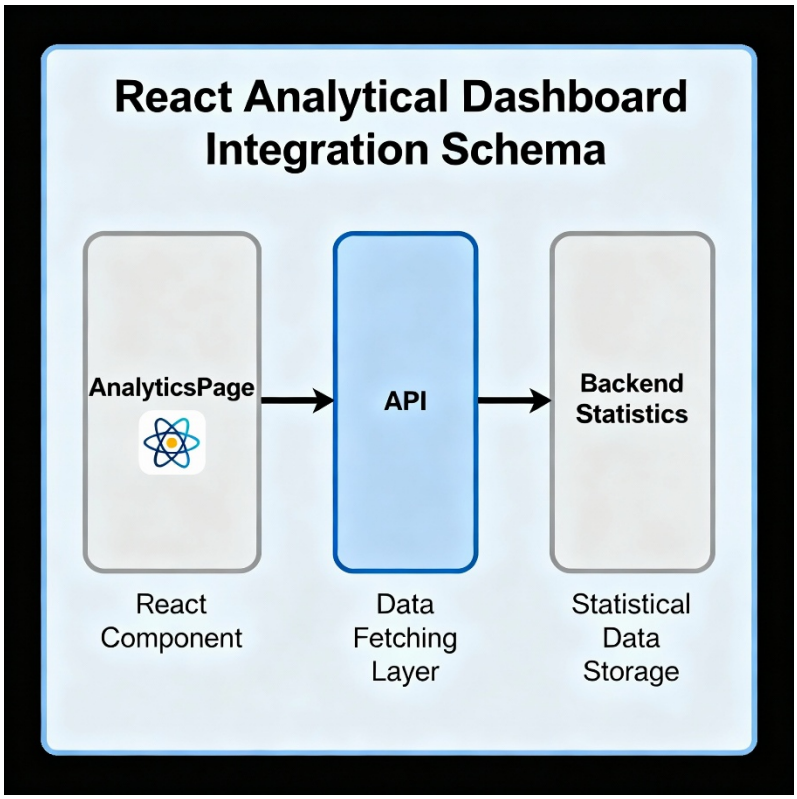


Рис. 2 Склад модулю аналітичної платформи

Модульна архітектура є дуже популярною для використання в проектах різного масштабу, адже ці модулі менш пов’язані з роботою іншої логіки застосунку, що доволі просто відстежити та контролювати у разі потреби.

Також модулі активно використовуються в Angular додатках, які і стали основним рушієм їх переносу в React, який до цього взагалі не міг дозволити побудувати розробнику додаток в стилі модульної архітектури, що було значним недоліком для виготовлення масштабних та не тільки розробок

Завдяки використанню модулів ми можемо робити його ізольованим, що дозволить нам створювати окремі сторінки, використовувати власні (модульні) компоненти, як то: UI графіки, таблиці, картки, та інші елементи.

На платформі WazerCode, використовуючи модульну архітектуру можна відмітити наступні складові модулю:

1. AnalyticsPage – окремий компонент, який містить усі аналітичні блоки.
2. Блоки метрик, графіків, топ-листів – кожен з них винесений у підкомпонент для легкості підтримки та модифікації.
3. API-шари – асинхронно отримують структуру даних з backend, агрерують і передають їх для рендерингу у відповідні UI-компоненти.

Зв'язок із основним застосунком – сторінка аналітики впроваджується в існуючий роутер або адміністративну частину застосунку без кардинальної зміни структури базового коду.

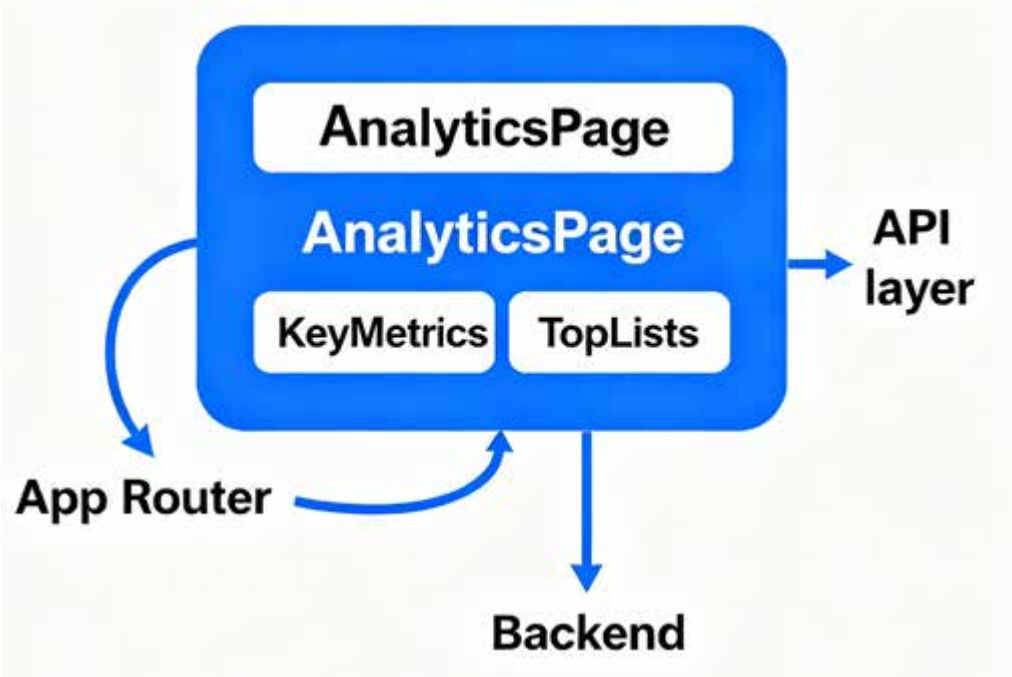


Рис. 3 Схематичний зв'язок модулю з іншими частинами проєкту

2.3 Оптимізація обчислювальних ресурсів

В процесі роботи з аналітичною сторінкою веб-застосунку, побудованої за компонентною архітектурою (наприклад, AnalyticsPage у середовищі React), одним з ключових процесів виступає асинхронне отримання даних з backend-сервісів та їх подальша обробка у клієнтському застосунку. Цей підхід є базовим для реалізації інтерактивної та масштабованої аналітики.

Для відображення нашого компонента потрібно використати його монтування. При цьому використовуються асинхронні функції, що звертаються саме до BackEnd API, вони повертають при цьому дані, які будуть пізніше знаходитись у статистиці. Тобто: курси, кількість користувачів, успішність, історія прогресу та інша інформація, яку вказує розробник.

Для повноцінної оптимізації роботи таких функцій потрібно використовувати принцип паралельного виконання процесів, а саме: Promises.

Проміс (англ. Promise) – це спеціальний об’єкт в JavaScript, котрий зв’язує “виробника” і “споживача” разом. В контексті нашої аналогії – це “список підписників”. Код-“виробник” виконується стільки часу, скільки потрібно щоб

отримати результат, а проміс робить результат доступним для коду-“споживача”, який на нього підсився, як тільки-но код-“виробник” поверне результат.

Після того, як додаток отримав інформацію – починається наступний процес її опрацювання – агрегація та трансформація даних.

Агрегува́ння — поєднання окремих одиниць або даних в одну одиницю або декілька одиниць. Наприклад, усі ціни окремих товарів і послуг об'єднують у загальний рівень цін або всі одиниці продукції поєднують у справжній валовий внутрішній продукт.

Трансформація даних (data transformation) означає перехід даних між носіями або зміна їхнього формату відповідно до заданих правил (наприклад, зміна типів, модифікація форми, форматування). У статистиці — це застосування математичних функцій до наборів даних для підготовки їх до аналізу або візуалізації.

Ці процеси напряму відносяться до правильної обробки інформації, яка вже буде демонструватись кінцевому користувачу, наприклад: розрахунок середнього балу (формулою середнього арифметичного), сортування, розрахунком агрегованого показнику (середній відсоток проходження курсу, тенденції за календарний місяць, тощо.) Ці розрахунки додають гнучкість програми, можуть легко модифікуватись, та динамічно сформуватись відповідно до потреб в розробці аналітичної платформи.

Останнім етапом опрацювання інформації можна назвати її візуалізацію, що полягає в рендерингу відповідних UI елементів, що прописані розробником, для цього можуть використовуватись сторонні бібліотеки React, що мають велику кількість налаштувань, наприклад: recharts, Nivo та інші. Такі бібліотеки мають широку підтримку компонентів дизайну, можуть надавати свої готові рішення та за рахунок гнучкості під час роботи – надають розробнику можливість зробити графік у відповідному до дизайну інтерфейсу стилі виконання.

2.4 Аналіз існуючих рішень у React

Завдяки широкому поширенню та популярності бібліотеки React, сторонні розробники розробили багато рішень, що дозволяють виконувати побудову аналітичних платформ легше, ніж здається на перший погляд. Кожен сайт середньої масштабованості та вище – повинен мати адміністративну панель для подальшого контролю зросту числа користувачів та їх поведінки.

React Admin

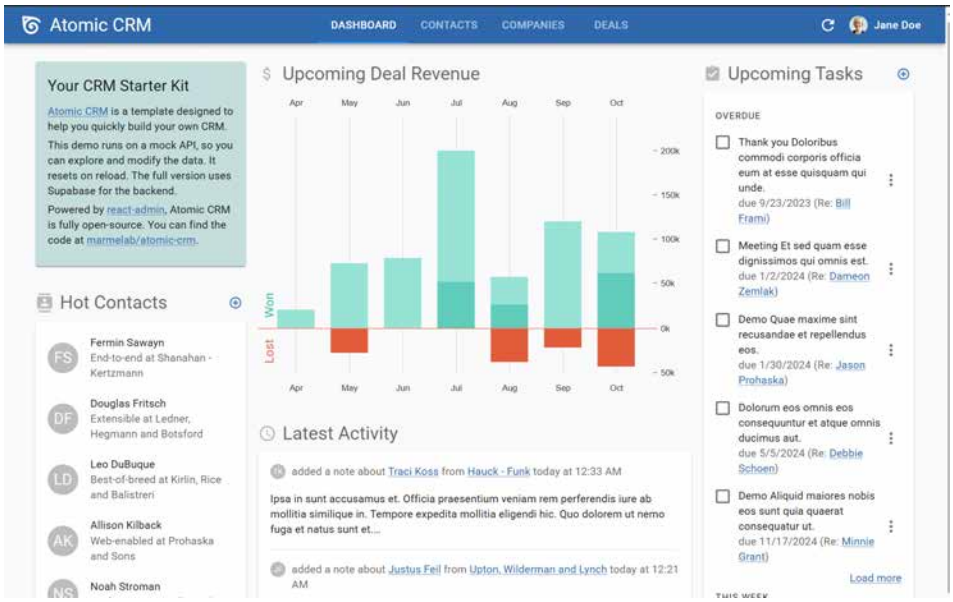


Рис. 4 Побудована CRM-система за допомогою бібліотеки React Admin

На рис. 4 зображений інтерфейс React Admin, що є відкритим програмним фреймворком для побудови адміністративних панелей і аналітичних платформ на основі бібліотеки React. Основна функціональна мета React Admin полягає у забезпеченні швидкої розробки веб-інтерфейсів для управління даними, що отримуються через REST або GraphQL API.

Бібліотека надає набір готових компонентів для реалізації основних операцій — перегляду списку сутностей (List), редагування (Edit), створення нових записів (Create), а також фільтрації, пошуку й навігації. Інтерфейс підтримує багаторівневу конфігурацію, кастомізацію логіки роботи з даними, гнучке стилізування та підключення додаткових віджетів. Фреймворк розроблений з урахуванням розширюваності, що дозволяє інтегрувати додаткові модулі та шаблони для складних бізнес задач або дашбордів.

В тому числі, React Admin є UI-агностичним, тобто може використовувати будь-яку компонентну бібліотеку для стилізації, але переймає Material UI як початковий стандарт. Зовнішня взаємодія із джерелами даних здійснюється через data providers, що забезпечують універсальність інтеграції з REST, GraphQL або іншими API. Це спрощує акумуляцію даних з різних бекендів для масштабних аналітичних платформ і корпоративних систем.

Tremor

Tremor — це сучасна open source React-бібліотека, орієнтована на швидку розробку аналітичних дашбордів та UI компонентів для візуалізації даних. Вона побудована на основі React і Tailwind CSS з використанням Radix UI, що забезпечує сучасний та адаптивний дизайн інтерфейсів з можливістю повного кастомізування стилів.

Tremor надає понад 35 готових компонентів для побудови графіків, таблиць, панелей та інших елементів аналітичної платформи. Вона підтримує створення інтерактивних і доступних дашбордів з використанням таких компонентів, як DonutChart, LineChart та інші типи візуалізацій. Усі компоненти бібліотеки мають налаштовувані пропси, які полегшують швидке налаштування та інтеграцію з бекенд даними без необхідності писати складний CSS-код.

Приклад розробленого Dashboard за допомогою Tremor можна побачити на рис. 5

Рис. 5 Демонстрація розробленого на Trello дашборду.

Tremor добре підходить для створення виробничих середовищ з великою кількістю даних — від простих звітів до складних інтерактивних дашбордів з можливістю навігації та фільтрації. Завдяки своїй гнучкості платформа дозволяє розробникам швидко розгорнути аналітичний інтерфейс, мінімізуючи витрати часу на дизайн та frontend-логіку. Бібліотека забезпечує підтримку всіх ключових типів графіків та інтерактивних віджетів для побудови комплексних бізнес-застосунків.

Ant Design Pro

Ant Design Pro — це потужний open source фреймворк для створення високопродуктивних адміністративних панелей, корпоративних дашбордів та аналітичних платформ на основі React. Рішення розроблене командою Ant Financial і широко використовується у великих підприємствах для швидкої побудови B2B-додатків

Ant Design Pro забезпечує готову структуру проекту, інтегрує сотні UI-компонентів, включно з таблицями, формами, графіками, списками й авторизацією. Фреймворк базується на принципах модульності та повторного використання компонентів — підтримуються блоки (Blocks), шаблони сторінок (Templates), які дозволяють збирати складну адміністративну платформу шляхом конструювання готових частин інтерфейсу.

Приклад розробленого Dashboard за допомогою Ant Design Pro можна побачити на рис. 6

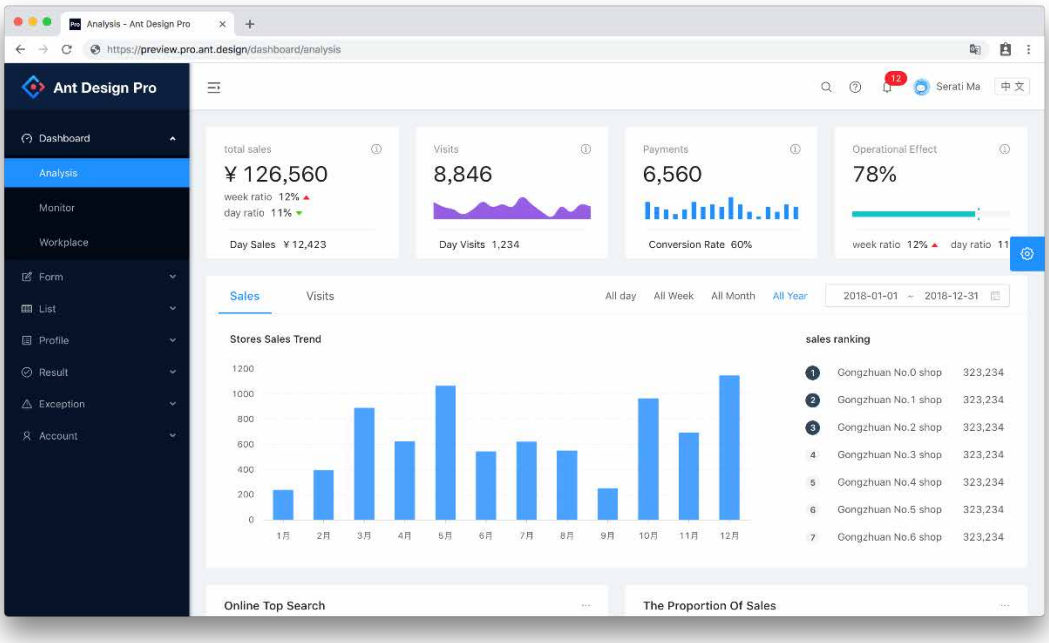


Рис. 6 Демонстрація розробленого за допомогою Ant Design Pro дашборду.

Ant Design Pro є комплексним рішенням для побудови бізнес-орієнтованих адміністративних і аналітичних платформ, що забезпечує високу продуктивність, масштабованість та інтеграцію з сучасним корпоративним тобто розробкою. Його модульна архітектура, численні шаблони та активна спільнота роблять фреймворк одним із найкращих виборів для створення enterprise-продуктів на базі React.

Порівняння

React Admin, Tremor та Ant Design Pro — це три сучасні фреймворки для побудови аналітичних платформ на основі React.

React Admin орієнтується на розробку адміністративних панелей та data-driven систем. Архітектура фреймворку наголошує на декларативному описі інтерфейсу для сутностей, потужній інтеграції з REST чи GraphQL API і гнучких CRUD-операціях. Сотні готових компонентів дозволяють швидко налаштувати управління даними одиниць, списків, фільтрів та складних форм. Висока розширюваність і підтримка кастомної бізнес-логіки роблять бібліотеку оптимальною для внутрішніх інструментів і корпоративних систем, де основний акцент — ефективна взаємодія користувача із структурованими наборами даних.

Tremor, навпаки, заточений на побудову сучасних аналітичних дашбордів з глибокою візуалізацією: графіки, діаграми, кільцеві чи лінійні звіти — усе це реалізується за лічені хвилини завдяки великому набору готових компонентів. Tremor використовує інтеграцію з Tailwind CSS та Radix UI, дозволяючи copy-paste підхід: компоненти можна швидко стилізувати або адаптувати просто через конфігураційні пропси, не вдаючись до тонких налаштувань CSS. Проста архітектура дає змогу максимально швидко зібрати прототип або виробничий дашборд з фокусом на візуальному представленні даних, але з меншою гнучкістю у побудові складної бізнес-логіки чи глибокому менеджменті сутностей.

Ant Design Pro — це комплексне рішення enterprise-класу для побудови адміністративних панелей із багатофункціональним UI, потенціалом масштабування, та гнучкою адаптацією під корпоративні стандарти. Фреймворк містить структуру проекту, блоки, шаблони сторінок, інтегрує сотні компонентів для таблиць, форм, графіків, моніторингу та контролю даних. Підтримка міжнародної локалізації, темізації та кастомних модулів дозволяє використовувати його для продуктів, де складність інтерфейсу, розподіл доступів та дизайн — ключовий фактор. Вбудовані інструменти для роботи з мок-сервісами, юніт-тестами та інтеграцією у TypeScript-екосистему роблять Ant Design Pro особливо затребуваним для enterprise-проєктів.

Підсумовуючи, React Admin оптимальний для систем управління даними із складною логікою та інтеграцією API; Tremor — для швидких, сучасних аналітичних дашбордів з акцентом на простоту й продуктивність; Ant Design Pro — для масштабних корпоративних платформ, де потрібен широкий вибір шаблонів, компонентів, гнучке налаштування інтерфейсу й готовність до enterprise-впровадження. Кожний з інструментів має активне ком’юніті, регулярні оновлення та хорошу документацію, що гарантує актуальність і стабільність для сучасних задач розробника.

2.5 Вимоги до системи

Для успішного функціонування аналітичної платформи необхідно врахувати функціональні, нефункціональні, апаратні та програмні вимоги.

Функціональні вимоги

Автоматичний збір та обробка освітніх даних

Система повинна автоматично збирати дані про навчальні досягнення студентів, включаючи результати виконання завдань, тривалість виконання, статус здачі та оцінки. Платформа має інтегруватися з LMS WazerCode для отримання актуальної інформації про курси, завдання та студентів у режимі реального часу.

Аналіз та прогнозування успішності

Система повинна аналізувати зібрані дані, застосовуючи методи інтелектуального аналізу даних, зокрема алгоритм 1-Rule для класифікації рівня успішності (High, Satisfactory, Unsatisfactory), метод Наївного Байеса для прогнозування результатів студентів та алгоритм Apriori для виявлення асоціативних правил між завданнями і курсами. Це дозволить визначати закономірності у навчальному процесі та надавати рекомендації викладачам.

Візуалізація ключових показників ефективності (KPI)

Система повинна забезпечувати наочну візуалізацію аналітичних даних через інтерактивні дашборди. Це включає графіки розподілу оцінок за курсами, діаграми успішності студентів, кластеризацію користувачів за рівнем досягнень та відображення KPI, таких як середній час виконання завдань, відсоток здачі вчасно і рівень складності завдань.

Гнучке управління доступом та ролями

Система повинна забезпечувати розмежування прав доступу для різних ролей користувачів (адміністратори, викладачі, студенти). Викладачі повинні мати можливість переглядати аналітику для своїх курсів, а адміністратори — доступ до загальної статистики платформи.

Нефункціональні вимоги

Мінімальне навантаження на ресурси

Система повинна бути розроблена таким чином, щоб не створювати зайвого навантаження на сервер під час збору та обробки великих обсягів освітніх даних. Це стосується оптимізації SQL-запитів до бази даних, ефективного використання кешування та застосування серверного рендерингу за допомогою Next.js 15 для зменшення навантаження на клієнтську частину.

Масштабованість

Система повинна бути масштабованою та готовою до обробки зростаючої кількості користувачів, курсів і завдань. Це включає можливість горизонтального масштабування бази даних Neon Postgres, використання CDN для статичних ресурсів через Vercel та підтримку мікросервісної архітектури для окремих аналітичних модулів.

Стабільність та надійність

Система повинна працювати стабільно за різних умов навантаження, не викликаючи збоїв чи втрати даних. Надійність роботи системи особливо важлива при виконанні аналітичних обчислень в реальному часі та інтеграції з зовнішніми сервісами.

Апаратні вимоги

Мінімальні апаратні вимоги для розробки та тестування

- Node.js: версія 20.9 або новіша (офіційна мінімальна для Next.js 15)
- Операційна система: Windows 10/11, macOS 12+ (M1/M2 і Intel), або сучасний дистрибутив Linux (Ubuntu 20.04+, Debian, Fedora, Arch)
- npm/pnpm/yarn: актуальна версія, залежно від менеджера пакетів
- Браузер: для розробки — Chrome 111+, Edge 111+, Firefox 111+, Safari 16.4+
- Git: для роботи з репозиторіями та CI/CDМобільні пристрої: Android 8.0 або новіше, мінімум 2 ГБ RAM, підтримка OpenGL ES 3.0 або Vulkan

Програмні вимоги

Платформа для розробки

- Специфічні вимоги по бібліотеках/інтеграціях
- Tailwind CSS: встановлюється як npm-пакет і не має додаткових вимог
- Shadcn UI: залежить від Tailwind CSS та React 19
- Better-Auth (Email OTP, GitHub OAuth): внутрішня бібліотека, потребує налаштування OAuth app на GitHub та SMTP серверу для email
- Arcjet Security: інтегрується через npm-пакет, працює з Node.js
- Stripe API: Node.js SDK, обліковий запис Stripe, ключі test/live
- Neon Postgres DB: Postgres 14+ (віддалена база), підключення за допомогою змінних середовища
- Prisma ORM: підтримує Node.js 18+, Postgres 14+
- Vercel: доступ до власного Vercel-акаунту, встановлений Vercel CLI для локального деплою

3 МЕТОДИ ТА ТЕХНОЛОГІЇ АНАЛІЗУ

3.1 Моделювання джерел даних

На рис.7 продемонстровано логічну модель даних аналітичної платформи для управління інформацією про навчальні результати студентів. Дана модель побудована за принципами реляційної бази даних і складається з кількох пов'язаних між собою сутностей (таблиць), кожна з яких виконує власну функцію та зберігає специфічний набір освітніх даних.

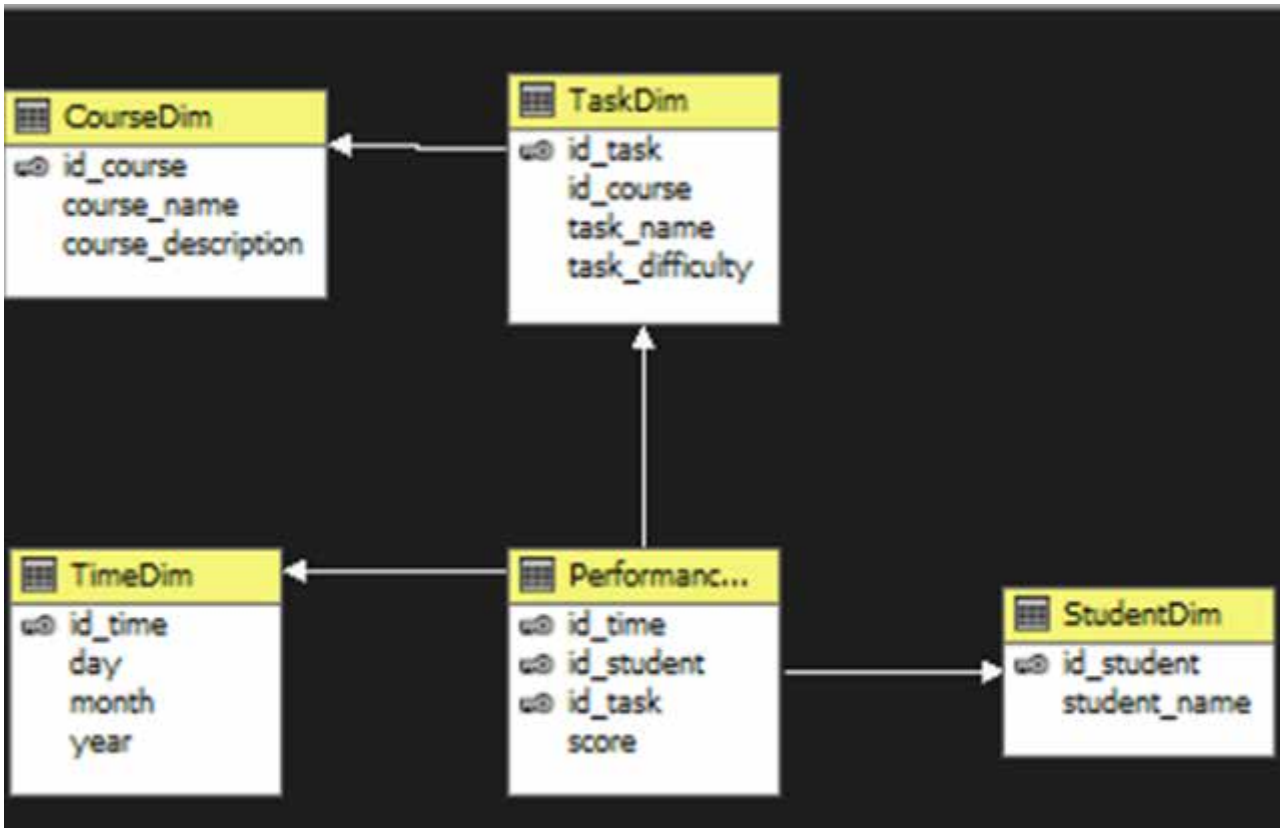


Рис. 7. Модель даних аналітичної платформи

Основні компоненти моделі:

1. CourseDim — таблиця курсу, яка містить унікальний ідентифікатор курсу, його назву та опис. Саме ця сутність є фундаментом, навколо якого організовується основна предметна структура освітнього процесу.

2. TaskDim — таблиця завдань, що містить ідентифікатор завдання, посилання на відповідний курс через зовнішній ключ, назву завдання та рівень

складності (task_difficulty). Це дозволяє фіксувати та аналізувати особливості завдань у межах кожної дисципліни.

3. StudentDim — таблиця студентів, яка містить унікальні ідентифікатори та імена студентів. Вона слугує для регістрації учасників навчального процесу та персоналізації освітньої аналітики.

4. TimeDim — таблиця календарного часу з деталізацією по днях, місяцях та роках. Дозволяє здійснювати хронологічний аналіз навчальної активності, будувати часові тренди та відстежувати динаміку освітніх досягнень.

5. Performance — фактова таблиця результатів, яка є центральною у структурі (fact table), містить інформацію про навчальні результати: ідентифікатор часу, студента, завдання та отриманий бал (score). Через зовнішні ключі ця таблиця зв’язана з усіма розмірними таблицями, забезпечуючи цілісність та інтегрованість моделі.

Вказані таблиці об’єднані зв’язками, що реалізують типову схему «зірка» (star schema), характерну для аналітичних і сховищних систем управління освітою. Кожна запис у таблиці Performance є агрегатом даних, які посилаються на конкретного студента, виконане завдання, дисципліну, і час, коли було отримано відповідний результат. Такі зв’язки забезпечують цілісний розгляд навчальних процесів з різних контекстних аспектів — предметних, персональних, часових та контексту завдання.

Описана структура бази даних є незамінною основою для впровадження сучасних аналітичних інструментів у навчальному закладі. Вона дозволяє забезпечити системний підхід до управління освітньою інформацією, підтримує транспарентність і оперативність аналізу, дає змогу отримувати комплексну картину навчального процесу для подальшого прийняття управлінських рішень.

Актуальність застосування такого підходу обумовлена розширенням обсягів освітніх даних і необхідністю підвищення ефективності навчання шляхом автоматизованої обробки, візуалізації і багаторівневого аналізу отриманих результатів. Така модель являє собою універсальний інструмент для побудови прозорих, гнучких, масштабованих освітніх платформ нового

покоління, що повністю відповідає сучасним вимогам цифровізації та персоніфікованого навчання у закладах освіти.

3.2 Метод аналізу 1-Rule

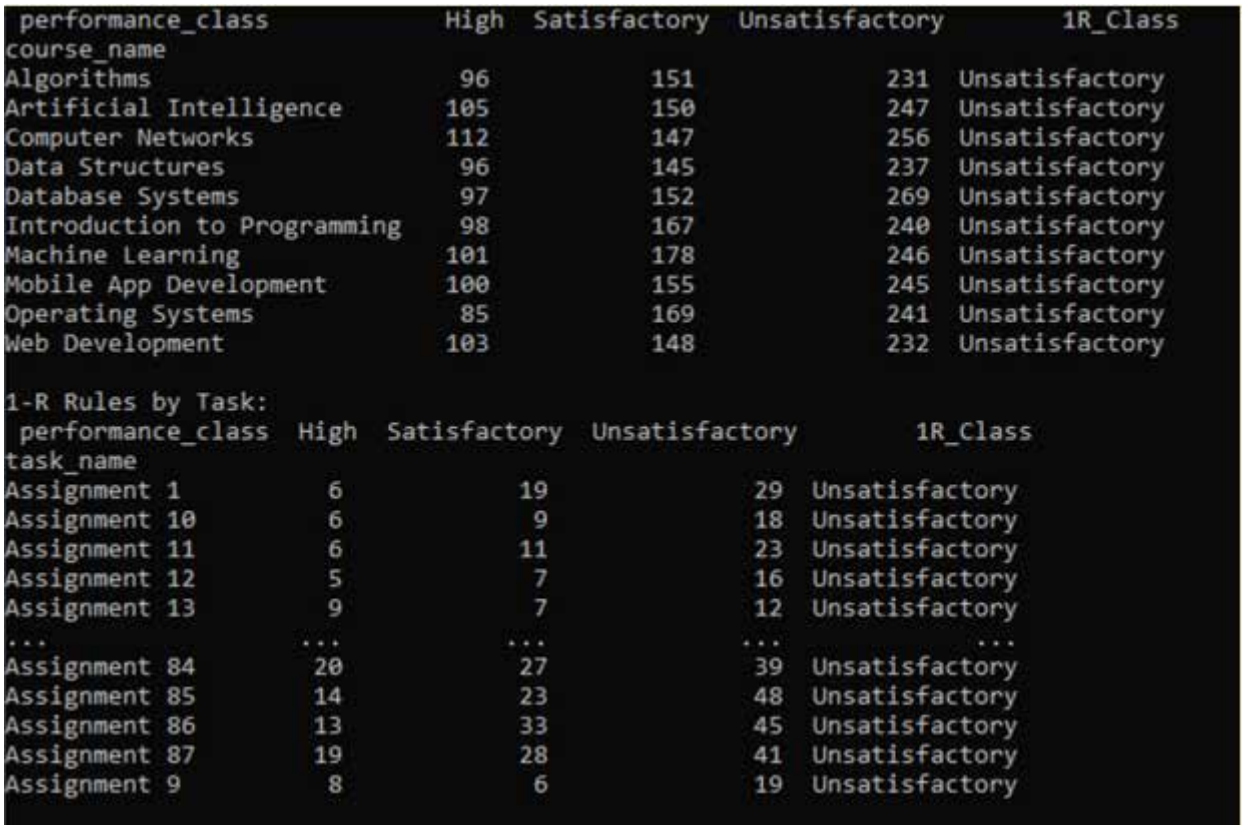
У межах дослідження, спрямованого на розробку аналітичної платформи для визначення досягнень здобувачів освіти, значну роль відіграє метод кластеризації, відомий як Правило 1-R. Цей метод закладений у концепцію обробки освітніх даних, що акумулюються в інформаційній системі WazerCode — платформі для автоматизованого збору, аналізу та візуалізації навчальної активності студентів у LMS (Learning Management System).

1-R (One Rule) — це метод кластеризації, що базується на виборі одного найкращого правила для поділу даних на класи. Метод виник у сфері машинного навчання як спрощений, але водночас ефективний підхід для класифікації об’єктів. Ідея полягає у тому, що серед усіх можливих одиночних правил, що базуються на одному атрибуті навчальних даних, обирається таке, яке найкращим чином поділяє сукупність об'єктів на кластери. Правило 1-R мінімізує помилки класифікації, забезпечуючи зрозумілу інтерпретацію класифікаційної моделі.

Застосування 1-R у контексті освітньої аналітики дозволяє розкрити закономірності в навчальній активності та академічних досягненнях здобувачів, використовуючи мінімальний набір показників. Це робить метод простим для втілення та ефективним інструментом для систем автоматичного оцінювання та виявлення груп ризику серед студентів.

За допомогою 1-R дані студентських досягнень розподіляються на кілька груп, які віддзеркалюють рівні якості навчальних результатів. Основні класи, що виділяються, включають групу з незадовільною успішністю ("Unsatisfactory"), а також інші класи з вищими результатами, які визначають різний рівень досягнень. Таким чином, метод 1-R дає змогу не лише автоматично відокремити студентів із різними рівнями досягнень, але й надає зрозумілі правила, які

пояснюють це розмежування. Такий підхід є особливо цінним у масових освітніх платформах, де ручне оцінювання є неможливим або неефективним.



performance_class	High	Satisfactory	Unsatisfactory	1R_Class
course_name				
Algorithms	96	151	231	Unsatisfactory
Artificial Intelligence	105	150	247	Unsatisfactory
Computer Networks	112	147	256	Unsatisfactory
Data Structures	96	145	237	Unsatisfactory
Database Systems	97	152	269	Unsatisfactory
Introduction to Programming	98	167	240	Unsatisfactory
Machine Learning	101	178	246	Unsatisfactory
Mobile App Development	100	155	245	Unsatisfactory
Operating Systems	85	169	241	Unsatisfactory
Web Development	103	148	232	Unsatisfactory

1-R Rules by Task:

performance_class	High	Satisfactory	Unsatisfactory	1R_Class
task_name				
Assignment 1	6	19	29	Unsatisfactory
Assignment 10	6	9	18	Unsatisfactory
Assignment 11	6	11	23	Unsatisfactory
Assignment 12	5	7	16	Unsatisfactory
Assignment 13	9	7	12	Unsatisfactory
...	...	...	...	...
Assignment 84	20	27	39	Unsatisfactory
Assignment 85	14	23	48	Unsatisfactory
Assignment 86	13	33	45	Unsatisfactory
Assignment 87	19	28	41	Unsatisfactory
Assignment 9	8	6	19	Unsatisfactory

Рис. 8. Результат роботи правила 1-R

Результати класифікації освітніх досягнень на основі правила 1-R представлені у вигляді таблиць що можна побачити на рис. 8, які відображають розподіл студентів за рівнями академічної успішності — «High», «Satisfactory» та «Unsatisfactory» — у межах окремих навчальних курсів та індивідуальних завдань. Аналіз засвідчив, що для кожної дисципліни, незалежно від тематики (наприклад, «Algorithms», «Artificial Intelligence», «Data Structures», «Mobile App Development» та інші), домінуючим класом за правилом 1-R є «Unsatisfactory».

Зокрема, кількість здобувачів із незадовільними результатами у кожному курсі значно перевищує представників груп «High» та «Satisfactory». Наприклад, у курсі «Algorithms» лише 96 студентів мають високий рівень, у той час як до категорії «Unsatisfactory» потрапляють 231 особа, а підсумкове значення класифікатора 1R_Class також позначено як «Unsatisfactory». Подібна закономірність спостерігається і в інших дисциплінах, що дозволяє зробити

висновок про системну наявність освітньої проблеми на рівні конкретних груп і предметів.

Аналогічна ситуація спостерігається при розгляді окремих завдань: майже для кожного завдання («Assignment 10», «Assignment 84», «Assignment 86» тощо) найбільша група студентів класифікована як «Unsatisfactory», що обумовлює і відповідний вибір кластеру за 1-R. Це свідчить про однорідність низької результативності як в межах індивідуальних активностей, так і на рівні комплексних курсів.

Такий розподіл є цінним аналітичним інструментом для забезпечення об'єктивного моніторингу освітнього процесу. Виявлення превалювання незадовільних результатів як на предметному, так і на завданнєвому рівнях дозволяє адміністрації та викладачам реалізувати комплексну стратегію підтримки студентів, спрямовану на підвищення якісних показників навчання. Таким чином, алгоритм кластеризації за правилом 1-R у системі WazerCode забезпечує не лише масову автоматизовану оцінку, але й формує підстави для ухвалення управлінських рішень щодо модернізації навчального контенту та посилення академічної підтримки, орієнтованої на потреби основної (найбільшої) групи здобувачів.

3.3 Метод аналізу Наївного Байеса

Наївний байєсівський класифікатор — це простий імовірнісний алгоритм машинного навчання, який базується на теоремі Байєса та припущенні незалежності ознак. Класифікатор використовує статистичні ймовірності для визначення, до якого з заданих класів належить об’єкт на основі значень його параметрів. Його простота, ефективність і здатність оперувати з багатовимірними даними роблять цей метод особливо поширеним для задач швидкої автоматизованої класифікації освітніх результатів.

=== Naive Bayes Predictions by Course + Task ===

	Course	Task	Predicted Class	P(High)	P(Satisfactory)	P(Unsatisfactory)
0	Introduction to Programming	Assignment 7	Unsatisfactory	16.67	39.12	44.22
1	Computer Networks	Assignment 54	Unsatisfactory	29.70	24.78	45.51
2	Web Development	Assignment 78	Unsatisfactory	25.72	33.39	40.89
3	Data Structures	Assignment 19	Unsatisfactory	22.49	32.39	45.12
4	Mobile App Development	Assignment 86	Unsatisfactory	14.39	36.01	49.60
5	Operating Systems	Assignment 46	Unsatisfactory	12.73	33.51	53.76
6	Algorithms	Assignment 28	Unsatisfactory	19.12	35.50	45.39
7	Artificial Intelligence	Assignment 74	Unsatisfactory	24.11	28.15	47.74
8	Computer Networks	Assignment 52	Unsatisfactory	23.07	30.07	46.86
9	Web Development	Assignment 76	Unsatisfactory	33.34	13.54	53.12
10	Data Structures	Assignment 14	Unsatisfactory	23.03	22.11	54.86
11	Database Systems	Assignment 39	Unsatisfactory	18.10	32.05	49.85
12	Introduction to Programming	Assignment 1	Unsatisfactory	10.82	37.13	52.05
13	Algorithms	Assignment 27	Unsatisfactory	12.98	33.74	53.29
14	Data Structures	Assignment 11	Unsatisfactory	15.14	26.65	58.21
15	Artificial Intelligence	Assignment 73	Unsatisfactory	25.17	25.72	49.11
16	Data Structures	Assignment 9	Unsatisfactory	24.38	17.55	58.07
17	Machine Learning	Assignment 65	Satisfactory	20.76	40.71	38.53
18	Database Systems	Assignment 41	Unsatisfactory	13.60	27.95	58.44
19	Computer Networks	Assignment 56	Unsatisfactory	22.94	25.12	51.93
20	Artificial Intelligence	Assignment 72	Unsatisfactory	16.61	39.44	43.95
21	Web Development	Assignment 77	Unsatisfactory	21.47	28.19	50.34
22	Algorithms	Assignment 24	Unsatisfactory	28.09	22.47	49.44
23	Web Development	Assignment 81	Unsatisfactory	16.47	31.46	52.07
24	Mobile App Development	Assignment 84	Unsatisfactory	23.41	31.14	45.44

Рис. 9 Результат роботи методу класифікації Наївного Байеса

На рисунку 9 представлено результати класифікації освітніх досягнень студентів із використанням алгоритму наївного байєсівського класифікатора, застосованого до комбінацій параметрів «Курс» та «Номер завдання». Модель на основі принципу умовної ймовірності розділила студентські роботи на три класи успішності — «High», «Satisfactory» та «Unsatisfactory», прогнозуючи кінцевий навчальний результат для кожного випадку.

У таблиці наведено 24 різних комбінації навчальних курсів («Introduction to Programming», «Computer Networks», «Web Development», «Data Structures», «Machine Learning» тощо) та номери завдань, для яких виконано прогнозування класу успішності. Для кожного рядка вказано передбачений клас (Predicted Class) і три ймовірнісні показники: P(High), P(Satisfactory), P(Unsatisfactory), які відображають рівень впевненості моделі в належності конкретної комбінації до кожної з освітніх категорій.

Аналіз отриманих даних свідчить, що у переважній більшості випадків модель надає максимальну ймовірність класу «Unsatisfactory» (незадовільний), що відповідає загальній тенденції низької академічної успішності серед студентів для проаналізованих завдань. Лише один запис (Machine Learning,

Assignment 65) ідентифіковано як «Satisfactory», що може вказувати як на особливості контенту курсу, так і на відносну підготовленість окремих студентів чи групи.

Використання наївного байєсівського класифікатора дозволяє не лише здійснити розподіл навчальних результатів за класами, а й формалізувати процес прийняття рішень з урахуванням умовних ймовірностей. Отримані результати актуалізують необхідність системного аналізу навчальних програм і індивідуальних завдань, а також забезпечують підґрунтя для реструктуризації навчального контенту у напрямку підвищення якості освіти та розробки цільових заходів підтримки найменш успішних груп студентів.

3.4 Метод асоціативних правил

Метод виявлення асоціативних правил (англ. Association Rule Learning) є одним із ключових методів інтелектуального аналізу даних, що дозволяє виявляти приховані залежності та закономірності при роботі з великими обсягами освітньої інформації. Основною метою застосування цього методу в контексті системи управління навчанням є виявлення зв'язків між різними навчальними активностями, курсами, завданнями та результатами студентів на основі моделей їх спільного виникнення. У платформі WazerCode цей підхід забезпечує автоматичне виявлення кореляцій, які не завжди є очевидними для викладачів, але можуть мати критичне значення для розуміння навчального процесу та його оптимізації.

Алгоритм Apriori, який використовується для генерації асоціативних правил, працює у два етапи: спочатку визначаються всі часті набори предметів у базі даних з використанням порогового значення мінімальної підтримки (support), а потім до цих наборів застосовуються обмеження на мінімальну впевненість (confidence) для формування правил. Перший крок є найбільш обчислювально складним, оскільки включає пошук усіх можливих комбінацій, проте ефективність досягається завдяки властивості спадного змикання, яка гарантує, що всі підмножини частих наборів також є частими. Асоціативні

правила надають цінну інформацію для різних сфер освітнього процесу, включаючи аналіз навчальних траєкторій, сегментацію студентів та формування персоналізованих рекомендацій щодо вибору курсів. З демонстрацією роботи методу можна ознайомитись на рис. 12

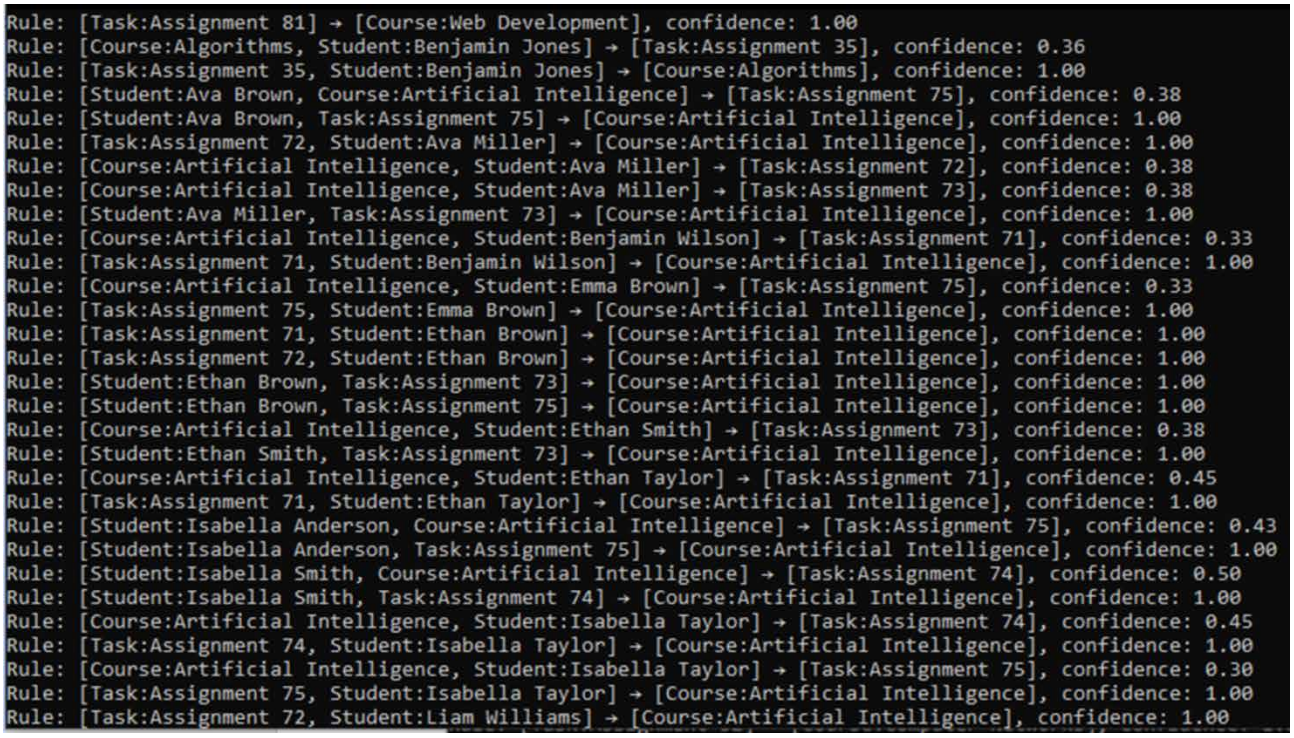


Рис. 10 Метод асоціативних джерел

На представленому рис. 10 відображено результати виявлення асоціативних правил для курсу "Artificial Intelligence" (Штучний інтелект) з використанням даних про виконання студентами різних завдань (Assignment). Кожне правило має структуру виду "якщо... то...", де ліва частина (антецедент) вказує на умови, а права частина (консеквент) показує очікувані наслідки. Наприклад, правило "[Task:Assignment 81] → [Course:Web Development], confidence: 1.00" означає, що всі студенти (100% впевненість), які виконали завдання 81, також зареєстровані на курс Web Development, що може вказувати на послідовне вивчення цих дисциплін.

Метрика впевненості (confidence) для кожного правила відображає ймовірність того, що консеквент буде істинним за умови істинності антецедента, і варіюється від 0.33 до 1.00 у представленій вибірці. Високі значення

впевненості (1.00) спостерігаються для правил типу "[Task:Assignment 75, Student:Emma Brown] → [Course:Artificial Intelligence]", що підтверджує детерміністичний зв'язок між виконанням конкретних завдань та зарахуванням на відповідний курс. Правила з нижчою впевненістю (0.33-0.50) вказують на ймовірнісні залежності, наприклад, "[Course:Algorithms, Student:Benjamin Jones] → [Task:Assignment 3s], confidence: 0.36" демонструє, що лише 36% студентів з курсу Algorithms з іменем Benjamin Jones виконують завдання 3s. Аналіз представлених правил дозволяє виділити декілька основних категорій асоціацій у навчальному процесі. Перша категорія – це зв'язки між завданнями та курсами, які демонструють, що виконання певних завдань є характерним для студентів конкретних дисциплін, як у правилі "[Task:Assignment 72, Student:Liam Williams] → [Course:Artificial Intelligence], confidence: 1.00". Друга категорія представлена правилами, що пов'язують студентів з курсами та завданнями одночасно, наприклад, "[Student:Ava Brown, Course:Artificial Intelligence] → [Task:Assignment 75], confidence: 0.38", що може вказувати на типові навчальні траєкторії для певних груп здобувачів.

Виявлені асоціативні правила надають можливість ідентифікувати типові навчальні траєкторії та оптимізувати структуру курсів у системі WazerCode. Правила з високою впевненістю можуть використовуватися для формування рекомендацій щодо послідовності вивчення дисциплін. Система може автоматично рекомендувати студентам наступні завдання або курси на основі їхньої поточної активності та виявлених патернів успішних студентів. Правила з низькою впевненістю можуть сигналізувати про проблемні завдання або розриви в навчальній програмі, які потребують уваги викладачів.

Асоціативні правила з низькою впевненістю можуть вказувати на завдання, що недостатньо інтегровані у навчальний процес. Порівняння правил для різних студентів дозволяє ідентифікувати альтернативні шляхи досягнення навчальних результатів та виявити критичні versus опціональні завдання. Систематичний аналіз асоціативних правил протягом декількох навчальних

періодів дозволяє відстежувати динаміку навчальних патернів та оцінювати ефективність внесених змін до структури курсу. Інтеграція з іншими методами інтелектуального аналізу даних, такими як кластеризація студентів, забезпечує комплексний підхід до управління якістю освіти в LMS.

3.5 Метод кластеризації користувачів

Кластерний аналіз у сучасних системах управління навчанням (LMS) виступає потужним інструментом для сегментації здобувачів освіти та оптимізації педагогічних стратегій. Основною метою застосування кластеризації є формування однорідних груп студентів на основі схожих характеристик навчальної діяльності, що дозволяє виявити споживацьку поведінку та адаптувати освітній процес до потреб кожного кластера. У контексті аналітичної платформи WazerCode цей метод забезпечує автоматизовану класифікацію без навчання, що є особливо цінним при роботі з великими масивами даних про академічну успішність.

Використання кластерного аналізу дозволяє ідентифікувати групи студентів із різними рівнями засвоєння матеріалу та своєчасно впроваджувати диференційований підхід до навчання. Дослідження показують, що системи дистанційного навчання з індивідуальним підходом, які базуються на результатах кластеризації, демонструють збільшення кластера успішних студентів порівняно з традиційними методами викладання. Методика кластерного аналізу базується на концепції подібності об'єктів за визначеними ознаками, що дозволяє розподілити сукупність здобувачів на стійкі, однорідні групи для подальшого стратегічного планування освітніх траєкторій. З діаграмою кластерів можна ознайомитись на рис. 11.

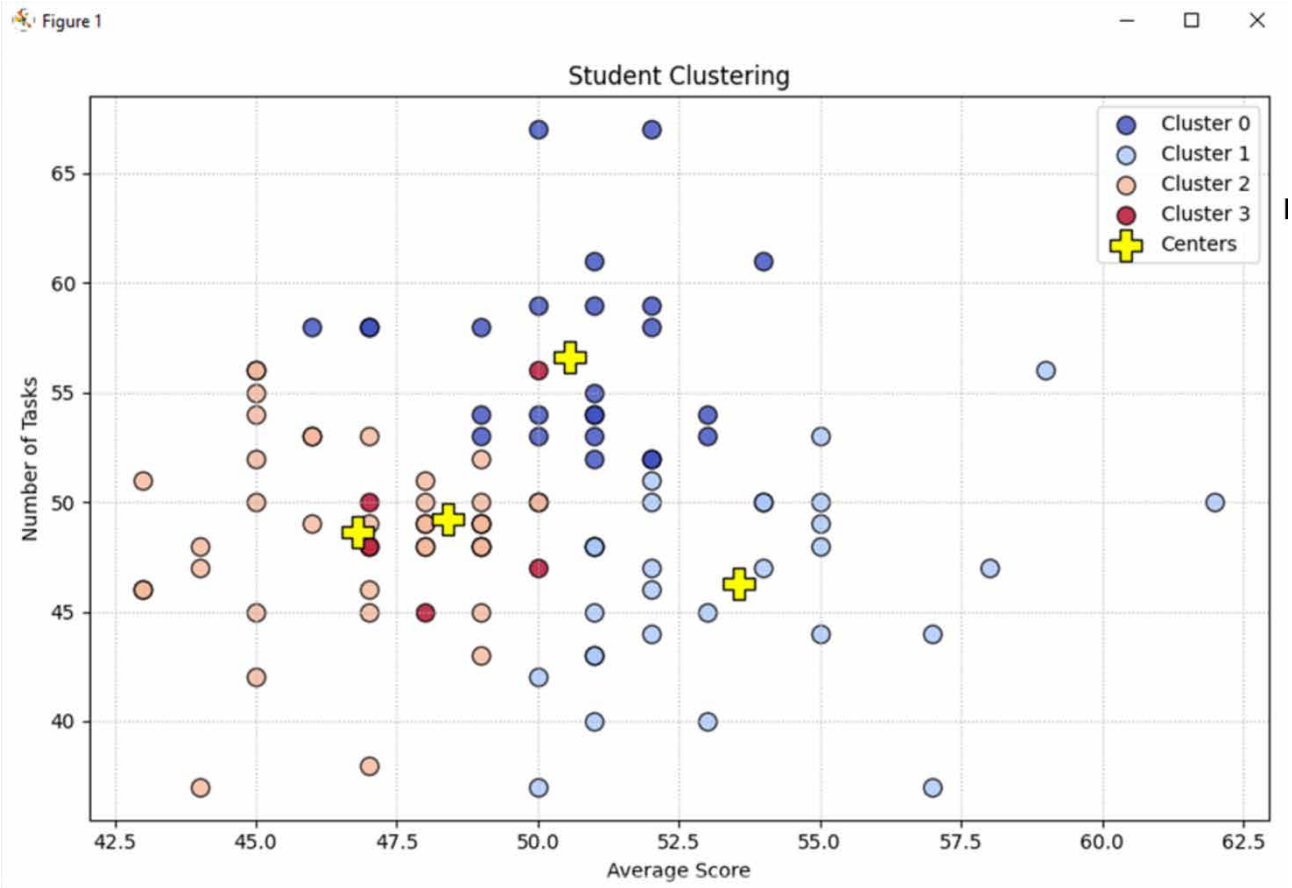


Рис. 11 Кластеризація користувачів

На представленій діаграмі відображено результати кластеризації здобувачів освіти за двома ключовими показниками академічної успішності: середнім балом (горизонтальна вісь, діапазон 42-63 бали) та кількістю виконаних завдань (вертикальна вісь, діапазон 37-67 завдань). Візуалізація демонструє застосування методу k-середніх, який є найпоширенішим неієрархічним алгоритмом кластерного аналізу, що забезпечує ефективне розбиття сукупності на заздалегідь визначену кількість кластерів. Вибір саме цих двох метрик обґрунтовується їхньою здатністю комплексно відображати навчальну активність студента: середній бал характеризує якість засвоєння матеріалу, тоді як кількість виконаних завдань демонструє залученість та систематичність роботи.

Процес кластеризації здійснюється у п'ять послідовних етапів: формування вибірки для аналізу, вибір характеристичних ознак об'єктів, розрахунок міри подібності (відстані) між об'єктами, формування кластерів та аналіз отриманої

інформації. Для визначення міри подібності між студентами використовується Евклідова відстань, яка є геометричною відстанню у двовимірному просторі ознак і піддається обчисленню незалежно від типу запису даних. Математично Евклідова відстань між двома точками визначається як корінь квадратний із суми квадратів різниць координат, що дозволяє об'єктивно оцінити схожість навчальних профілів студентів.

Алгоритм виділив чотири основні кластери (Cluster 0-3), позначені різними кольорами на діаграмі розсіювання, що відповідає оптимальній сегментації для педагогічного втручання. Кластер 0 (темно-синій колір) представляє здобувачів з найвищими показниками як за середнім балом (переважно 50-53 бали), так і за кількістю виконаних завдань (55-67 завдань), що характеризує цю групу як високомотивованих та систематичних студентів. Кластер 1 (світло-синій) об'єднує студентів із середнім рівнем успішності, що характеризується середніми балами у діапазоні 50-62 та помірною кількістю виконаних завдань (40-56), демонструючи нерівномірний розподіл зусиль у навчальному процесі.

Кластер 2 (помаранчевий) охоплює здобувачів з нижчими показниками за обома метриками, з середніми балами 43-50 та кількістю завдань 38-56, що вказує на необхідність додаткової педагогічної підтримки для цієї групи. Кластер 3 (червоний) є найменшим за кількістю членів і характеризується специфічним патерном: помірна кількість виконаних завдань при нижчих середніх балах, що може свідчити про труднощі у засвоєнні матеріалу попри регулярну участь у навчальних активностях. Така детальна сегментація дозволяє педагогам розробити цільові стратегії підтримки для кожної групи, від додаткового стимулювання для високоефективних студентів до інтенсивної методичної допомоги для тих, хто відстає. Програмний вигляд кластеризації можна побачити на рис 12.

```
Optimal number of clusters: 4

Clustered Students:
    id_student  avg_score  task_count  course_count  Cluster
0           23         46          49            10        2
1           46         54          50            10        1
2           69         52          50            10        1
3           92         47          58            10        0
4           29         51          52            10        0
..         ...        ...         ...           ...      ...
95          14         45          50            10        2
96          37         52          52            10        0
97           8         51          55            10        0
98          51         50          42            10        1
99         100         51          43            10        1

[100 rows x 5 columns]

Cluster Centers:
   avg_score  task_count  course_count  Cluster
0  50.560000   56.640000          10.0        0
1  53.551724   46.275862          10.0        1
2  46.804878   48.658537          10.0        2
3  48.400000   49.200000           9.0        3
```

Рис. 12 Програмна реалізація кластеризації

Повертаючись до діаграми кластеризації, на ній зображені жовті хрестоподібні маркери на діаграмі позначають центроїди – що позначають центральні точки кожного кластера, що представляють середні значення показників для всіх членів відповідної групи та виступають репрезентативними "портретами" кожного сегменту. Позиціювання центроїдів демонструє чітку просторову сепарацію кластерів: центроїд Cluster 0 розташований у верхній правій області графіка (високий середній бал, велика кількість завдань), тоді як центроїд Cluster 2 знаходиться у лівій середній зоні (нижчий бал, менша активність). Значення F-статистики для кожної з двох вимірювальних змінних виступає індикатором того, наскільки ефективно відповідна метрика дискримінує кластери між собою.

Спостерігається помірне перекриття між деякими кластерами у середній області графіка, що є природним явищем при роботі з реальними освітніми даними та відображає континуальність навчальних досягнень. Водночас чіткість основних кластерів підтверджує адекватність вибраної кількості груп (k=4) та валідність обраних метрик для диференціації здобувачів освіти. Розподіл точок навколо центроїдів характеризується різною щільністю: Cluster 1 демонструє найбільшу дисперсію, що вказує на гетерогенність цієї групи та потенційну необхідність подальшої субсегментації.

3.6 Розрахунок KPI

Ключові показники ефективності (KPI, Key Performance Indicators) становлять собою системно визначені та формалізовані метрики, які відображають якість і результативність освітніх процесів, а також динаміку розвитку академічного середовища. В освітній аналітиці KPI виконують функцію об’єктивного інструменту оцінювання, оптимізації та управління навчальними траєкторіями на різних організаційних рівнях – від окремого курсу, групи чи студента до всього закладу освіти. Запровадження системи KPI забезпечує науково обґрунтований зворотний зв’язок для моделювання освітніх стратегій, підтримки прийняття рішень і моніторингу результатів впровадження методичних інтервенцій.

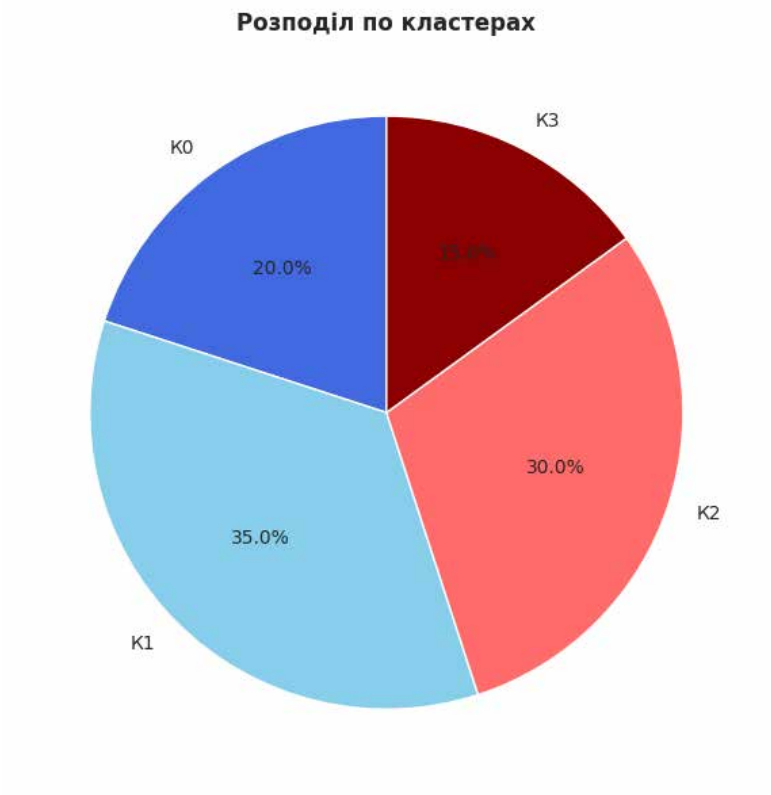


Рис 13. KPI розподіл по кластерам

У представленому розподілі студенти згруповані у чотири основних кластери: К0, К1, К2, К3. Частка кожного з кластерів позначена у відсотковому співвідношенні, що дозволяє оцінити структурну композицію навчального середовища. Зокрема, найбільшу питому вагу має кластер К1 — 35%, який об’єднує найчисельнішу групу здобувачів. Далі йде К2 — 30%, К0 — 20% та К3 — 15%.

Структурне домінування одного кластера над іншими (як К1 у цьому випадку) сигналізує про наявність провідної освітньої групи, на яку орієнтується навчальний процес. Водночас існування менш чисельних кластерів (К3 — 15%) вимагає від викладачів уваги до проблемних зон — зокрема, для ідентифікації причин відставання, адаптації освітніх стратегій чи розробки індивідуальних освітніх траєкторій.

На основі роботи з окремими кластерами платформа може автоматично пропонувати індивідуалізовані стратегії, враховувати особисті освітні потреби, особливості сприйняття матеріалу, стиль мислення та мотиваційні чинники.

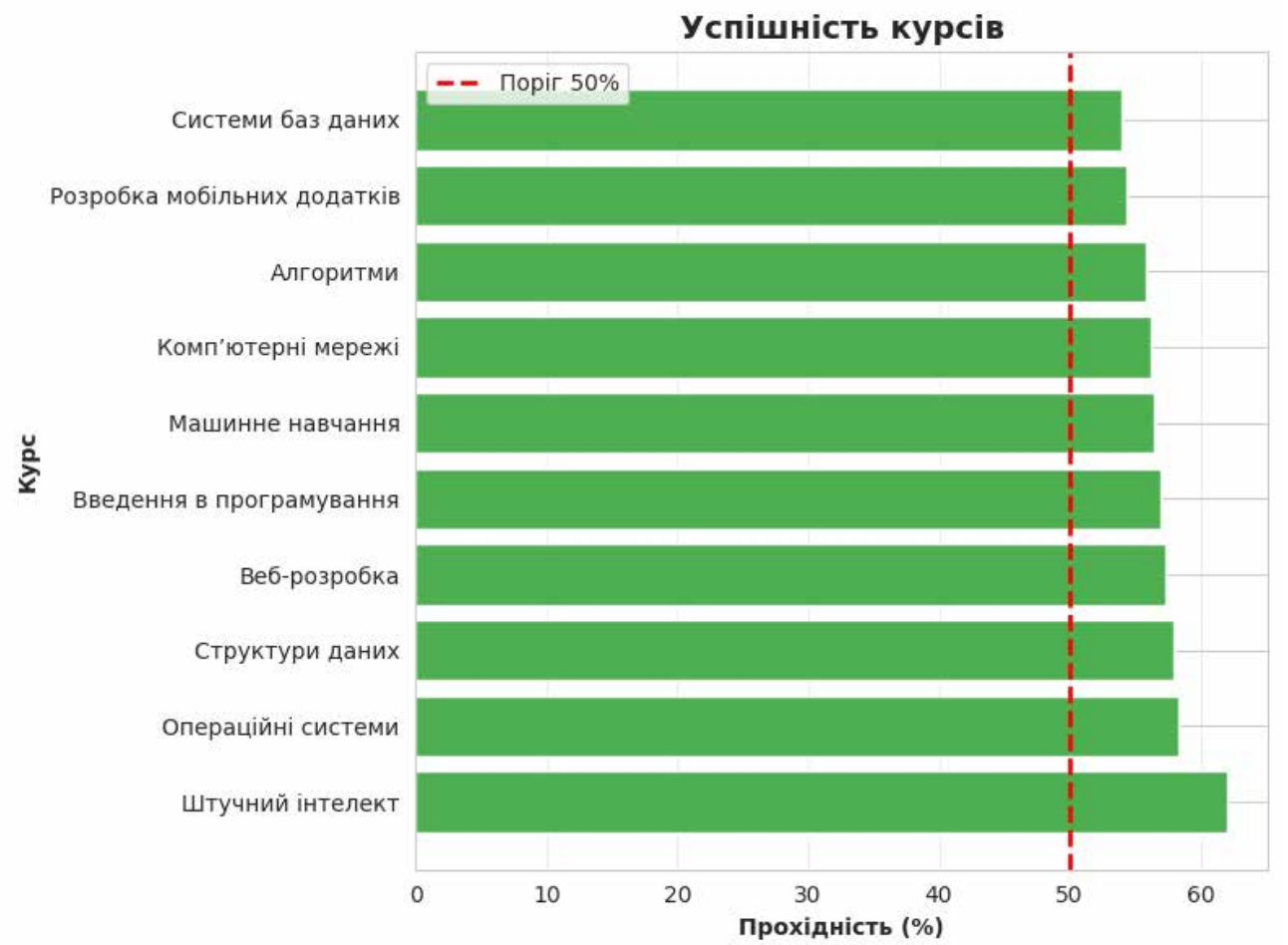


Рис 14. KPI успішність по курсам

Представлений на рис.14 графік відображає показник прохідності (успішності) за різними освітніми курсами у відсотковому вираженні. Аналіз даного KPI є ключовим для комплексної оцінки якості освітнього процесу та прийняття рішень щодо оптимізації змісту й методики навчання. У діаграмі горизонтальні смуги представляють значення прохідності для кожного окремого курсу — тобто відсоток студентів, які успішно опанували зміст дисципліни та виконали всі передбачені програмою завдання.

Ключовий показник ефективності для курсу — це відношення кількості студентів, які успішно завершили курс, до загальної кількості тих, хто розпочинав навчання. У контексті представленої реалізації до категорії “успішно” можуть належати студенти з категоріями High та Satisfactory, тобто весь масив даних поділяється за порогами оцінювання.

Реалізація цього алгоритму виконувалась за допомогою SQL-запитів до фактологічних даних, агрегування за курсами та обрахунку відповідних метрик.

Кожний бар (курс) ілюструє частку студентів, які подолали поріг успішності (наприклад, 50% чи інша обрана межа), а пунктирна лінія відображає нормативний орієнтир для якісної порівняльної оцінки. Саме таку форму має графік із попереднього кроку — він демонструє, що для жодного з курсів не було зафіксовано системної неуспішності, однак розподіл свідчить про наявність варіацій між навчальними напрямками.

Застосування такого підходу забезпечує валідність і відтворюваність результатів, а використання реляційної моделі й SQL запитів гарантує масштабованість для різних вибірок і сценаріїв навчального процесу. Методика дозволяє гнучко визначати критерії успішності для окремих курсів (наприклад, різний поріг для складних чи вибіркових дисциплін) та швидко ідентифікувати зони ризику чи потреби у педагогічних змінах.

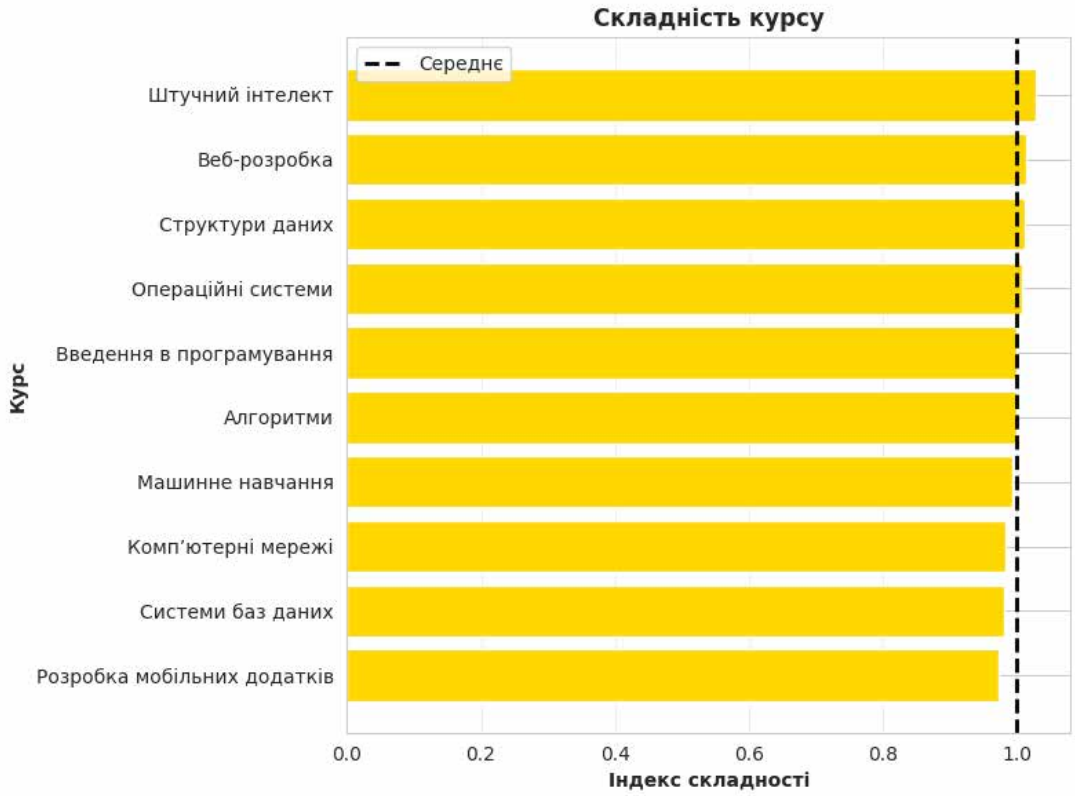


Рис 15. KPI складності курсів

Діаграма на рис. 15 відображає ключовий аналітичний показник — індекс складності курсу, що характеризує рівень навчального навантаження та складність засвоєння матеріалу для кожної дисципліни. В академічному контексті такий інтегральний індекс дозволяє порівнювати курси між собою, виявляти баланс програм, а також оперативно реагувати на перевантаження навчального процесу.

У реалізованій системі індекс складності формується на основі мультифакторного аналізу навчальних даних, що генеруються для студентів кожної дисципліни. В структурованій базі AssignmentDim та PerformanceFact накопичуються показники:

- кількість завдань на курс;
- середній бал студентів і їх розподіл по категоріях успішності (High, Satisfactory, Unsatisfactory);
- кількість спроб та тривалість виконання завдань (days_to_submit);
- відсоток студентів, що завершили всі завдання;
- співвідношення студентів у різних кластерах за результатами.

Для обрахунку індексу складності застосовується нормалізація показників у проміжку, де 1 — максимальна складність серед усіх курсів, а 0 — мінімальна.

На діаграмі видно, що майже всі курси мають високий індекс складності (відносно середньої лінії/«порогу») — це свідчить про узгоджену структуру навчального навантаження та відсутність курсів із аномально низьким рівнем вимог. «Штучний інтелект», «Веб-розробка», «Структури даних», «Операційні системи» тощо демонструють максимально близькі до 1 індекси. Важливим технічним результатом є можливість адаптивної модифікації навчальної програми: при зміні кількості завдань, критеріїв оцінювання, часу на виконання курс швидко змінює значення свого індексу і відповідно впливає на загальний баланс навчального плану.

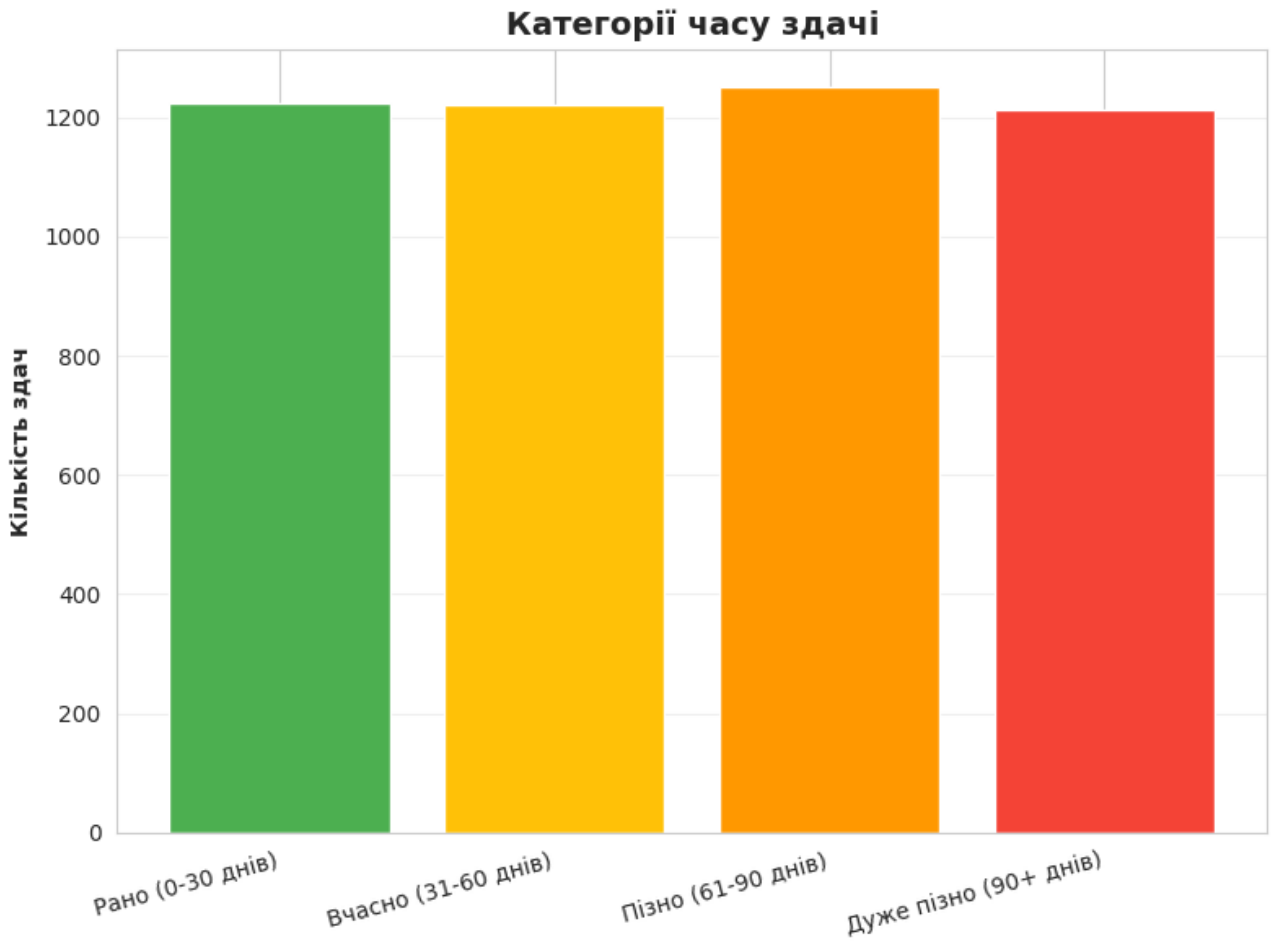


Рис 16. KPI категорії часу завантаження задачі

Дана діаграма ілюструє розподіл кількості задач, виконаних студентами, за категоріями часу здачі. У контексті аналітики системи студентських даних подібний розподіл має суттєве значення для оцінки дисциплінованості, організованості, а також характеру поведінкових патернів навчального процесу.

Графік містить чотири категорії:

1. “Рано (0–30 днів)”
2. “Вчасно (31–60 днів)”
3. “Пізно (61–90 днів)”
4. “Дуже пізно (90+ днів)”.

Всі категорії мають приблизно однакову наповненість (близько ~1200 задач у кожній), що свідчить про відсутність яскраво виражених екстремальних тенденцій щодо термінів виконання завдань.

У базі даних PerformanceFact кожен факт здачі завдання містить дату подачі (submission_date) та кількість днів до здачі (days_to_submit), обраховану як різницю між датою призначення й датою виконання. Розподіл по категоріях отримується SQL-агрегуванням. Результати агрегуються у категорії, для кожної з яких виводиться кількість виконаних задач.

Переважання категорій “Рано” та “Вчасно” і відсутність різких перепадів — позитивна ознака, яка характеризує більшість студентів як організованих і відповідальних. Водночас значна частка задач із категорій “Пізно” та “Дуже пізно” вказує на наявність проблем з тайм-менеджментом, можливо перевантаженням, або на недосконалості навчальної стратегії для окремих груп.

4 ІНТЕГРАЦІЯ ТА ВДОСКОНАЛЕННЯ WAZERCODE

4.1 Вибір та впровадження сучасних інструментів веб-розробки для LMS

Для реалізації покращення та реалізації додаткових можливостей основної платформи був обраний фреймворк Next.js v15, який підтримує серверний рендеринг та підтримує швидке розгортання проекту. Він став базовим для розробки додатків, які розраховують на своє кратне збільшення та подальшу популяризації за рахунок зручностей, які він надає розробнику.

Так як ми працюємо з різними бібліотеками та розширеннями, для збірки проекту потрібно використовувати збірник веб додатку, на який покладено оптимізацію, виведення часу роботу коду на новий рівень. Саме цим займається Turbopack, який став на заміну іншого популярного рішення – Webpack, що втратив свої позиції, після того як Turbopack інтегрувався у Next.js v15.

За рахунок підтримки React v19 – ми можемо використовувати новітні хуки, які були використані при побудові цього проєкту. Вони використовуються для асинхронного використання даних, що є потрібним для побудови аналітичної платформи з метою збільшення її продуктивності.

На проекті також впроваджено esLint, що дає можливість виконувати статичний аналіз коду, знаходити невідповідності, помилки, і перевіряє код на узгоджений стиль програмування проєктів, який використовує глобальна індустрія, що зменшує кількість помилкових призначень, та зменшує прогалини в безпеці додатку, який піде в Production.

Також, Next.js дає можливість розробнику краще слідкувати за кешуванням. Це окрема тема, вона буде розглянута в наступних пунктах моєї роботи.

Окремо варто згадати про удосконалення механізмів кешування у Next.js, що дозволяє оптимізувати швидкодію платформи та мінімізувати час завантаження сторінок для кінцевого користувача. Завдяки гнучкій системі

кешування запити до серверу зменшуються, а статистичні дані та основний контент стають доступними швидше.

У фокусі розробки також знаходиться питання безпеки: проєкт використовує актуальні підходи до забезпечення захисту даних користувачів, у тому числі—ролеву модель доступу на рівні додатку та бази даних, що дозволяє уникати несанкціонованого доступу до важливої інформації без зайвої деталізації щодо інфраструктури.

Візуальна частина платформи також не залишилась без уваги. Впроваджено систему змінних тем оформлення (світла, темна теми). Також реалізовано простий редактор для створення власних освітніх матеріалів і завдань, що дозволяє викладачам швидко актуалізувати контент, не залучаючи технічних спеціалістів.

Подальший розвиток платформи передбачає розширення функціоналу аналітичних модулів, інтеграцію з новими освітніми сервісами та поступове вдосконалення користувацького досвіду на основі технологій React і Next.js.

4.2 Архітектура DevOps та інфраструктури

У процесі організації середовища розробки для масштабованого веб-додатку було використано файл конфігурацій .env, що містить усі чутливі параметри: строки підключення до бази даних, ключі доступу до платіжних сервісів, параметри автентифікації, дані для взаємодії з хмарною інфраструктурою AWS та іншими сторонніми сервісами. Така модель дозволяє централізовано керувати конфігураціями та змінювати поведінку системи залежно від середовища (development, staging, production) без ризику ненавмисного розкриття секретів.

Усі ключові значення (на кшталт DATABASE_URL, AWS_ACCESS_KEY_ID, STRIPE_SECRET_KEY, інші API-ключі) ізольовано від вихідного коду, а їх ротація та актуалізація здійснюється централізовано — це підвищує рівень безпеки додатку та унеможливлює витік конфіденційної інформації.

DevOps-підхід базується на автоматизації основних етапів розробки та розгортання (Continuous Integration/Continuous Delivery — CI/CD). Для цього були впроваджені наступні практики:

Все оновлення коду, після внесення змін до репозиторію, автоматично підлягають перевірці за допомогою лінтингу (наприклад, через esLint) та юніт-тестів.

Побудова проєкту, запуск тестів та деплой у тестове або продукційне середовище виконуються через автоматизовані пайплайни, які отримують необхідні секрети з менеджера секретів або файлів .env.

При невдалих змінах CI/CD система здійснює відкат на попередню стабільну версію, а логування процесу дає змогу швидко ідентифікувати причину помилок чи збоїв.

Демонстрацію виконання CI/CD пайплайнів та функціонал логування можна побачити на рис. 7 та 8 відповідно.

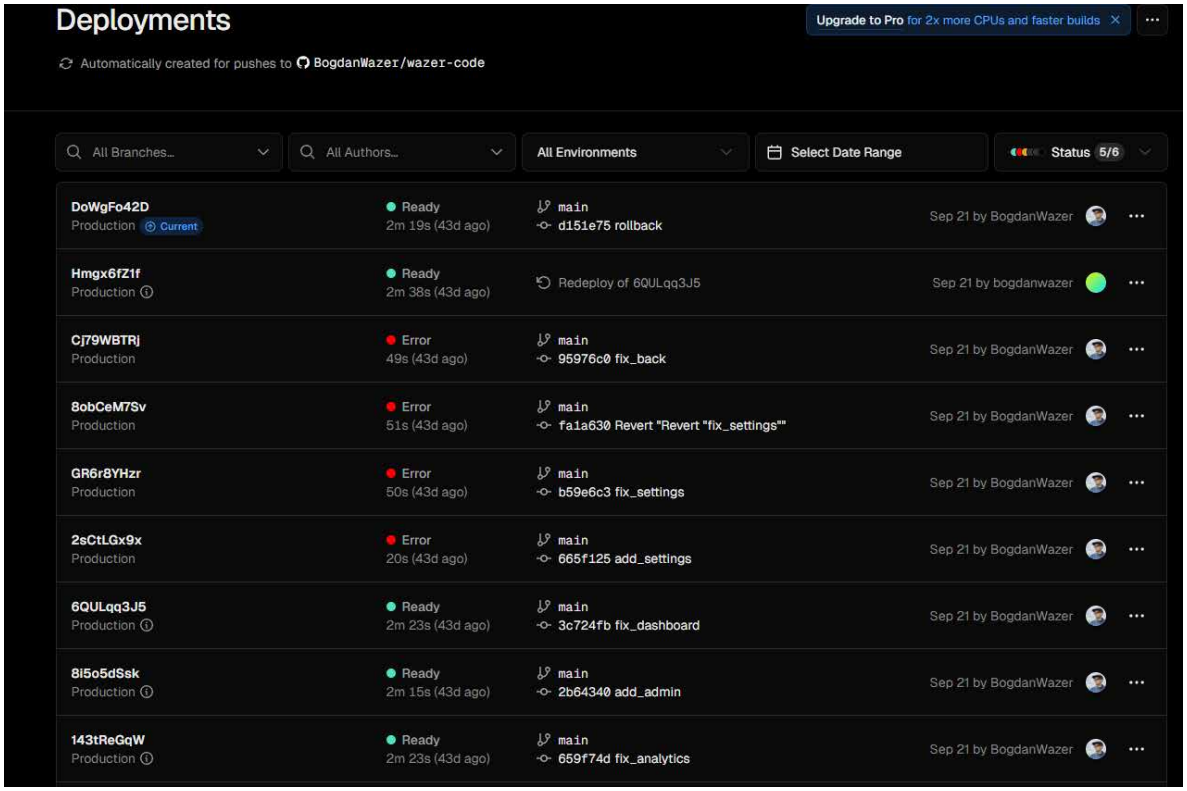


Рис 17. CI/CD панель на хмарній платформі Vercel

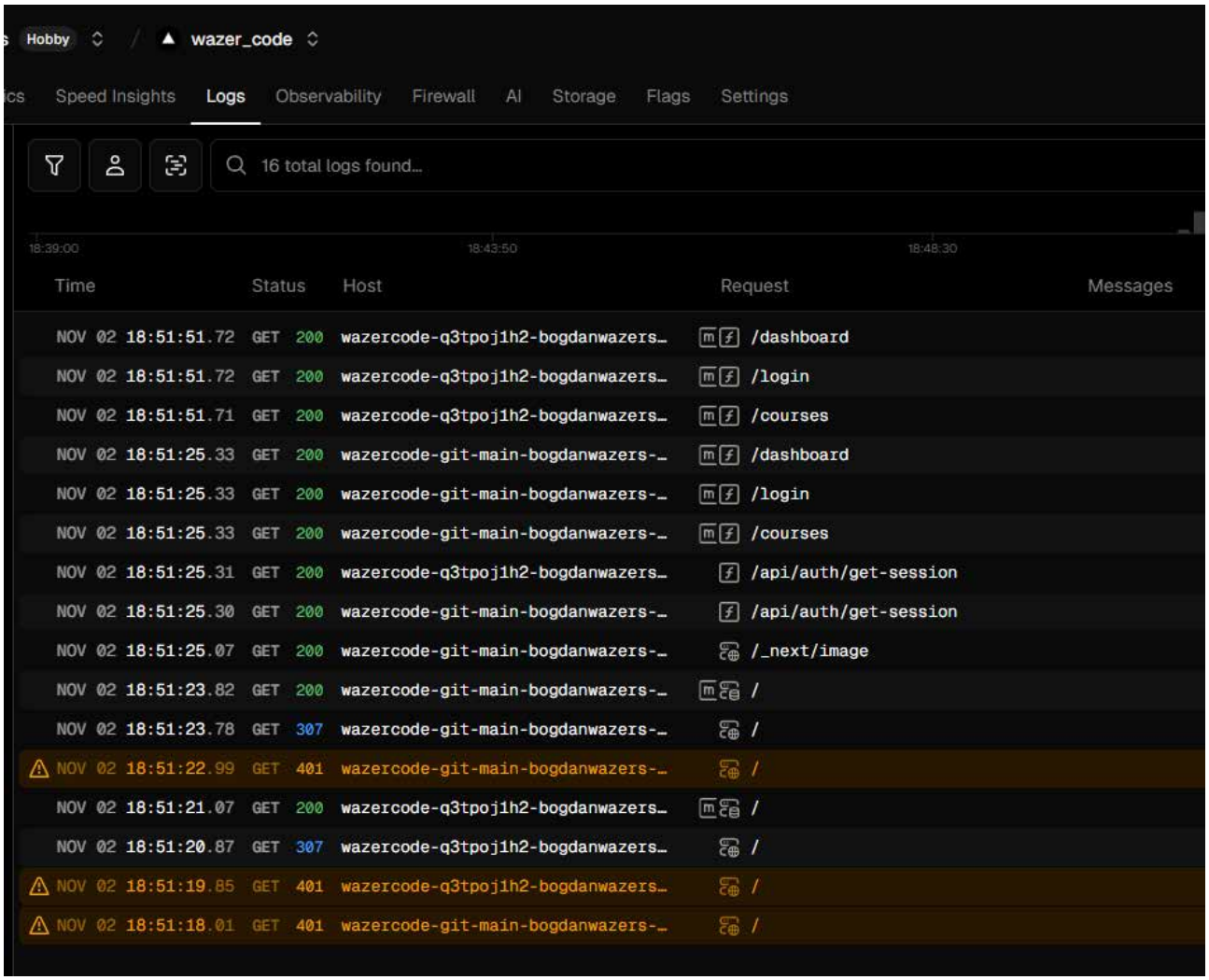


Рис 18. Система логування на хмарній платформі Vercel

Завдяки автоматизації деплою забезпечується швидка поставка оновлень, мінімізується людський фактор, а кожна релізна ітерація стає прозорою та контрольованою.

Особливу роль в архітектурі додатку займає хмарна інфраструктура Amazon Web Services (AWS), яка інтегрується через відповідні змінні середовища та забезпечує надійне, гнучке і масштабоване робоче середовище для ключових компонентів платформи. Використання значень AWS дозволяє ефективно підключати додаток до необхідних ресурсів у хмарі, при цьому забезпечуючи глибокий контроль над всіма аспектами доступу і конфігурації.

Зокрема, для зберігання і обробки великої кількості медіа-даних, таких як зображення, документи, відео- та презентаційні матеріали студентів та викладачів, використовується сервіс Amazon S3. Параметр

NEXT_PUBLIC_S3_BUCKET_NAME_IMAGES визначає конкретний бакет для розміщення файлів, а доступ до нього регламентується переданими ключами та ендпоінтами. Така схема дозволяє не лише швидко масштабувати дисковий простір згідно потреб освітнього процесу, а й автоматизувати процес бек-апу та версіонування контенту без втручання розробника. Кожний файл, який завантажується через платформу, автоматично проходить валідацію доступу і зберігається у відповідній структурі, захищеній від несанкціонованих змін.

Інтеграція з AWS IAM через параметр AWS_ENDPOINT_URL_IAM дає змогу гнучко керувати політиками доступу до ресурсів, створювати окремі ролі для адміністратора, викладача, студента або автоматизованих сервісів, обмежуючи права лише необхідними для виконання конкретних задач. Завдяки цьому знижується ризик випадкової або навмисної компрометації даних, обмежується доступ до найдовіреніших ділянок системи на основі принципу "мінімальних привілеїв". Усі операції з AWS проходять через зашифровані канали, що підвищує рівень інформаційної безпеки платформи.

Використання AWS також відкриває можливість для автоматизації резервного копіювання та відновлення даних — AWS S3 та пов’язані сервіси дозволяють налаштовувати регулярне створення snapshots, зберігати історію змін та швидко відновлювати дані у разі технічних збоїв чи втрати доступу. Це критично важливо для стабільності освітнього процесу — інформація студентів та викладачів зберігається із гарантією довгострокової доступності та незмінності.

4.3 Безпека та захист даних у аналітичній платформі на Next.js 15

Питання захисту сучасних освітніх платформ особливо актуальні у контексті зберігання персональної інформації, навчальних результатів та медіаконтенту. Для реалізації надійного захисту у даній платформі застосовано багаторівневу архітектуру контролю доступу з використанням технологій Next.js 15, Better-Auth та Arcjet Security.

Arcjet middleware впроваджується у всі запити до сервіса для моніторингу та блокування типових загроз: ботів, XSS-ін’єкцій, спроб SQL-атаки. Це дозволяє централізовано фільтрувати небезпечний трафік до обробки бізнес-логіки, відповідний фрагмент коду зображений на рис.19:

```
async function authMiddleware(request: NextRequest) {
  const sessionCookie = getSessionCookie(request);
  if (!sessionCookie) {
    return NextResponse.redirect(new URL("/login", request.url));
  }
  return NextResponse.next();
}
// інтеграція лише на /admin маршрутах
```

Рис. 19 Фрагмент коду перенаправлення неавторизованих користувачів

Всі завантаження здійснюються через строго контрольований endpoint, на якому застосовано rate limiting, типову валідацію (zod schema) та виклик presigned URL із обмеженим часом життя, відповідний фрагмент коду зображений на рис.20:

```
const aj = arcjet.withRule(
  fixedWindow({ mode: "LIVE", window: "1m", max: 5 })
);

const presignedUrl = await getSignedUrl(S3, command, {
  expiresIn: 360, // 6 хвилин
});
```

Рис. 20 Фрагмент коду валідації presigned URL із обмеженим часом життя

З метою попередження масових атак, endpoint для завантаження файлів на S3 використовує жорсткі обмеження на частоту запитів (fixed window rate limiting), відповідний фрагмент коду зображений на рис.21:

```
const aj = arcjet.withRule(
  fixedWindow({
    mode: "LIVE",
    window: "1m",
    max: 5,
  })
);
```

Рис. 21 Фрагмент коду попередження масових атак.

Обмеження 5 операцій за хвилину забезпечує стійкість системи до brute force та масового спаму у файловому менеджері платформи.

Для забезпечення прозорості усіх дій користувачів, система автоматично веде журнал аудиту змін, спроб логіну, видалення чи завантаження файлів. Це дозволяє легко відслідковувати аномальні дії, своєчасно реагувати на інциденти, та дотримуватися вимог стандартів GDPR.

Всі реалізовані механізми разом формують цілісну концепцію захисту, яка відповідає сучасним стандартам та best practices OWASP. Інтегровані рішення надають можливість не просто реагувати на атаки, а й запобігати їм у режимі реального часу – що критично важливо для довіри користувача до аналітичної освітньої платформи.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-практичне завдання розробки та впровадження аналітичної платформи визначення досягнень здобувачів освіти на базі системи управління навчанням WazerCode. Отримані результати підтверджують доцільність застосування сучасних методів інтелектуального аналізу даних для підвищення якості освітнього процесу та забезпечення об'єктивного моніторингу академічних досягнень студентів.

Проведено комплексний аналіз предметної області, який засвідчив стрімке зростання ринку навчальних онлайн-платформ (з 35,2 млрд доларів США у 2023 році до прогнозованих 130,79 млрд доларів у 2032 році) та підтвердив критичну необхідність впровадження аналітичних інструментів для забезпечення прозорості, персоналізації та ефективності освітнього процесу в умовах дистанційного та гібридного навчання.

Здійснено дослідження методів створення аналітичних платформ для освітніх систем. Виконано порівняльний аналіз провідних фреймворків (React Admin, Tremor, Ant Design Pro), що дозволило обґрунтувати вибір технологічного стеку на основі Next.js 15, React 19 та TypeScript. Розроблено логічну модель даних за схемою «зірка» (star schema), яка забезпечує ефективну агрегацію та багатовимірний аналіз освітніх показників через реляційні таблиці CourseDim, TaskDim, StudentDim, TimeDim та фактову таблицю Performance.

Реалізовано комплекс методів інтелектуального аналізу даних для автоматизованої обробки освітніх досягнень. Впроваджено метод класифікації 1-Rule для сегментації студентів за рівнями успішності, наївний байєсівський класифікатор для прогнозування результатів, алгоритм Apriori для виявлення асоціативних правил між курсами та завданнями, а також кластерний аналіз за методом k-середніх для розподілу здобувачів на чотири групи за показниками середнього балу та активності.

Розроблено та інтегровано систему ключових показників ефективності (KPI) для багаторівневого оцінювання освітнього процесу, включно з метриками розподілу по кластерам, показниками прохідності курсів, індексами складності дисциплін та аналізом своєчасності виконання завдань. Система забезпечує науково обґрунтований зворотний зв'язок для підтримки управлінських рішень на рівні окремих курсів, груп студентів та закладу освіти.

Виконано практичну реалізацію платформи з використанням Next.js v15, серверного рендерингу, збірника Turbopack та бібліотек візуалізації даних (recharts, Nivo). Впроваджено багаторівневу архітектуру безпеки на основі Better-Auth та Arcjet Security з механізмами rate limiting та валідацією. Інтеграція з Amazon Web Services (S3, IAM) забезпечила масштабованість та надійність, а CI/CD пайплайни на базі Vercel автоматизували деплой та моніторинг.

Результати роботи мають практичне значення для закладів освіти, оскільки платформа забезпечує автоматизований моніторинг та аналіз освітніх досягнень у реальному часі. Наукова новизна полягає у комплексній інтеграції методів машинного навчання, аналітичних моделей та сучасних веб-технологій для створення гнучкої, масштабованої платформи освітньої аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пінкевич А. Модульна архітектура для розширеного React-застосунку [Електронний ресурс] – Режим доступу до ресурсу – https://journal.gen.tech/post/modules-for-react-app (дата звернення: 10.09.2025)
2. Проміси – Сучасний підручник з JavaScript [Електронний ресурс] – Режим доступу до ресурсу – https://uk.javascript.info/promise-basics (дата звернення: 12.11.2025)
3.
4. A Systematic Literature Review of Key Performance Indicators (KPIs) Implementation [Електронний ресурс] – Режим доступу до ресурсу – https://www.researchgate.net/publication/345941517_A_Systematic_Literature_Review_of_Key_Performance_Indicators_KPIs_Implementation%20 (дата звернення: 18.09.2025)
5. Перетворення даних (статистика) – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу – https://uk.wikipedia.org/wiki/Перетворення_даних_(статистика) (дата звернення: 12.09.2024)
6. Агрегування – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу – https://uk.wikipedia.org/wiki/Агрегування (дата звернення: 18.09.2025)
7. Brusilovsky P. Evaluating Adaptive E-Learning Systems [Електронний ресурс] Режим доступу до ресурсу – https://www.researchgate.net/publication/228772661_Layered_evaluation_of_adaptive_learning_systems (дата звернення: 14.09.2025)
8. Siemens G. Learning Analytics: Concepts and Techniques [Електронний ресурс] – Режим доступу до ресурсу – https://solaresearch.org/wp-content/uploads/2017/05/hla17.pdf(дата звернення: 27.08.2025)
9. Bichsel J. Analytics in Higher Education [Електронний ресурс] – Режим доступу до ресурсу –

https://library.educause.edu/~/media/files/library/2012/6/ers1207.pdf?la=en (дата звернення: 05.10.2025)

10. Kumar V. Big Data in Education [Електронний ресурс] – Режим доступу до ресурсу – https://shorturl.at/fEqXo (дата звернення: 31.09.2025)
11. Aljawarneh S. E-learning Platforms: Design and Implementation [Електронний ресурс] – Режим доступу до ресурсу – https://www.researchgate.net/publication/391083683_A_Study_of_Cloud_Computing_Based_E-Learning_Platforms_Design_and_Implementation (дата звернення: 21.09.2025)
12. Dringus L. P. Learning Analytics Considered Harmful [Електронний ресурс] – Режим доступу до ресурсу – https://files.eric.ed.gov/fulltext/EJ982677.pdf(дата звернення: 12.08.2025)
13. Domínguez A. Gamifying Learning Experiences [Електронний ресурс] – Режим доступу до ресурсу – https://www.researchgate.net/publication/256194365_Gamifying_Learning_Experiences_Practical_Implications_and_Outcomes (дата звернення: 03.09.2025)
14. Shah D. MOOCs Revisited [Електронний ресурс] – Режим доступу до ресурсу https://www.researchgate.net/publication/375333247_MOOCs_Revisited_Still_Transformative_or_Passing_Fad (дата звернення: 26.10.2025)
15. Next.js Documentation [Електронний ресурс] – Режим доступу до ресурсу – https://nextjs.org/docs (дата звернення: 19.09.2025)
16. React Documentation [Електронний ресурс] – Режим доступу до ресурсу – https://react.dev (дата звернення: 20.10.2025)
17. Prisma ORM Docs [Електронний ресурс] – Режим доступу до ресурсу – https://www.prisma.io/docs (дата звернення: 24.09.2025)
18. NeonDB Documentation [Електронний ресурс] – Режим доступу до ресурсу – https://neon.tech/docs (дата звернення: 11.10.2025)
19. Stripe API Documentation [Електронний ресурс] – Режим доступу до ресурсу – https://docs.stripe.com/api (дата звернення: 02.09.2025)

20. AWS Documentation [Електронний ресурс] – Режим доступу до ресурсу – https://docs.aws.amazon.com (дата звернення: 07.08.2025)
21. Vercel Deployment Docs [Електронний ресурс] – Режим доступу до ресурсу – https://vercel.com/docs (дата звернення: 23.09.2025)
22. Marmelab React Admin Docs [Електронний ресурс] – Режим доступу до ресурсу – https://marmelab.com/react-admin/ (дата звернення: 25.09.2025)
23. Tremor Data Visualization [Електронний ресурс] – Режим доступу до ресурсу – https://www.tremor.so/docs (дата звернення: 06.09.2025)
24. Ant Design Pro Docs [Електронний ресурс] – Режим доступу до ресурсу – https://pro.ant.design/docs/getting-started (дата звернення: 05.10.2025)
25. Wikipedia – Learning Management System [Електронний ресурс] – Режим доступу до ресурсу – https://en.wikipedia.org/wiki/Learning_management_system (дата звернення: 18.09.2025)
26. Wikipedia – Learning Analytics [Електронний ресурс] – Режим доступу до ресурсу – https://en.wikipedia.org/wiki/Learning_analytics (дата звернення: 27.08.2025)
27. Coursera: Офіційна сторінка [Електронний ресурс] – Режим доступу до ресурсу – https://www.coursera.org (дата звернення: 08.09.2025)
28. edX: Офіційна сторінка [Електронний ресурс] – Режим доступу до ресурсу – https://www.edx.org (дата звернення: 15.10.2025)
29. Moodle Docs [Електронний ресурс] – Режим доступу до ресурсу – https://docs.moodle.org (дата звернення: 17.09.2025)

ДОДАТКИ

ДОДАТОК А

ПРОГРАМНИЙ КОД СТВОРЕННЯ KPI:

```
import warnings
import sqlite3
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

# Налаштування середовища
warnings.filterwarnings("ignore")
pd.set_option("display.max_columns", None)
pd.set_option("display.width", None)
pd.set_option("display.max_colwidth", None)
sns.set_style("whitegrid")

fake = Faker("uk_UA")
random.seed(42)
np.random.seed(42)


# ========================== Вхідні дані ==========================

COURSES_DATA = {
    "Алгоритми": {"High": 96, "Satisfactory": 151, "Unsatisfactory": 231},
    "Штучний інтелект": {"High": 105, "Satisfactory": 158, "Unsatisfactory": 247},
    "Комп’ютерні мережі": {"High": 112, "Satisfactory": 147, "Unsatisfactory": 256},
    "Структури даних": {"High": 96, "Satisfactory": 145, "Unsatisfactory": 237},
    "Системи баз даних": {"High": 97, "Satisfactory": 152, "Unsatisfactory": 269},
    "Введення в програмування": {"High": 98, "Satisfactory": 167, "Unsatisfactory": 240},
    "Машинне навчання": {"High": 101, "Satisfactory": 178, "Unsatisfactory": 246},
    "Розробка мобільних додатків": {"High": 100, "Satisfactory": 155, "Unsatisfactory": 245},
    "Операційні системи": {"High": 85, "Satisfactory": 169, "Unsatisfactory": 241},
    "Веб-розробка": {"High": 103, "Satisfactory": 148, "Unsatisfactory": 232},
}

CLUSTER_CENTERS = {
    0: {"avg_score": 50.56, "task_count": 56.64, "course_count": 10.0},
    1: {"avg_score": 53.55, "task_count": 46.28, "course_count": 10.0},
    2: {"avg_score": 46.80, "task_count": 48.66, "course_count": 10.0},
    3: {"avg_score": 48.40, "task_count": 49.20, "course_count": 9.0},
}

STUDENT_NAMES = [
    "Богдан Кічак",
```

```
    "Василь Петренко",
    "Юлія Шевченко",
    "Артем Яценко",
    "Ірина Коваленко",
    "Андрій Ткачук",
    "Марія Дяченко",
    "Олександр Бондар",
    "Олена Іванова",
    "Владислав Соловей",
    "Катерина Романюк",
    "Анна Павлик",
]


# ====================== Ініціалізація БД ========================

conn = sqlite3.connect("student_analytics_complete.db")
cursor = conn.cursor()

for table in ["PerformanceFact", "AssignmentDim", "CourseDim", "StudentDim"]:
    cursor.execute(f"DROP TABLE IF EXISTS {table}")

cursor.execute(
    """
    CREATE TABLE StudentDim (
        id_student INTEGER PRIMARY KEY AUTOINCREMENT,
        student_name TEXT NOT NULL,
        student_group TEXT NOT NULL,
        enrollment_year INTEGER NOT NULL,
        avg_score REAL,
        task_count INTEGER,
        course_count INTEGER,
        cluster INTEGER,
        performance_class TEXT
    )
    """
)

cursor.execute(
    """
    CREATE TABLE CourseDim (
        id_course INTEGER PRIMARY KEY AUTOINCREMENT,
        course_code TEXT NOT NULL,
        course_name TEXT NOT NULL,
        credits INTEGER NOT NULL
    )
    """
)

cursor.execute(
    """
    CREATE TABLE AssignmentDim (
```

```
        id_assignment INTEGER PRIMARY KEY AUTOINCREMENT,
        id_course INTEGER NOT NULL,
        assignment_number INTEGER NOT NULL,
        assignment_name TEXT NOT NULL,
        max_score INTEGER DEFAULT 100,
        FOREIGN KEY (id_course) REFERENCES CourseDim(id_course)
    )
    """
)

cursor.execute(
    """
    CREATE TABLE PerformanceFact (
        id_performance INTEGER PRIMARY KEY AUTOINCREMENT,
        id_student INTEGER NOT NULL,
        id_assignment INTEGER NOT NULL,
        id_course INTEGER NOT NULL,
        score REAL NOT NULL,
        percentage REAL NOT NULL,
        performance_class TEXT NOT NULL,
        submission_date TEXT,
        days_to_submit INTEGER,
        FOREIGN KEY (id_student) REFERENCES StudentDim(id_student),
        FOREIGN KEY (id_assignment) REFERENCES AssignmentDim(id_assignment),
        FOREIGN KEY (id_course) REFERENCES CourseDim(id_course)
    )
    """
)

conn.commit()


# ====================== Заповнення вимірів ======================

for course_code, _ in COURSES_DATA.items():
    cursor.execute(
        """
        INSERT INTO CourseDim (course_code, course_name, credits)
        VALUES (?, ?, ?)
        """,
        (course_code, course_code, random.randint(4, 6)),
    )

conn.commit()

assignment_global_num = 1

for course_id in range(1, 11):
    num_assignments = 9 if course_id <= 7 else 8

    for local_num in range(1, num_assignments + 1):
        cursor.execute(
```

```
            """
            INSERT INTO AssignmentDim (id_course, assignment_number, assignment_name,
max_score)
            VALUES (?, ?, ?, ?)
            """,
            (course_id, local_num, f"Завдання {assignment_global_num}", 100),
        )
        assignment_global_num += 1

        if assignment_global_num > 87:
            break

    if assignment_global_num > 87:
        break

conn.commit()


# ====================== Генерація студентів ======================

groups = ["CS-401", "CS-402", "CS-403", "CS-404"]
cluster_distribution = {0: 20, 1: 35, 2: 30, 3: 15}

student_profiles = []
student_id_seq = 1

for cluster, count in cluster_distribution.items():
    center = CLUSTER_CENTERS[cluster]

    for i in range(count):
        if i < len(STUDENT_NAMES) and cluster in [0, 1]:
            name = STUDENT_NAMES[i % len(STUDENT_NAMES)]
        else:
            name = fake.name()

        group = random.choice(groups)
        avg_score = float(
            np.clip(np.random.normal(center["avg_score"], 3.5), 42.5, 62.5)
        )
        task_count = int(
            np.clip(np.random.normal(center["task_count"], 4), 42, 58)
        )
        course_count = int(center["course_count"])

        if avg_score >= 52:
            perf_class = "High"
        elif avg_score >= 48:
            perf_class = "Satisfactory"
        else:
            perf_class = "Unsatisfactory"

        cursor.execute(
```

```
            """
            INSERT INTO StudentDim (
                student_name,
                student_group,
                enrollment_year,
                avg_score,
                task_count,
                course_count,
                cluster,
                performance_class
            )
            VALUES (?, ?, ?, ?, ?, ?, ?, ?)
            """,
            (
                str(name),
                str(group),
                2021,
                float(avg_score),
                int(task_count),
                int(course_count),
                int(cluster),
                str(perf_class),
            ),
        )

        student_profiles.append(
            {
                "id": student_id_seq,
                "name": name,
                "avg_score": avg_score,
                "task_count": task_count,
                "course_count": course_count,
                "cluster": cluster,
                "perf_class": perf_class,
            }
        )
        student_id_seq += 1

conn.commit()


# ================== Фактичні показники (Fact) ==================

start_date = datetime(2024, 9, 1)

for student_profile in student_profiles:
    student_id = student_profile["id"]
    target_avg = student_profile["avg_score"]
    target_tasks = student_profile["task_count"]
    perf_class = student_profile["perf_class"]

    all_assignments = list(range(1, 88))
```

```
selected_assignments = random.sample(all_assignments, target_tasks)

student_scores = []

for assignment_id in selected_assignments:
    cursor.execute(
        "SELECT id_course FROM AssignmentDim WHERE id_assignment = ?",
        (assignment_id,),
    )
    course_id = cursor.fetchone()[0]

    rand_val = random.random()

    if perf_class == "High":
        if rand_val < 0.35:
            score_class = "High"
        elif rand_val < 0.70:
            score_class = "Satisfactory"
        else:
            score_class = "Unsatisfactory"
    elif perf_class == "Satisfactory":
        if rand_val < 0.20:
            score_class = "High"
        elif rand_val < 0.65:
            score_class = "Satisfactory"
        else:
            score_class = "Unsatisfactory"
    else:
        if rand_val < 0.10:
            score_class = "High"
        elif rand_val < 0.35:
            score_class = "Satisfactory"
        else:
            score_class = "Unsatisfactory"

    if score_class == "High":
        percentage = np.random.uniform(70, 95)
        perf_ukr = "Високий"
    elif score_class == "Satisfactory":
        percentage = np.random.uniform(50, 69)
        perf_ukr = "Задовільний"
    else:
        percentage = np.random.uniform(25, 49)
        perf_ukr = "Незадовільний"

    score = percentage
    student_scores.append(score)

    days_offset = random.randint(0, 120)
    submission_date = start_date + timedelta(days=days_offset)

    cursor.execute(
```

```
            """
            INSERT INTO PerformanceFact (
                id_student,
                id_assignment,
                id_course,
                score,
                percentage,
                performance_class,
                submission_date,
                days_to_submit
            )
            VALUES (?, ?, ?, ?, ?, ?, ?, ?)
            """,
            (
                int(student_id),
                int(assignment_id),
                int(course_id),
                float(score),
                float(percentage),
                perf_ukr,
                submission_date.strftime("%Y-%m-%d"),
                int(days_offset),
            ),
        )

    actual_avg = np.mean(student_scores) if student_scores else target_avg
    cursor.execute(
        "UPDATE StudentDim SET avg_score = ? WHERE id_student = ?",
        (float(actual_avg), int(student_id)),
    )

conn.commit()


# ====================== Аналітичні запити (KPI) ==================

df_clustered = pd.read_sql_query(
    """
    SELECT
        id_student,
        CAST(avg_score AS INTEGER) AS avg_score,
        task_count,
        course_count,
        cluster AS Кластер
    FROM StudentDim
    ORDER BY id_student
    """,
    conn,
)

df_centers = pd.read_sql_query(
    """
```

```
    SELECT
        cluster AS Кластер,
        ROUND(AVG(avg_score), 1) AS Середній_бал,
        ROUND(AVG(CAST(task_count AS REAL)), 1) AS Завдань,
        ROUND(AVG(CAST(course_count AS REAL)), 1) AS Курсів
    FROM StudentDim
    GROUP BY cluster
    ORDER BY cluster
    """,
    conn,
)

df_success = pd.read_sql_query(
    """
    SELECT
        c.course_code AS Курс,
        COUNT(DISTINCT pf.id_student) AS Студентів,
        SUM(CASE WHEN pf.percentage >= 50 THEN 1 ELSE 0 END) AS Склали,
        ROUND(
            SUM(CASE WHEN pf.percentage >= 50 THEN 1 ELSE 0 END) * 100.0 /
            COUNT(*),
            2
        ) AS Прохідність
    FROM PerformanceFact pf
    JOIN CourseDim c ON pf.id_course = c.id_course
    GROUP BY c.course_code
    ORDER BY Прохідність DESC
    """,
    conn,
)

df_engagement = pd.read_sql_query(
    """
    SELECT
        student_group AS Група,
        ROUND(AVG(task_count * 100.0 / 87), 2) AS Залученість,
        MIN(task_count) AS Мін_завдання,
        MAX(task_count) AS Макс_завдання,
        COUNT(*) AS Студентів
    FROM StudentDim
    GROUP BY student_group
    ORDER BY Залученість DESC
    """,
    conn,
)

df_group_perf = pd.read_sql_query(
    """
    SELECT
        student_group AS Група,
        SUM(CASE WHEN performance_class = 'Високий' THEN 1 ELSE 0 END) AS Високий,
```

```
    SUM(CASE WHEN performance_class = 'Задовільний' THEN 1 ELSE 0 END) AS Задовільний,
    SUM(CASE WHEN performance_class = 'Незадовільний' THEN 1 ELSE 0 END) AS Незадовільний,
    ROUND(AVG(avg_score), 2) AS Середній_бал,
    COUNT(*) AS Студентів
  FROM StudentDim
  GROUP BY student_group
  ORDER BY Середній_бал DESC
  """,
  conn,
)

df_difficulty = pd.read_sql_query(
  """
  SELECT
    c.course_code AS Курс,
    ROUND(AVG(pf.percentage), 2) AS Середній_бал_курсу,
    ROUND((SELECT AVG(percentage) FROM PerformanceFact), 2) AS Загальний_середній,
    ROUND(
      AVG(pf.percentage) / (SELECT AVG(percentage) FROM PerformanceFact),
      3
    ) AS Індекс_складності,
    CASE
      WHEN AVG(pf.percentage) / (SELECT AVG(percentage) FROM PerformanceFact) < 0.85
        THEN 'Складний'
      WHEN AVG(pf.percentage) / (SELECT AVG(percentage) FROM PerformanceFact) > 1.15
        THEN 'Легкий'
      ELSE 'Помірний'
    END AS Рівень_складності
  FROM PerformanceFact pf
  JOIN CourseDim c ON pf.id_course = c.id_course
  GROUP BY c.course_code
  ORDER BY Індекс_складності ASC
  """,
  conn,
)

df_timing = pd.read_sql_query(
  """
  SELECT
    CASE
      WHEN days_to_submit < 30 THEN 'Рано (0-30 днів)'
      WHEN days_to_submit < 60 THEN 'Вчасно (31-60 днів)'
      WHEN days_to_submit < 90 THEN 'Пізно (61-90 днів)'
      ELSE 'Дуже пізно (90+ днів)'
    END AS Категорія_здачі,
    COUNT(*) AS Кількість,
    ROUND(AVG(percentage), 2) AS Середній_бал,
    ROUND(
      COUNT(*) * 100.0 / (SELECT COUNT(*) FROM PerformanceFact),
      2
```

```
        ) AS Відсоток_загальної_кількості
    FROM PerformanceFact
    GROUP BY Категорія_здачі
    ORDER BY CASE Категорія_здачі
        WHEN 'Рано (0-30 днів)' THEN 1
        WHEN 'Вчасно (31-60 днів)' THEN 2
        WHEN 'Пізно (61-90 днів)' THEN 3
        ELSE 4
    END
    """,
    conn,
)


# ========================= Візуалізація ==========================

colors = ["#4169E1", "#87CEEB", "#FF6B6B", "#8B0000"]

# 1. Кластеризація студентів
fig1, ax1 = plt.subplots(figsize=(8, 6))

for cluster in range(4):
    cluster_data = df_clustered[df_clustered["Кластер"] == cluster]
    ax1.scatter(
        cluster_data["avg_score"],
        cluster_data["task_count"],
        c=colors[cluster],
        label=f"Кластер {cluster}",
        s=80,
        alpha=0.6,
        edgecolors="black",
        linewidth=1,
    )

centers_data = df_centers[["Середній_бал", "Завдань"]].values

ax1.scatter(
    centers_data[:, 0],
    centers_data[:, 1],
    c="yellow",
    marker="P",
    s=300,
    edgecolors="black",
    linewidths=2,
    label="Центри",
    zorder=5,
)

ax1.set_xlabel("Середній бал", fontweight="bold")
ax1.set_ylabel("Кількість завдань", fontweight="bold")
ax1.set_title("Кластеризація студентів", fontweight="bold", fontsize=14)
ax1.legend()
```

```
ax1.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# 2. Розподіл студентів по кластерах
fig2, ax2 = plt.subplots(figsize=(6, 6))

cluster_counts = df_clustered["Кластер"].value_counts().sort_index()

ax2.pie(
    cluster_counts.values,
    labels=[f"К{str(i)}" for i in cluster_counts.index],
    autopct="%1.1f%%",
    colors=colors,
    startangle=90,
)

ax2.set_title("Розподіл по кластерах", fontweight="bold", fontsize=12)

plt.tight_layout()
plt.show()

# 3. Прохідність курсів
fig3, ax3 = plt.subplots(figsize=(8, 6))

ax3.barh(df_success["Курс"], df_success["Прохідність"], color="#4CAF50")
ax3.axvline(x=50, color="red", linestyle="--", linewidth=2, label="Поріг 50%")

ax3.set_xlabel("Прохідність (%)", fontweight="bold")
ax3.set_ylabel("Курс", fontweight="bold")
ax3.set_title("Успішність курсів", fontweight="bold", fontsize=14)
ax3.legend()
ax3.grid(True, alpha=0.3, axis="x")

plt.tight_layout()
plt.show()

# 4. Успішність по академічних групах
fig4, ax4 = plt.subplots(figsize=(8, 6))

df_group_plot = df_group_perf.set_index("Група")[
    ["Високий", "Задовільний", "Незадовільний"]
]

df_group_plot.plot(
    kind="bar",
    stacked=True,
    ax=ax4,
    color=["#90EE90", "#FFD700", "#FF6B6B"],
)
```

```
ax4.set_xlabel("Група", fontweight="bold")
ax4.set_ylabel("Студенти", fontweight="bold")
ax4.set_title("Успішність по групах", fontweight="bold", fontsize=12)
ax4.legend(title="Клас успішності")
ax4.tick_params(axis="x", rotation=45)

plt.tight_layout()
plt.show()

# 5. Залученість студентів
fig5, ax5 = plt.subplots(figsize=(8, 6))

ax5.bar(df_engagement["Група"], df_engagement["Залученість"], color="#2196F3")
ax5.axhline(y=60, color="orange", linestyle="--", linewidth=2, label="Ціль 60%")

ax5.set_xlabel("Група", fontweight="bold")
ax5.set_ylabel("Залученість (%)", fontweight="bold")
ax5.set_title("Залученість студентів", fontweight="bold", fontsize=12)
ax5.legend()
ax5.grid(True, alpha=0.3, axis="y")

plt.tight_layout()
plt.show()

# 6. Індекс складності курсу
fig6, ax6 = plt.subplots(figsize=(8, 6))

difficulty_colors = {
   "Легкий": "#90EE90",
   "Помірний": "#FFD700",
   "Складний": "#FF6B6B",
}

colors_list = [
   difficulty_colors[level] for level in df_difficulty["Рівень_складності"]
]

ax6.barh(df_difficulty["Курс"], df_difficulty["Індекс_складності"], color=colors_list)
ax6.axvline(x=1.0, color="black", linestyle="--", linewidth=2, label="Середнє")

ax6.set_xlabel("Індекс складності", fontweight="bold")
ax6.set_ylabel("Курс", fontweight="bold")
ax6.set_title("Складність курсу", fontweight="bold", fontsize=12)
ax6.legend()
ax6.grid(True, alpha=0.3, axis="x")

plt.tight_layout()
plt.show()

# 7. Категорії часу здачі
fig7, ax7 = plt.subplots(figsize=(8, 6))
```

```
submission_labels = df_timing["Категорія_здачі"].tolist()
submission_values = df_timing["Кількість"].tolist()
colors_timing = ["#4CAF50", "#FFC107", "#FF9800", "#F44336"]

ax7.bar(range(len(submission_labels)), submission_values, color=colors_timing)
ax7.set_xticks(range(len(submission_labels)))
ax7.set_xticklabels(submission_labels, rotation=15, ha="right")

ax7.set_ylabel("Кількість здач", fontweight="bold")
ax7.set_title("Категорії часу здачі", fontweight="bold", fontsize=14)
ax7.grid(True, alpha=0.3, axis="y")

plt.tight_layout()
plt.show()

# 8. Вплив часу здачі на оцінку
fig8, ax8 = plt.subplots(figsize=(8, 6))

ax8.plot(
    range(len(submission_labels)),
    df_timing["Середній_бал"].tolist(),
    marker="o",
    linewidth=3,
    markersize=10,
    color="#2196F3",
)

ax8.set_xticks(range(len(submission_labels)))
ax8.set_xticklabels(submission_labels, rotation=15, ha="right")
ax8.set_ylabel("Середній бал", fontweight="bold")
ax8.set_title("Вплив часу здачі на оцінку", fontweight="bold", fontsize=14)
ax8.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

conn.close()
print("Усі графіки й легенди подано українською мовою.")
```

ДОДАТОК Б

ПРОГРАМНИЙ КОД ДЛЯ СТВОРЕННЯ ДАШБОРДУ:

```
import React from "react";
import {
  Card,
  CardContent,
  CardDescription,
  CardHeader,
  CardTitle,
} from "../../../components/ui/card";
import {
  adminGetCourseCompletionStats,
  adminGetUserProgressStats,
  adminGetMonthlyProgressStats,
  adminGetDashboardStats,
} from "../../data/admin/admin-get-course-completion-stats";
import {
  IconBook,
  IconUsers,
  IconTrendingUp,
  IconCheck,
} from "@tabler/icons-react";
import { CourseCompletionChart } from "./_components/CourseCompletionChart";
import { MonthlyProgressChart } from "./_components/MonthlyProgressChart";
import { EnrollmentChart } from "./_components/EnrollmentChart";

/**
 * Головна сторінка аналітики — відображає загальні показники, графіки та топи.
 */
export default async function AnalyticsPage() {
  // Отримання статистики з різних джерел
  const [
    courseStats,
    userStats,
    monthlyStats,
    dashboardStats,
  ] = await Promise.all([
    adminGetCourseCompletionStats(),
    adminGetUserProgressStats(),
    adminGetMonthlyProgressStats(),
    adminGetDashboardStats(),
  ]);

  // Визначення топ-курсів (за відсотком завершення)
  const topCourses = courseStats
    .filter((course) => course.totalEnrollments > 0)
    .sort((a, b) => b.completionRate - a.completionRate)
    .slice(0, 5);

  // Визначення топ-студентів (за прогресом)
```

```
const topUsers = userStats
  .filter((user) => user.totalLessons > 0)
  .sort((a, b) => b.progressPercentage - a.progressPercentage)
  .slice(0, 5);

return (
  <div className="space-y-6">
    {/* Заголовок */}
    <div className="flex flex-col gap-2">
      <h1 className="text-3xl font-bold">Аналітика</h1>
      <p className="text-muted-foreground">
        Повна аналітика щодо завершення курсів і залученості користувачів
      </p>
    </div>

    {/* Основні показники */}
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4">
      <Card>
        <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
          <div>
            <CardDescription>Усього курсів</CardDescription>
            <CardTitle className="text-2xl font-semibold">
              {dashboardStats.totalCourses}
            </CardTitle>
          </div>
          <IconBook className="size-6 text-muted-foreground" />
        </CardHeader>
        <CardContent>
          <p className="text-sm text-muted-foreground">
            Доступні курси на платформі
          </p>
        </CardContent>
      </Card>

      <Card>
        <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
          <div>
            <CardDescription>Активні студенти</CardDescription>
            <CardTitle className="text-2xl font-semibold">
              {dashboardStats.totalCustomers}
            </CardTitle>
          </div>
          <IconUsers className="size-6 text-muted-foreground" />
        </CardHeader>
        <CardContent>
          <p className="text-sm text-muted-foreground">
            Студенти, що записані на курси
          </p>
        </CardContent>
      </Card>

      <Card>
```

```
    <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
     <div>
      <CardDescription>Середній відсоток завершення</CardDescription>
      <CardTitle className="text-2xl font-semibold">
       {courseStats.length > 0
        ? Math.round(
           courseStats.reduce(
            (acc, course) => acc + course.completionRate,
            0
           ) / courseStats.length
          )
        : 0}
       %
      </CardTitle>
     </div>
     <IconTrendingUp className="size-6 text-muted-foreground" />
    </CardHeader>
    <CardContent>
     <p className="text-sm text-muted-foreground">
      Середній відсоток завершення курсів
     </p>
    </CardContent>
   </Card>

   <Card>
    <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">
     <div>
      <CardDescription>Усього уроків</CardDescription>
      <CardTitle className="text-2xl font-semibold">
       {dashboardStats.totalLessons}
      </CardTitle>
     </div>
     <IconCheck className="size-6 text-muted-foreground" />
    </CardHeader>
    <CardContent>
     <p className="text-sm text-muted-foreground">
      Доступний навчальний контент
     </p>
    </CardContent>
   </Card>
  </div>

  {/* Графіки завершення курсів та щомісячної активності */}
  <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
   <Card>
    <CardHeader>
     <CardTitle>Відсоток завершення курсів</CardTitle>
     <CardDescription>
      Показники завершення за курсами (лише з записами)
     </CardDescription>
    </CardHeader>
    <CardContent>
```

```
          <CourseCompletionChart data={courseStats} />
        </CardContent>
      </Card>

      <Card>
        <CardHeader>
          <CardTitle>Щомісячні завершення уроків</CardTitle>
          <CardDescription>
            Динаміка завершення за останні 6 місяців
          </CardDescription>
        </CardHeader>
        <CardContent>
          <MonthlyProgressChart data={monthlyStats} />
        </CardContent>
      </Card>
    </div>

    {/* Топи курсів та студентів */}
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
      <Card>
        <CardHeader>
          <CardTitle>Найрезультативніші курси</CardTitle>
          <CardDescription>
            Курси з найвищими відсотками завершення
          </CardDescription>
        </CardHeader>
        <CardContent>
          <div className="space-y-4">
            {topCourses.map((course, index) => (
              <div
                key={course.courseId}
                className="flex items-center justify-between p-3 border rounded-lg"
              >
                <div className="flex items-center space-x-3">
                  <div className="flex items-center justify-center w-8 h-8 bg-primary/10 rounded-full
text-sm font-medium">
                    {index + 1}
                  </div>
                  <div>
                    <p className="font-medium">{course.courseTitle}</p>
                    <p className="text-sm text-muted-foreground">
                      {course.totalEnrollments} студентів • {course.totalLessons} уроків
                    </p>
                  </div>
                </div>
                <div className="text-right">
                  <p className="font-semibold">{course.completionRate}%</p>
                  <p className="text-xs text-muted-foreground">завершення</p>
                </div>
              </div>
            ))}
          </div>
```

```
        </CardContent>
      </Card>

      <Card>
        <CardHeader>
          <CardTitle>Найкращі студенти</CardTitle>
          <CardDescription>Студенти з найвищим прогресом</CardDescription>
        </CardHeader>
        <CardContent>
          <div className="space-y-4">
            {topUsers.map((user, index) => (
              <div
                key={user.userId}
                className="flex items-center justify-between p-3 border rounded-lg"
              >
                <div className="flex items-center space-x-3">
                  <div className="flex items-center justify-center w-8 h-8 bg-primary/10 rounded-full
text-sm font-medium">
                    {index + 1}
                  </div>
                  <div>
                    <p className="font-medium">{user.userName}</p>
                    <p className="text-sm text-muted-foreground">
                      {user.enrolledCourses} курсів • {user.completedLessons}/{user.totalLessons}
уроків
                    </p>
                  </div>
                </div>
                <div className="text-right">
                  <p className="font-semibold">{user.progressPercentage}%</p>
                  <p className="text-xs text-muted-foreground">прогрес</p>
                </div>
              </div>
            ))}
          </div>
        </CardContent>
      </Card>
    </div>

    {/* Розподіл записів на курси */}
    <Card>
      <CardHeader>
        <CardTitle>Розподіл записів на курси</CardTitle>
        <CardDescription>Кількість записів на курс</CardDescription>
      </CardHeader>
      <CardContent>
        <EnrollmentChart data={courseStats} />
      </CardContent>
    </Card>
  </div>
 );
}
```

ДОДАТОК В

ЗОВНІШНІЙ ВИГЛЯД ДАШБОРДУ