**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система аналітики успішності студентів. Спец-частина: розробка Frontend-частини системи»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Олексій КРИВОБОК
*(підпис)*

Виконав: здобувач вищої освіти групи ПД-44

Олексій КРИВОБОК

Керівник: Ігор ЦАПРО
*викладач*

Рецензент: _____

**Київ 2026**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кривобоку Олексію Анатолійовичу

1. Тема кваліфікаційної роботи: «Система аналітики успішності студентів. Спец-частина: розробка Frontend-частини системи»
керівник кваліфікаційної роботи викладач кафедри ІПЗ Ігор ЦАПРО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.
2. Строк подання кваліфікаційної роботи «29» травня 2026 року.
3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про архітектуру клієнт-серверних застосунків, принципи побудови користувацьких веб-інтерфейсів, документація до фреймворків React, React Router, Recharts, Vite, підходи до реалізації безпеки, авторизації та розмежування прав доступу.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі систем, пов’язаних з аналітикою успішності студентів; огляд аналогів, визначення вимог до клієнтської частини системи.
2. Огляд програмних засобів реалізації клієнтської частини системи.
3. Проєктування архітектури клієнтської частини системи аналітики успішності студентів.
4. Реалізація та тестування клієнтської частини системи.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма прецедентів.
5. Діаграма послідовності завантаження головної сторінки.
6. Діаграма послідовності додавання студента.
7. Мапа системи «SDF».
8. Екранні форми.
9. Апробація результатів дослідження.

6. Дата видачі завдання «23» лютого 2026 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та аналіз науково-технічної літератури | 23.02-28.02.2026 | |
| 2 | Аналіз предметної галузі | 01.03-07.03.2026 | |
| 3 | Аналіз існуючих аналогів та визначення вимог | 01.03-07.03.2026 | |
| 4 | Огляд програмних засобів реалізації системи | 09.03-13.03.2026 | |
| 5 | Проєктування архітектури системи | 16.03-20.03.2026 | |
| 6 | Програмна реалізація клієнтської частини системи | 23.03-17.04.2026 | |
| 7 | Тестування системи | 20.04.2026 | |
| 8 | Оформлення роботи: вступ, висновки, реферат | 01.05-06.05.2026 | |
| 9 | Розробка демонстраційних матеріалів | 07.05-08.05.2026 | |
| 10 | Попередній захист роботи | 11.05-22.05.2026 | |

Здобувач вищої освіти _____ Олексій КРИВОБОК

*(підпис)*

Керівник кваліфікаційної роботи _____ Ігор ЦАПРО

*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 57 стор., 1 табл., 26 рис., 25 джерел.

*Мета роботи* – удосконалення процесу управління аналізом успішності у закладах вищої освіти за рахунок створення та впровадження комплексного веб-застосунку для безпечного збереження, агрегації та візуалізації освітніх даних.

*Об'єкт дослідження* – процеси аналізу успішності та інтерактивного представлення аналітичних даних у веб-середовищі.

*Предмет дослідження* – методи, засоби та технології розробки систем аналітики освітніх даних у закладах вищої освіти.

*Короткий зміст роботи*: В роботі проаналізовано особливості систем контролю та аналітики успішності студентів. Проведено аналіз засобів, що виконують такі задачі: Canvas LMS, Moodle, Blackboard Predict. Розроблено архітектуру клієнтської частини системи, алгоритм її роботи, та програмно реалізовані поставлені функціональні й нефункціональні вимоги та ключові можливості, зокрема: авторизація адміністратора, головний інформаційний дашборд, список студентів з фільтрами, сторінка динаміки оцінок, сторінка порівняння навчальних груп; визначення студентів у групі ризику, тобто з занадто низьким середнім балом. В роботі використано мову розмітки веб-сторінок HTML, мову стилю веб-сторінок CSS та мову програмування JavaScript у фреймворках для створення веб-застосунків React, React Router та інструменті розгортання Vite.

Сферою використання застосунку є візуалізація та агрегація освітніх даних студентів вищих навчальних закладів.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, ПРОГРАМНА ІНЖЕНЕРІЯ, FRONT-END, REACT, JAVASCRIPT, АНАЛІЗ УСПІШНОСТІ СТУДЕНТІВ, ОБРОБКА ОСВІТНІХ ДАНИХ, АНАЛІТИКА ОСВІТНІХ ДАНИХ, СИСТЕМА МОНІТОРИНГУ СТУДЕНТІВ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, UX

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

API - Application Programming Interface (інтерфейс прикладного програмування)

REST - Representational State Transfer (передача репрезентативного стану)

HTTP - Hypertext Transfer Protocol (протокол передачі гіпертексту)

JSON - JavaScript Object Notation (формат обміну даними)

CRUD - Create, Read, Update, Delete (базові операції з даними)

LMS - Learning Management System (система управління навчанням)

IDE - Integrated Development Environment (інтегроване середовище розробки)

ЗВО - Заклад вищої освіти

GPA - Grade Point Average (середній бал)

UI - User Interface (користувацький інтерфейс)

UX - User Experience (користувацький досвід)

# ВСТУП

**Актуальність теми.** У сучасних умовах глобальної цифровізації та стрімкого переходу закладів вищої освіти до змішаних і дистанційних форм навчання, традиційні методи моніторингу успішності студентів демонструють свою недостатню ефективність. Величезні масиви даних, що генеруються в освітньому процесі, вимагають якісно нових інструментів для їхнього збору, систематизації та інтелектуальної обробки. Для України це питання набуває особливої значущості у контексті інтеграції до європейського освітнього простору та необхідності забезпечення прозорості та об’єктивності оцінювання знань. Шляхом порівняння з існуючими рішеннями встановлено, що більшість систем обмежуються лише фіксацією оцінок, не пропонуючи глибокої аналітики та прогнозних моделей. Розробка системи аналітики успішності на базі сучасного технологічного стеку, що включає нереляційні бази даних та легкий у використанні інтерфейс, є науково-прикладним завданням, спрямованим на підвищення якості управління освітнім процесом.

**Метою роботи** є удосконалення процесу управління аналізом успішності у закладах вищої освіти за рахунок створення та впровадження комплексного веб-застосунку для безпечного збереження, агрегації та візуалізації освітніх даних.

**Об’єктом дослідження** є процеси аналізу успішності та інтерактивного представлення аналітичних даних у веб-середовищі.

**Предметом дослідження** є методи, засоби та технології розробки систем аналітики освітніх даних у закладах вищої освіти.

**Методи дослідження.** Для досягнення поставленої мети було використано комплекс методів, що включає: системний аналіз для вивчення стану освітніх процесів; статистичні методи обробки та агрегації даних для розрахунку показників успішності; методи проєктування користувацьких інтерфейсів; методи розробки веб-застосунків за допомогою компонентного підходу та фреймворку React.

**Наукова новизна** полягає в удосконаленні методики моніторингу успішності шляхом поєднання нереляційного підходу до зберігання освітніх даних із використанням зручного веб-інтерфейсу з можливістю легкої навігації. Запропоновано архітектурне рішення, що забезпечує миттєву агрегацію великих масивів оцінок без використання складних реляційних запитів, що підвищує швидкодію системи у кілька разів.

**Практична значущість** роботи полягає у створенні функціонального програмного інструменту, який може бути впроваджений у роботу кафедр та деканатів для автоматизації генерації звітів, виявлення груп ризику серед студентів та оптимізації зворотного зв’язку.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Роль цифрових платформ у системі вищої освіти

Сучасний етап розвитку глобального суспільства характеризується тотальною цифровізацією всіх сфер людської життєдіяльності, серед яких система вищої освіти посідає одне з ключових місць. Трансформація освітнього процесу під впливом інформаційних технологій зумовила перехід від традиційних моделей навчання до інноваційних парадигм, де центральну роль відіграють цифрові платформи. Ці інструменти перестали бути просто допоміжними засобами збереження інформації, перетворившись на складні багатофункціональні середовища, що визначають ефективність взаємодії між усіма учасниками академічного процесу [1].

Для глибокого розуміння даної проблематики необхідно дати фундаментальне визначення базового терміну. Цифрова освітня платформа - це інтегрована інформаційна система, яка забезпечує комплексну підтримку освітнього процесу шляхом надання доступу до навчального контенту, засобів комунікації, інструментів контролю знань та систем адміністрування в єдиному цифровому інтерфейсі. Вона виступає як фундамент для побудови персоналізованих траєкторій навчання, дозволяючи студентам та викладачам працювати в асинхронному режимі без втрати якості методичного супроводу.

Важливим аспектом функціонування таких платформ є їхня здатність консолідувати величезні масиви даних, що генеруються в процесі навчання. У науковій літературі часто використовується поняття навчальної екосистеми, під якою розуміють динамічну сукупність цифрових ресурсів, сервісів та користувачів, що взаємодіють між собою для досягнення освітніх цілей у віртуальному просторі.

Роль цифрових платформ у закладах вищої освіти виходить далеко за межі звичайної дистрибуції навчальних матеріалів у форматі PDF чи відеолекцій. Вони

забезпечують автоматизацію рутинних адміністративних процесів, таких як формування розкладів, ведення електронних журналів та реєстрація на вибіркові дисципліни [2].

Окрему увагу слід приділити концепції LMS (Learning Management System), яка є найбільш розповсюдженою формою реалізації цифрових платформ (рис. 1.1). Система управління навчанням (LMS) - це програмний додаток або веб-технологія, що використовується для планування, реалізації та оцінювання певного процесу навчання. Саме в межах таких систем відбувається основний збір даних про успішність студентів, що стає підґрунтям для подальшого аналізу та прийняття управлінських рішень [3].

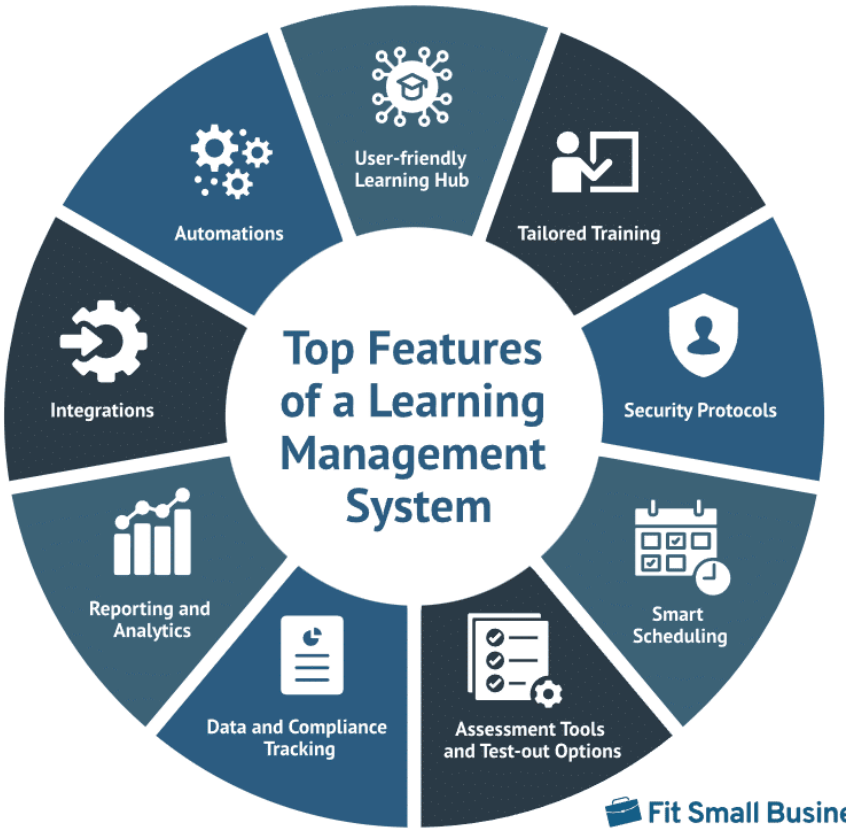



Рис. 1.1 Система Learning Management System

Впровадження цифрових платформ змінює методику моніторингу успішності, роблячи її прозорою та об’єктивною. Замість дискретного контролю під час сесій, викладачі отримують можливість здійснювати безперервний моніторинг активності здобувачів протягом усього семестру. Кожна взаємодія

студента з платформою залишає цифровий слід, який може бути використаний для ідентифікації проблемних зон у засвоєнні матеріалу на ранніх етапах.

Цифровізація освіти сприяє розвитку культури самостійної роботи, де платформа виступає навігатором у складному інформаційному полі. Студент перестає бути пасивним отримувачем інформації, перетворюючись на активного суб'єкта, який самостійно планує свій час та контролює власний прогрес через систему зворотного зв'язку. Це формує навички самоменеджменту, які є критично важливими для сучасної IT-індустрії та фрілансу [4].

Інтеграція аналітичних модулів у структури освітніх платформ дозволяє реалізувати предиктивний підхід до оцінювання результатів навчання. Використання методів інтелектуального аналізу даних дає змогу прогнозувати результати підсумкового контролю на основі поточної успішності, відвідуваності та залученості студента. Такий підхід трансформує роль цифрової платформи з інструменту обліку в інструмент стратегічного управління якістю освіти.

Соціальний аспект цифрових платформ проявляється у створенні горизонтальних зв'язків між студентами через форуми, чати та системи спільного редагування документів. Комунікаційний простір платформи стає місцем обміну досвідом та спільного вирішення складних навчальних кейсів [5].

Важливим чинником ефективності цифрових платформ є їхня масштабованість та гнучкість щодо інтеграції з іншими програмними продуктами. Використання сучасних архітектурних підходів, таких як мікросервіси чи хмарні обчислення, дозволяє університетам розширювати функціонал своїх систем без зупинки навчального процесу.

Інформаційна безпека та захист персональних даних здобувачів освіти є пріоритетним напрямком при проєктуванні сучасних цифрових платформ. Оскільки в системі накопичується конфіденційна інформація про академічні досягнення та особисті дані користувачів, розробники повинні приділяти особливу увагу механізмам автентифікації та шифрування. Надійність платформи впливає на рівень довіри до неї з боку академічної спільноти [6].

Використання відкритих стандартів та API в освітніх платформах дозволяє створювати цілісні інформаційні ланцюги, де дані про успішність автоматично передаються до загальнодержавних реєстрів чи систем внутрішнього аудиту. Це забезпечує високий рівень академічної доброчесності, оскільки будь-які маніпуляції з оцінками стають технічно складними та легко виявляються під час верифікації. Цифрова платформа виступає гарантом якості освітніх послуг.

Економічна ефективність впровадження цифрових платформ полягає в оптимізації витрат на паперовий документообіг та утриманні великих управлінських штатів. Централізоване зберігання даних у нереляційних сховищах, таких як MongoDB, забезпечує високу швидкість обробки запитів та гнучкість структури документів [7]. Принцип роботи MongoDB наведено на рисунку 1.2.

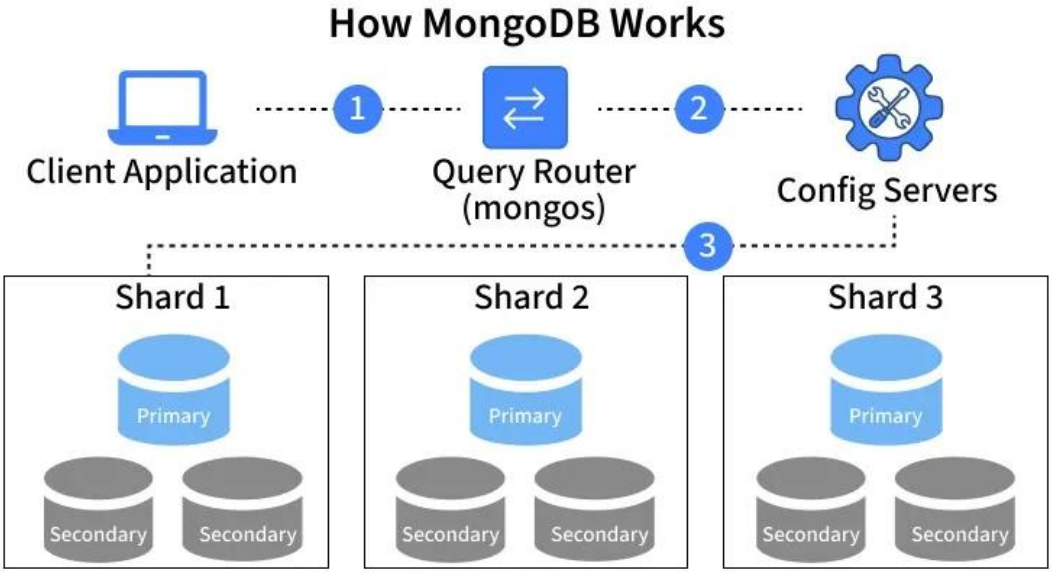



Рис. 1.2 Принцип роботи MongoDB

У контексті підготовки майбутніх фахівців, цифрова платформа сама по собі стає об’єктом вивчення та середовищем для відпрацювання практичних навичок розробки. Студенти технічних спеціальностей, взаємодіючи з якісно спроєктованим інтерфейсом на базі React, отримують уявлення про сучасні стандарти UI/UX дизайну [8].

Підсумовуючи, можна стверджувати, що роль цифрових платформ у системі вищої освіти є фундаментальною та системоутворюючою. Вони не лише

автоматизують існуючі процеси, а й створюють нові можливості для аналізу, прогнозування та покращення результатів навчання.

### 1.2 Методологія аналізу академічної успішності

Методологія аналізу академічної успішності є складною теоретичною конструкцією, що об’єднує педагогічні, статистичні та інформаційні підходи до оцінювання результатів навчання. Вона базується на систематичному вивченні динаміки засвоєння знань, формуванні компетенцій та досягненні програмних результатів освіти протягом визначеного періоду. Основним завданням цієї методології є не просто констатація фактів отримання оцінок, а виявлення прихованих закономірностей, які впливають на якість освітнього процесу в межах вищого навчального закладу [9].

Центральним поняттям даного підрозділу є академічна успішність, яку слід визначити як ступінь відповідності реальних результатів навчальної діяльності студента вимогам освітньої програми та встановленим стандартам оцінювання.

Кількісний підхід у методології аналізу успішності базується на оперуванні числовими даними, отриманими під час поточного, модульного та підсумкового контролів. Основним інструментом тут виступає середній бал або GPA (Grade Point Average), що розраховується як відношення суми здобутих оцінок до кількості дисциплін. Проте в сучасних системах аналітики цей показник часто модифікується через введення вагових коефіцієнтів, що враховують складність предметів та їхню значущість для фахової підготовки [10].

Якісний аспект аналізу передбачає дослідження змістовної сторони навчання, де оцінюється не лише бал, а й характер помилок, глибина розуміння матеріалу та творчий підхід до виконання завдань. Такий підхід дозволяє ідентифікувати специфічні труднощі, з якими стикаються студенти при вивченні окремих модулів, що є критично важливим для коригування методичних планів. Методологія інтегрує ці два підходи, створюючи цілісну картину інтелектуального розвитку здобувача освіти [11].

Важливою складовою сучасного аналізу є Educational Data Mining (EDM) (рис. 1.3), що визначається як галузь науки, яка використовує методи інтелектуального аналізу даних для дослідження унікальних типів даних, що надходять з освітніх систем. EDM дозволяє автоматично знаходити закономірності в поведінці студентів, які неможливо помітити при звичайному перегляді відомостей [12].

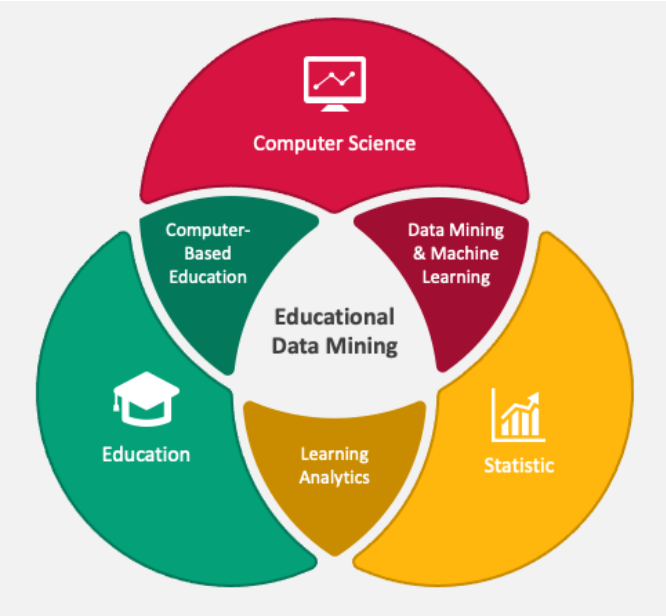



Рис. 1.3 Модель Educational Data Mining

Статистичний фундамент аналізу включає використання мір центральної тенденції, таких як середнє арифметичне, медіана та мода. Медіана є особливо корисною у випадках значного розкиду оцінок у групі, оскільки вона менш чутлива до екстремальних значень, ніж середнє арифметичне [13].

Аналіз дисперсії та середньоквадратичного відхилення в методології аналітики успішності допомагає оцінити ступінь однорідності знань студентів. Високе значення відхилення свідчить про значне розшарування в групі, що вимагає впровадження диференційованого підходу до навчання. Навпаки, низькі показники дисперсії демонструють стабільність групи та рівномірне засвоєння матеріалу більшістю здобувачів, що є ознакою збалансованості навчальної програми [14].

Лонгітюдний метод аналізу, що полягає у вивченні динаміки успішності одного й того самого об’єкта протягом тривалого часу, є незамінним для

відстеження професійного зростання студента. Порівнюючи результати навчання на першому та старших курсах, можна зробити висновки про адаптивність студента до університетського середовища та ефективність обраної ним освітньої траєкторії.

Методологія також передбачає врахування трудомісткості дисциплін через систему кредитів ECTS, що дозволяє зважувати успішність відповідно до обсягу навчального навантаження. Оцінка з дисципліни обсягом у п’ять кредитів має більший вплив на підсумковий рейтинг, ніж оцінка з двокредитного курсу.

Кореляційний аналіз у системі моніторингу успішності спрямований на виявлення зв’язків між результатами навчання та іншими факторами, такими як відвідуваність, активність у цифровій платформі або час, витрачений на виконання домашніх завдань. Розрахунок коефіцієнтів кореляції дає змогу математично підтвердити або спростувати гіпотези про вплив певних форм роботи на фінальну оцінку.

Метод порівняльного бенчмаркінгу дозволяє зіставляти індивідуальні досягнення студента з середніми показниками курсу або факультету. Для адміністрації університету такий аналіз є інструментом виявлення «лідерів» та «аутсайдерів», що необхідно для реалізації програм підтримки або заохочення обдарованої молоді [15].

Предиктивна аналітика є найбільш прогресивною частиною методології, оскільки вона дозволяє прогнозувати майбутню успішність на основі історичних даних. Використання методів регресійного аналізу та класифікації дає можливість виділити групи ризику - студентів, які мають високу ймовірність отримання незадовільних оцінок на іспитах.

Кластерний аналіз застосовується для групування студентів за схожими стратегіями навчання та рівнем успішності. Це дозволяє створювати типологічні профілі здобувачів, для кожного з яких можна розробити специфічні методичні рекомендації. Наприклад, виділення кластера студентів, які мають високі бали лише з технічних дисциплін, допомагає у профорієнтації та виборі спеціалізації в межах основної освітньої програми [16].

Інтерпретація результатів аналізу є завершальним та найважливішим етапом методології, оскільки сухі статистичні дані повинні бути трансформовані в педагогічні висновки.

Методологія аналізу академічної успішності є синтезом математичної точності та педагогічної доцільності. Вона забезпечує об’єктивність оцінювання, прогнозованість результатів та можливість постійного вдосконалення освітнього середовища.

### 1.3 Огляд архітектурних підходів до побудови веб-застосунків

Вибір архітектурного підходу є фундаментом проєктування будь-якої програмної системи, оскільки він визначає стратегію взаємодії компонентів, методи масштабування та загальну надійність застосунку. У сучасній інженерії програмного забезпечення під архітектурою веб-застосунку розуміють сукупність структурних елементів, їхніх зовнішніх властивостей та зв’язків між ними, що забезпечують виконання функціональних та нефункціональних вимог. Якісна архітектура створює потенціал для модифікації системи без докорінної зміни її внутрішньої логіки.

Історично першим та найбільш розповсюдженим підходом до побудови систем була монолітна архітектура, де всі компоненти застосунку об’єднані в єдиний виконуваний модуль. У такій моделі інтерфейс користувача, бізнес-логіка та механізми доступу до даних тісно переплетені, що забезпечує високу швидкість розробки на початкових етапах та спрощує процес розгортання. Однак зі зростанням складності системи аналітики монолітний підхід стає перешкодою для масштабування, оскільки зміна навіть незначного елемента потребує повного перескладання всього програмного комплексу [17].

Альтернативою моноліту є клієнт-серверна архітектура, яка стала базовою парадигмою для сучасної веб-індустрії та передбачає чіткий розподіл функцій між постачальником ресурсів та їхнім споживачем. Клієнт-серверна архітектура - це модель мережевої взаємодії, у якій завдання або робоче навантаження

розподіляються між постачальниками послуг, званими серверами, і замовниками послуг, званими клієнтами.

У межах клієнт-серверної моделі особливого значення набула трирівнева архітектура (three-tier architecture), що передбачає виділення рівнів представлення, логіки та збереження даних. Рівень представлення забезпечує взаємодію з користувачем, рівень логіки відповідає за обробку аналітичних запитів та розрахунок показників успішності, а рівень даних фокусується на ефективному зберіганні інформації в базі даних [18].

Сучасний розвиток веб-технологій призвів до популяризації мікросервісного підходу, який базується на декомпозиції застосунку на набір слабко пов'язаних сервісів. Мікросервісна архітектура - це метод розробки програмного забезпечення, що полягає у створенні системи як сукупності невеликих автономних сервісів, кожен з яких реалізує окрему бізнес-функцію та взаємодіє з іншими через легковагові протоколи. Для системи аналітики це означає можливість виділення модуля збору даних, модуля статистичних розрахунків та модуля генерації звітів у незалежні одиниці, що можуть розроблятися різними мовами [19].

Особливу увагу при огляді архітектур слід приділити концепції Single Page Application (SPA), яка є найбільш актуальною при використанні бібліотеки React. На рисунку 1.4 наведено порівняння концепції SPA та MPA – багатосторінкового застосунку.

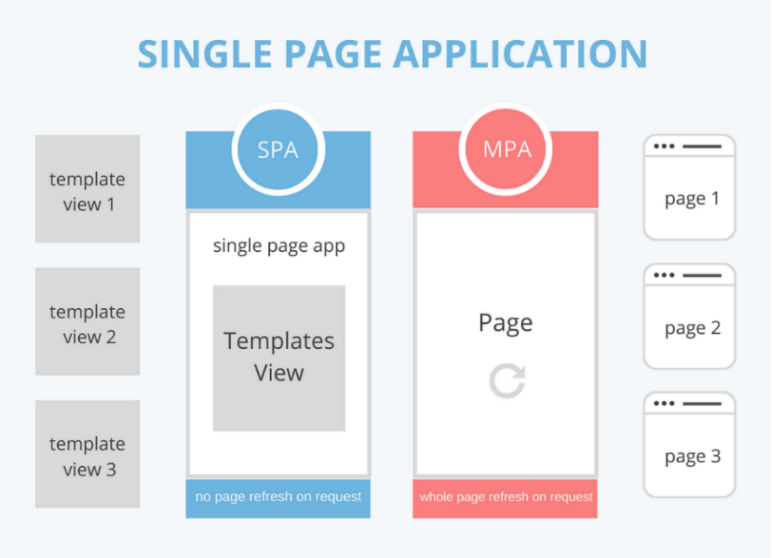



Рис. 1.4 Порівняння концепцій Single Page Application та Multi Page Application

Односторінковий застосунок (SPA) - це веб-додаток або веб-сайт, який взаємодіє з користувачем шляхом динамічного переписування поточної сторінки даними з веб-сервера, замість традиційного методу завантаження цілих нових сторінок [20].

Архітектурний стиль REST (Representational State Transfer) виступає стандартом де-факто для організації зв'язку між клієнтом та сервером у сучасних веб-застосунках. Він базується на використанні стандартних методів протоколу HTTP для маніпуляції ресурсами, що робить систему передбачуваною та легкою для тестування. Застосування RESTful архітектури дозволяє серверній частині на Python надавати дані у форматі JSON, який легко обробляється клієнтською частиною на React, забезпечуючи високу швидкість візуалізації результатів.

Важливим аспектом проєктування є вибір між клієнтським (CSR) та серверним (SSR) рендерингом, що визначає місце формування кінцевого HTML-коду. Клієнтський рендеринг перекладає основне навантаження з формування інтерфейсу на браузер користувача, що ідеально підходить для інтерактивних панелей моніторингу успішності. Серверний рендеринг, навпаки, забезпечує швидше перше завантаження сторінки, що важливо для публічних інформаційних ресурсів, проте для закритих систем внутрішньої аналітики пріоритет зазвичай надається гнучкості CSR [21].

Архітектура веб-застосунку також повинна враховувати принципи безстановості (statelessness), що означає відсутність потреби зберігати інформацію про сесію користувача безпосередньо на сервері. Це дозволяє легко масштабувати систему шляхом додавання нових екземплярів сервера, оскільки кожен запит містить усю необхідну інформацію для своєї обробки, наприклад, у формі токена доступу.

Розглядаючи архітектурні підходи, неможливо оминути питання інтеграції з базами даних через рівень абстракції даних (Data Access Layer). У випадку використання нереляційної СУБД MongoDB архітектура повинна підтримувати схему «документ-орієнтованого» доступу, що відрізняється від традиційного табличного підходу [22].

Подієво-орієнтована архітектура (Event-Driven Architecture) (рис. 1.5) стає актуальною в системах аналітики при необхідності реагування на певні тригери в реальному часі. Наприклад, внесення нової оцінки до бази даних може генерувати подію, яка автоматично запускає перерахунок рейтингу групи та оновлює графіки на екранах користувачів.

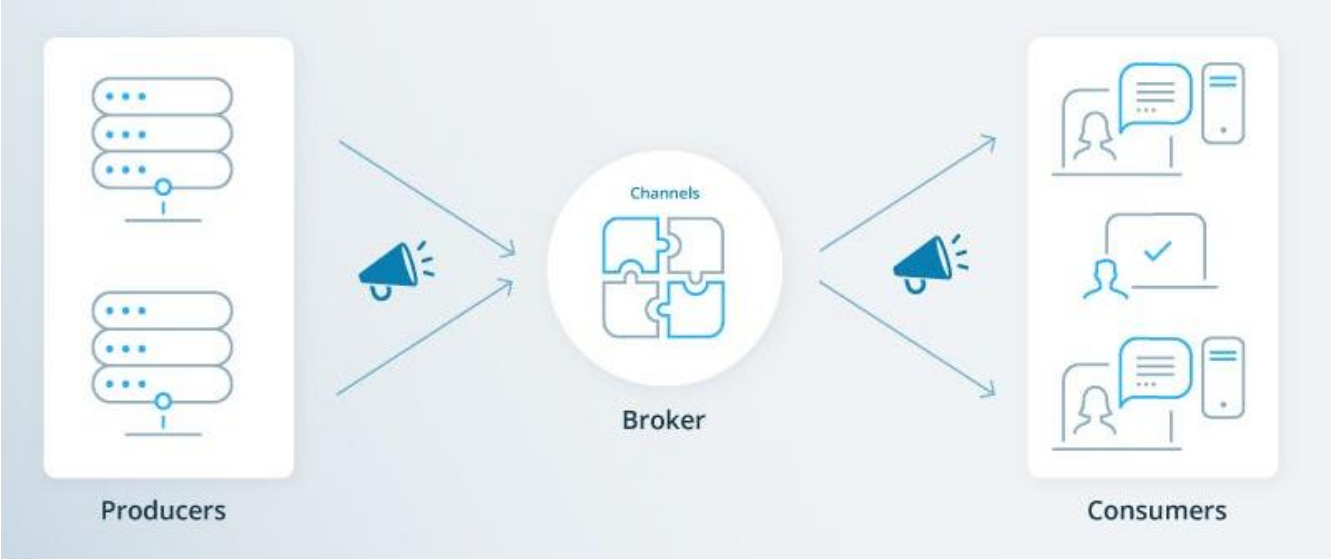



Рис. 1.5 Подієво-орієнтована архітектура Event-Driven Architecture

Архітектурна цілісність системи забезпечується також через дотримання принципів SOLID та використання патернів проєктування на рівні програмного коду. Патерн «Модель-Вид-Контролер» (MVC) або його модифікації залишаються фундаментальними для структурування логіки серверної частини, дозволяючи відокремити обробку вхідних HTTP-запитів від безпосередньої бізнес-логіки аналізу даних.

Безпека в архітектурі веб-застосунків реалізується через багаторівневу систему перевірки повноважень та валідації вхідних даних на кожному етапі обробки. Архітектурний дизайн повинен передбачати ізоляцію чутливих даних про успішність студентів від зовнішнього світу, використовуючи захищені шлюзи (API Gateways) та шифрування каналів зв’язку [23].

Ефективність обраної архітектури безпосередньо впливає на показник Time-to-Market та загальну вартість володіння програмним продуктом. Гнучка та модульна структура дозволяє впроваджувати нові методики аналізу успішності,

розглянуті у попередніх розділах, без необхідності проведення глобального рефакторингу. Отже, архітектурний підхід виступає містком між теоретичними моделями моніторингу та їхньою практичною реалізацією в цифровому середовищі [24].

Підсумовуючи огляд архітектурних підходів, можна зробити висновок, що для розробки системи аналітики успішності студентів найбільш доцільним є використання комбінованого підходу. Поєднання Single Page Application на клієнтській частині, RESTful API на серверній частині та трирівневої структури з нереляційним сховищем даних забезпечує необхідний баланс між продуктивністю, гнучкістю та зручністю використання [25].

## 1.4 Аналіз аналогів

Для визначення можливостей покращення роботи з аналітикою освітніх даних студентів, буде розглянуто декілька вже існуючих таких систем.

### 1.4.1 Canvas LMS

Найбільш відомою з систем контролю навчання (LMS) є Canvas від компанії Instructure (рис. 1.6-1.7). Це одна з найсучасніших та найбільш технологічних таких систем у світі на даний момент. Вона працює за принципом SaaS (програмне забезпечення як послуга), забезпечує керівний склад освітніх закладів динамічними даними про активність студентів.

Ця система працює на базі даних у реальному часі, агрегує інформацію про доступ до сторінок, час перегляду навчальних матеріалів та результати оцінювання. Аналітичні інструменти інтегруються безпосередньо в інтерфейс курсів.

Із переваг Canvas LMS можна виділити такі:

- можливість швидких переходів між показами даних в середньому по групах та окремо по кожному студенту;
- наявність відстежування метрик залученості, таких як кількості взаємодій з ресурсами курсу/дисципліни;

- можливість розсилки сповіщень групам напряму з адміністраторських інтерфейсів.

В свою чергу, з недоліків виділяється наступне:

- закритий вихідний код, тобто неможливість зміни (персоналізації) алгоритмів обробки даних;

- висока складність експорту даних для зовнішньої обробки – для цього потребуються платні надбудови.

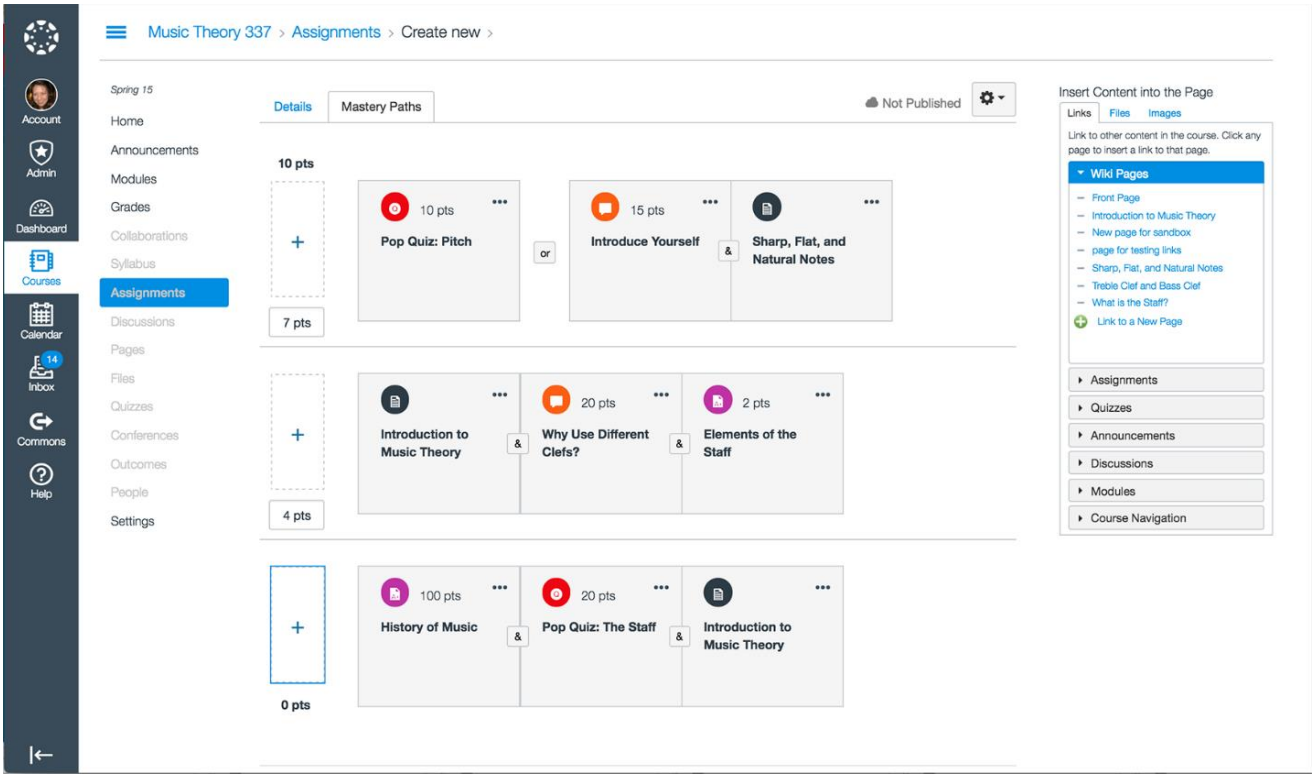



Рис. 1.6 Приклад екранної форми Canvas LMS – створення нового завдання

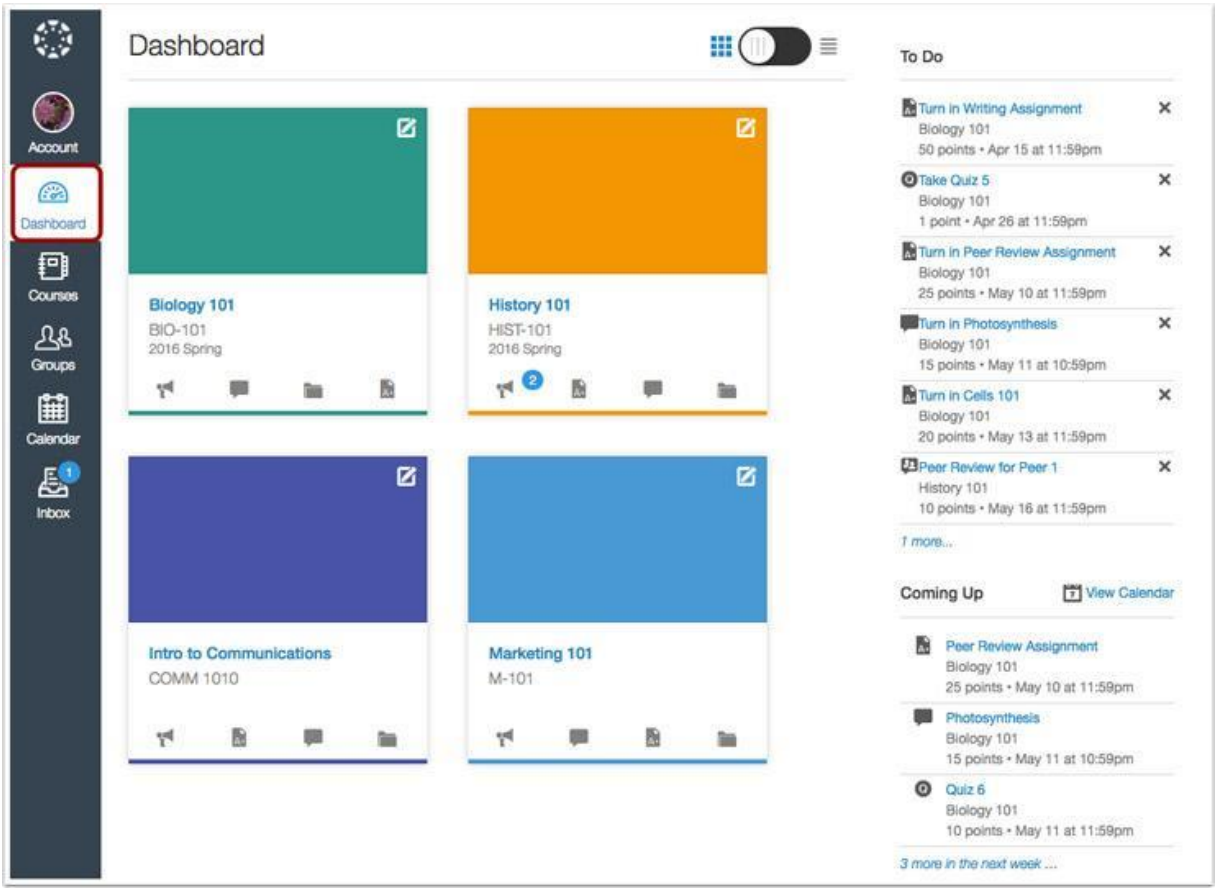



Рис. 1.7 Приклад екранної форми Canvas LMS – головний дашборд

### 1.4.2 Moodle

Наступною системою є широко відома Moodle. Це найпопулярніша система контролю навчання в Україні. Ця система використовує архітектуру плагінів та працює з базами даних SQL. Деякими з функцій Moodle є: дискусійні «форуми», календар подій, створення та управління (з боку викладачів) та здача (з боку студентів) завдань (рис. 1.8), онлайн-тестування (за прикладом на рис. 1.9), новини, оцінювання, завантаження файлів тощо.

У системі Moodle реалізований API для машинного навчання.

Перевагами цієї системи є:

- відкритий код (open-source), що надає величезні можливості персоналізації та – в тому числі – можливість створення власних плагінів під потреби кожного окремого закладу;
- гнучкість функцій системи;

- можливість зручного експорту даних у файли для зовнішнього аналізу (наприклад, JSON).

Із недоліків можна виділити:

- застарілий та іноді перевантажений стандартний інтерфейс, оскільки система існує з 2002 року;
- високий поріг входження для налаштування машинного навчання.

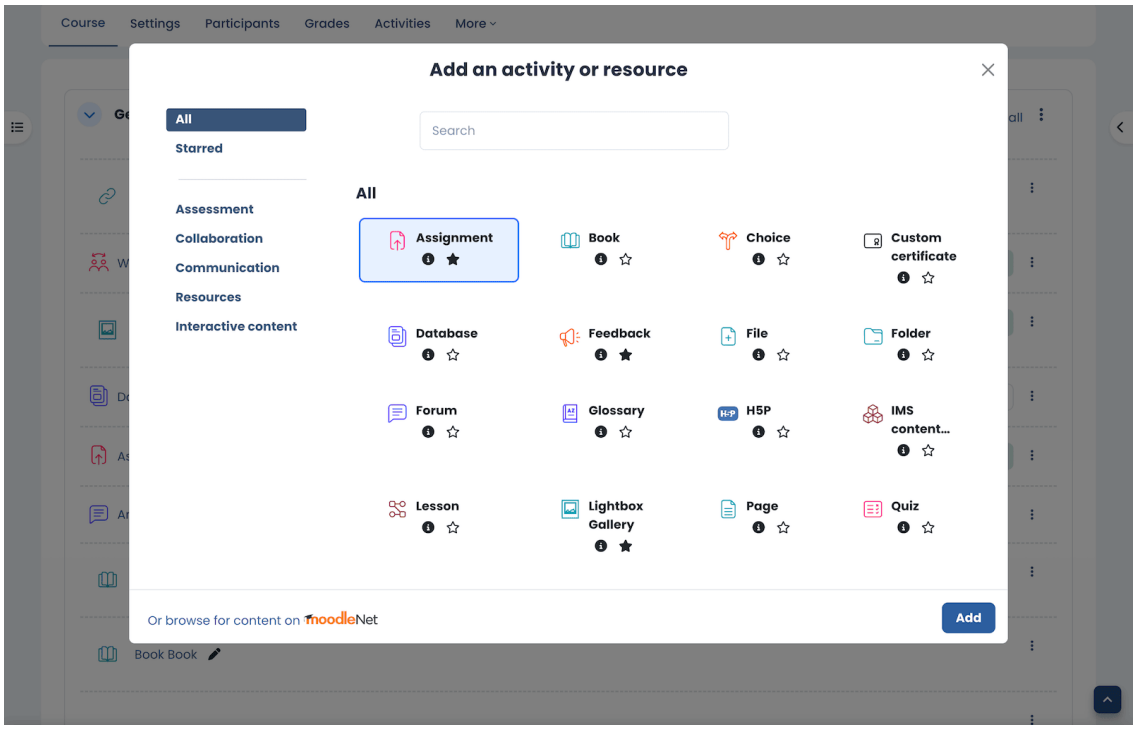



Рис. 1.8 Приклад екранної форми Moodle – додавання ресурсів

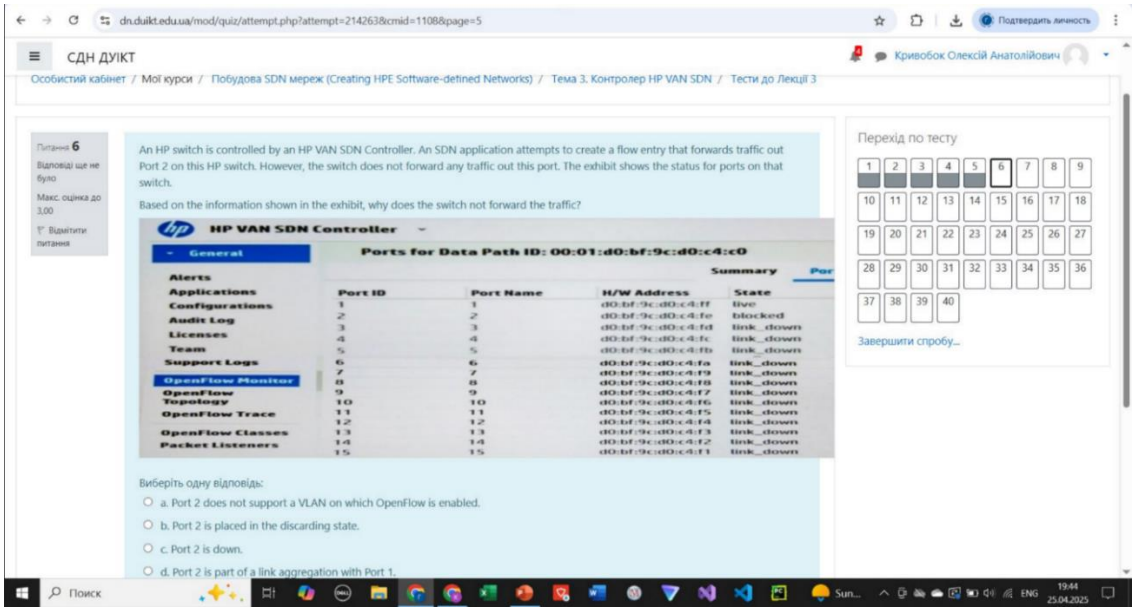



Рис. 1.9 Приклад екранної форми Moodle конкретно у середовищі ДУІКТ – тест із дисципліни «Побудова SDN-мереж» (2025)

### 1.4.3 Blackboard Predict

Ще однією системою, пов’язаною з аналітикою даних студентів, є Blackboard Engage (раніше Blackboard Predict) (рис. 1.10). Це корпоративне рішення, яке фокусується суто на передбаченні успішності студентів. Ця система базується на хмарних обчисленнях та статистичних моделях, які оброблюють дані студентів за попередні роки та визначають вірогідність успішного чи негативного закінчення навчання кожним окремим студентом. Ця система використовується деякими навчальними закладами у США.

Виділяються наступні переваги:

- найвища точність прогнозування серед комерційних систем аналітики освітніх даних;
- глибока інтеграція з системами навчальних закладів.

Також існують значні недоліки, такі як:

- виключно платна основа та вкрай висока вартість;
- відсутність детальних пояснень, наприклад, причин знаходження того чи іншого студента у «групі ризику».

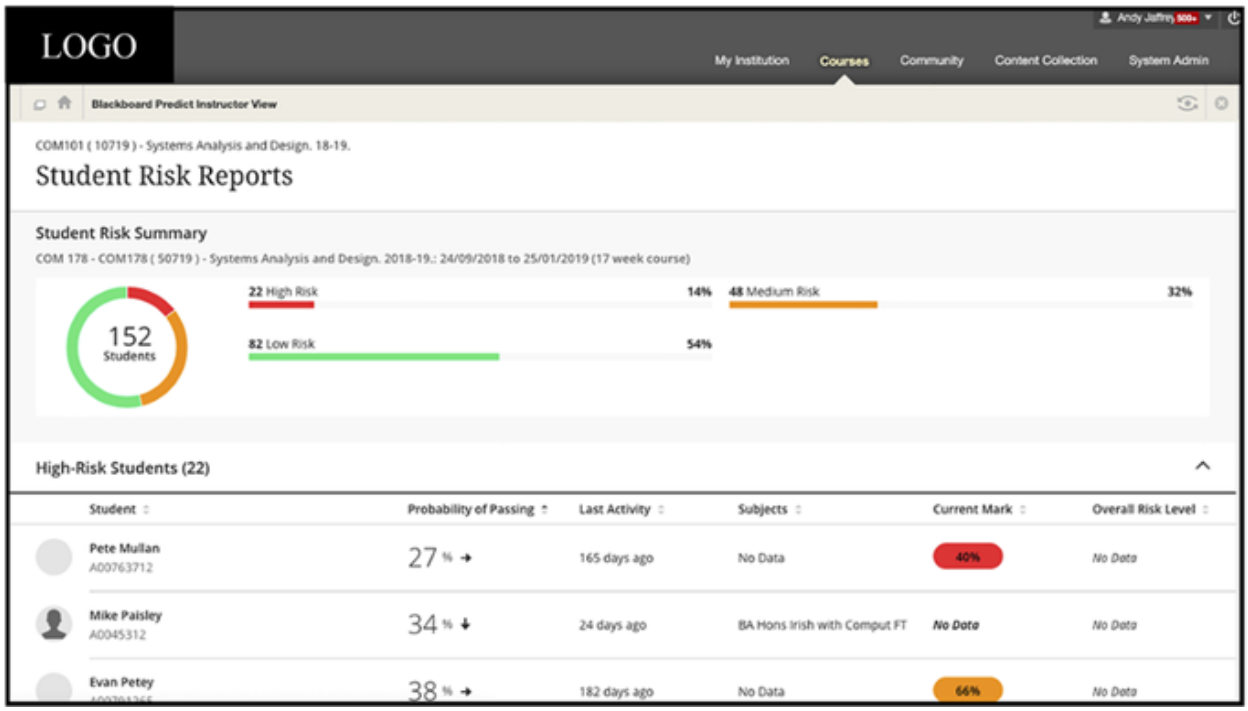



Рис. 1.10 Приклад екранної форми Blackboard Engage (Predict)

У таблиці 1.1 наведено зведені результати аналізу характеристик наведених вище програмних систем.

Таблиця 1.1

Зведені результати аналізу додатків, пов’язаних з аналітикою успішності студентів

| **Параметр** | **Canvas LMS** | **Moodle** | **Blackboard Predict** |
|---|---|---|---|
| **Тип рішення** | SaaS, повноцінна система LMS | Гібридна LMS | SaaS, виключно корпоративна система контролю успішності студентів |
| **Складність впровадження** | Середня | Висока | Висока |
| **Гнучкість UI** | + | - | +/- |
| **Ліцензія** | Пропрієтарна (закритий вихідний код) | Відкритий код | Пропрієтарна (закритий вихідний код) |
| **Масштабованість** | + | +, але залежить від конкретного use-case | + |
| **Підтримувані бази даних** | PostgreSQL | MySQL/PostgreSQL, MariaDB тощо | SQL Server, Oracle |

## 1.5 Визначення вимог до програмного забезпечення

На основі вивчення предметної області та порівняння з існуючими системами складено перелік функціональних та нефункціональних вимог до клієнтської частини розроблюваної системи.

Функціональними вимогами визначено такі:

- Автентифікація адміністратора системи за поштою та паролем.
- Візуалізація освітніх даних студентів, збережених у системі, на інтерактивних дашбордах.
- Наявність інтерактивної фільтрації даних.
- Виконання повного циклу CRUD-операцій для роботи з даними студентів.
- Визначення студентів у «групі ризику» - тобто студентів із середнім балом нижче за 60.

Нефункціональними вимогами визначено наступні:

- Безпека: обмеження доступу до системи для незареєстрованих користувачів та без авторизації.
- Продуктивність: оптимізація клієнтської частини, щоб час первинного завантаження сторінки після авторизації не перевищував 2 секунд.
- Взаємодія з серверною частиною через централізований сервісний шар.
- Можливість навігації без необхідності перезавантаження сторінки.
- Можливість масштабування шляхом наповнення системи великою кількістю студентів.

# 2 ЗАСОБИ РЕАЛІЗАЦІЇ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ

## 2.1 Середовище розробки Cursor

Cursor - це інтегроване середовище розробки, основою якого є широко відомий Visual Studio Code. Середовище використовується для створення програмного забезпечення мовами HTML, CSS, JavaScript, TypeScript, C++, C# та іншими. Однією зі значних особливостей Cursor у порівнянні з іншими IDE є вбудована підтримка інструментів штучного інтелекту та агентів для роботи з програмним кодом.

Середовище підтримує більшість розширень Visual Studio Code, що дозволяє використовувати вже знайомі інструменти для роботи з проєктами. Під час розробки клієнтської частини системи використовувалися можливості підсвітки синтаксису, автозавершення рядків коду, інтегрований термінал, засоби відладки та засоби навігації між файлами проєкту.

Cursor використовувався як основне середовище розробки клієнтської частини системи аналітики успішності студентів. Інтеграція з бібліотеками Node.js, React, інструментом розгортання Vite та системою контролю версій Git дозволило працювати з проєктом без потреби використання жодних сторонніх програм.

Вбудовані засоби роботи зі штучним інтелектом використовувалися для пришвидшення окремих етапів розробки, зокрема під час створення шаблонного коду та перевірки роботи окремих фрагментів системи. Це дозволило прискорити розробку та зменшити кількість типових помилок під час написання коду.

## 2.2 Мова розмітки HTML

HTML є базовою мовою розмітки, що використовується для створення структури вебсторінок та організації контенту в мережі Інтернет. Вона визначає логічну структуру сторінок (документів), дозволяючи браузерам перетворювати та

відображати текстову, графічну, мультимедійну та іншу інформацію. HTML є фундаментальною складовою сучасних вебтехнологій і використовується спільно з CSS та JavaScript для побудови інтерактивних веб-застосунків.

Основною особливістю HTML є використання спеціальних тегів, за допомогою яких описується структура документа. Кожен HTML-документ складається з набору елементів, що формують заголовки, абзаци, таблиці, списки, форми, зображення, посилання та інші компоненти вебінтерфейсу. Сучасний стандарт HTML5 значно розширив наявні раніше функціональні можливості мови, додавши підтримку мультимедіа, семантичних елементів та API для роботи з графікою та локальним сховищем даних.

HTML є декларативною мовою. Вона описує структуру та зміст документа, а не алгоритми його виконання. Для визначення логічних частин сторінки використовуються семантичні теги, такі як <header>, <nav>, <section>, <article>, <footer>,  тощо. Використання семантичної розмітки покращує доступність веб-ресурсів, оптимізацію для пошукових систем та підтримуваність коду.

Визначною характеристикою HTML є платформонезалежність. Документи, створені за допомогою HTML, можуть коректно відображатися на різних операційних системах, пристроях та браузерах за умови підтримки окремим браузером відповідних стандартів. Це забезпечує універсальність веб-технологій та сприяє розвитку багатоплатформних інформаційних систем.

Сучасний HTML також підтримує інтеграцію мультимедійних елементів без використання сторонніх плагінів. Наприклад, теги <video> та <audio> дозволяють відтворювати відеофайли та аудіофайли безпосередньо в браузері. Крім того, елемент <canvas> використовується для динамічного рендерингу графіки та анімацій за допомогою JavaScript.

HTML широко використовується у веб-розробці завдяки простоті освоєння, гнучкості та сумісності з іншими технологіями. У поєднанні з CSS мова дозволяє створювати адаптивні інтерфейси, а інтеграція з JavaScript забезпечує реалізацію інтерактивної поведінки вебсторінок. Важливою перевагою HTML є також

підтримка міжнародних стандартів W3C, що забезпечує стабільність розвитку технології та сумісність між різними браузерами.

Одним із ключових аспектів використання HTML у сучасній розробці є забезпечення доступності веб-ресурсів для користувачів з особливими потребами. Для цього застосовуються спеціальні атрибути, зокрема alt, aria-label, role та інші механізми, які допомагають програмам, таким як екранне озвучення (для людей з порушеннями зору) коректно інтерпретувати вміст сторінки. Таким чином, HTML є не лише засобом структуризації контенту, а й важливим інструментом забезпечення універсального та інклюзивного доступу до інформації.

Отже, HTML виступає основою сучасних веб-технологій та забезпечує створення структурованого, доступного й стандартизованого веб-контенту. Його розвиток та інтеграція з іншими веб-технологіями, які будуть наведені нижче, роблять HTML незамінним інструментом у процесі створення інформаційних систем і веб-застосунків.

### 2.3 Мова стилю сторінок CSS

CSS є мовою стилів, що використовується для опису зовнішнього вигляду та форматування HTML-документів. Основне призначення CSS полягає у відокремленні структури веб-сторінки від її візуального оформлення, що забезпечує гнучкість розробки та спрощує підтримку програмного коду.

CSS дозволяє визначати кольори, шрифти, відступи, розташування елементів, анімації, адаптивність інтерфейсу та загалом зовнішній вигляд веб-ресурсу. Завдяки використанню каскадного механізму стилі можуть успадковуватися та комбінуватися відповідно до визначених правил пріоритетності. Це дозволяє централізовано керувати оформленням великої кількості вебсторінок.

Однією з найважливіших переваг CSS є забезпечення адаптивного дизайну. За допомогою медіазапитів (@media) веб-інтерфейси можуть автоматично змінювати своє оформлення залежно від типу пристрою, розміру екрана або

орієнтації дисплея. Це особливо актуально в умовах широкого використання мобільних пристроїв та планшетів.

CSS підтримує різні моделі компонування елементів, серед яких найбільш популярними є Flexbox та Grid Layout. Flexbox використовується для організації елементів в одному напрямку, тоді як Grid Layout дозволяє створювати складні двовимірні макети сторінок. Ці технології значно спрощують процес створення сучасних адаптивних інтерфейсів.

Ще однією важливою функцією CSS є можливість реалізації анімацій та переходів без обов'язкового використання JavaScript. Властивості transition, transform та animation дозволяють створювати плавні ефекти взаємодії з користувачем, покращуючи ергономіку веб-застосунків.

CSS також забезпечує повторне використання стилів. Один файл стилів може бути підключений до багатьох HTML-документів, що зменшує дублювання коду та полегшує внесення змін до дизайну системи. Крім того, використання засобів попередньої обробки, таких як Sass або Less, дозволяє розширити стандартний функціонал CSS за рахунок змінних, вкладеності, функцій тощо.

У сучасній веб-розробці CSS відіграє ключову роль у створенні користувацького інтерфейсу. Від якості реалізації стилів залежить не лише естетичний вигляд системи, а й зручність використання, швидкість взаємодії та доступність ресурсу для користувачів. CSS активно використовується разом із фреймворками React, Bootstrap, Tailwind CSS та іншими бібліотеками, які дозволяють прискорити процес створення інтерфейсів.

Таким чином, CSS є невід’ємною складовою сучасних вебтехнологій, забезпечуючи ефективне управління візуальним оформленням вебресурсів та підтримуючи створення адаптивних, доступних і зручних інтерфейсів.

### 2.4 Мова програмування JavaScript

JavaScript є високорівневою мовою програмування, що використовується для створення інтерактивних елементів веб-застосунків. Вона забезпечує динамічну

поведінку веб-сторінок, дозволяючи реалізовувати взаємодію з користувачем, обробку подій, асинхронний обмін даними та маніпуляцію елементами HTML-документа.

JavaScript був розроблений компанією Netscape у 1995 році та згодом стандартизований як ECMAScript. Сьогодні ця мова є однією з трьох основних технологій веб-розробки, поряд з HTML та CSS. Вона підтримується усіма сучасними браузерами та широко використовується як на клієнтській, так і на серверній стороні.

Однією з ключових особливостей JavaScript є динамічна типізація та підтримка різних парадигм програмування, включаючи об’єктно-орієнтований (ООП), функціональний та імперативний підходи. Мова дозволяє створювати функції як об’єкти першого класу, що значно підвищує гнучкість програмного коду.

JavaScript активно використовується для роботи з DOM (Document Object Model) - програмним інтерфейсом, що представляє HTML-документ у вигляді дерева об’єктів. За допомогою DOM-маніпуляцій розробник може динамічно змінювати структуру сторінки, стилі елементів та вміст без необхідності повторного завантаження документа.

Сучасний JavaScript підтримує асинхронне програмування через механізми callback-функцій, Promise та async/await. Це дозволяє ефективно працювати з мережевими запитами, файлами та іншими ресурсами без блокування роботи інших частин системи. Асинхронність є критично важливою для створення швидких та масштабованих веб-застосунків.

Важливу роль у розвитку JavaScript відіграють сучасні фреймворки та бібліотеки, такі як React, Angular та Vue.js. Вони значно спрощують розробку складних односторінкових застосунків (SPA) та забезпечують компонентний підхід до побудови інтерфейсів. На серверній стороні JavaScript використовується разом із платформою Node.js, що дозволяє створювати високопродуктивні серверні рішення.

JavaScript також підтримує модульну архітектуру, що дозволяє розділяти програмний код на окремі незалежні компоненти. Це покращує підтримуваність та масштабованість програмного забезпечення. Крім того, мова має потужну екосистему пакетного менеджера npm, який містить мільйони готових бібліотек та інструментів.

Завдяки своїй універсальності JavaScript використовується не лише у веб-розробці, а й – з-поміж іншого - у створенні мобільних застосунків, настільного програмного забезпечення, серверних систем та IoT-рішень. Постійний розвиток стандарту ECMAScript сприяє появі нових можливостей та оптимізації продуктивності мови.

Отже, JavaScript є ключовим інструментом сучасної веб-розробки, який забезпечує створення динамічних, інтерактивних та високофункціональних веб-застосунків. Його популярність, багатоплатформність та потужна екосистема роблять JavaScript однією з трьох найважливіших технологій у сфері інформаційних веб-систем.

### 2.5 Фреймворк React

React є JavaScript-бібліотекою з відкритим вихідним кодом, призначеною для створення клієнтських інтерфейсів веб-застосунків. Основною метою React є спрощення процесу розробки складних, інтерактивних інтерфейсів шляхом використання компонентного підходу та ефективного оновлення елементів сторінки. Сьогодні React належить до найбільш популярних бібліотек frontend-розробки та широко використовується під час створення односторінкових веб-застосунків (SPA).

Ключовою особливістю React є компонентна архітектура. Інтерфейс користувача формується з незалежних компонентів, кожен з яких відповідає за окрему частину функціональності або візуального представлення. Компоненти можуть бути повторно використані в різних частинах застосунку, що значно зменшує дублювання коду та підвищує масштабованість програмного

забезпечення. Крім того, такий підхід сприяє модульності системи та спрощує командну розробку великих проєктів.

Важливу роль у роботі React відіграє концепція Virtual DOM. На відміну від традиційної моделі взаємодії з DOM, React створює віртуальне представлення структури сторінки в пам’яті. Після зміни стану компонента бібліотека виконує порівняння між попереднім і новим станом Virtual DOM, після чого оновлює лише ті елементи реального DOM, які зазнали змін. Такий механізм значно підвищує швидкодію вебзастосунків, особливо в умовах великої кількості динамічних елементів інтерфейсу.

Ще однією важливою концепцією React є односпрямований потік даних (one-way data flow). Дані передаються від батьківських компонентів до дочірніх через властивості (properties), що забезпечує передбачуваність поведінки системи та спрощує процес налагодження. Для зберігання локального стану компонентів використовується механізм state. У сучасних версіях React (16.8 та вище) керування станом часто реалізується за допомогою хуків (hooks), зокрема useState, useEffect, useContext та інших.

Хуки стали важливим етапом розвитку бібліотеки. Вони дозволили використовувати стан і життєвий цикл компонентів без необхідності створення класових компонентів. Це суттєво спростило структуру програмного коду та зробило функціональні компоненти основним підходом до розробки React-застосунків.

React використовує синтаксичне розширення JSX (JavaScript XML), яке дозволяє поєднувати JavaScript-код із HTML-подібною розміткою в одному й тому самому файлі. JSX спрощує опис структури інтерфейсу та робить код зрозумілішим та більш читабельним. Під час компіляції JSX перетворюється на звичайні виклики JavaScript-функцій, що забезпечує сумісність із браузерами.

Важливою перевагою React є велика екосистема інструментів та бібліотек. Для керування глобальним станом системи часто використовуються Redux, Zustand або Context API. Для роботи з маршрутизацією застосовується бібліотека React Router, а для створення серверного рендерингу та оптимізації SEO - фреймворки

Next.js та Remix. Наявність розвиненої екосистеми дозволяє адаптувати React до потреб проєктів різного масштабу та складності.

React також підтримує концепцію декларативного програмування: розробник описує бажаний стан інтерфейсу, а бібліотека автоматично забезпечує його оновлення відповідно до змін даних. Такий підхід значно спрощує створення інтерактивних інтерфейсів та зменшує кількість помилок, пов’язаних із прямою маніпуляцією DOM.

Ще однією важливою характеристикою React є підтримка серверного рендерингу (Server-Side Rendering, SSR). SSR дозволяє генерувати HTML-код на сервері ще до завантаження сторінки користувачем, що позитивно впливає на швидкість першого відображення контенту та оптимізацію для пошукових систем. Це особливо важливо для інформаційних систем, електронної комерції та веб-ресурсів із високими вимогами до SEO.

У сучасній веб-розробці React використовується не лише для створення браузерних застосунків, а й для розробки мобільних програм через платформу React Native. Це дозволяє використовувати єдиний підхід до створення інтерфейсів для різних платформ та зменшує витрати на розробку програмного забезпечення.

Таким чином, React є потужною та гнучкою бібліотекою для створення сучасних веб-інтерфейсів. Завдяки компонентній архітектурі, високій продуктивності, декларативному підходу та широкій екосистемі React став одним із провідних інструментів frontend-розробки та широко застосовується під час створення масштабованих інформаційних систем.

### 2.6 Фреймворк React Router

React Router є спеціалізованою бібліотекою для реалізації маршрутизації у React-застосунках. Вона забезпечує навігацію між різними компонентами без необхідності повного перезавантаження сторінки, що є однією з основних характеристик односторінкових застосунків (SPA). React Router інтегрується з React та дозволяє будувати складні системи навігації, підтримуючи вкладені

маршрути, динамічні параметри URL, захищені маршрути та асинхронне завантаження компонентів.

У традиційних веб-застосунках перехід між сторінками супроводжується запитом до сервера та повторним завантаженням HTML-документа. React Router реалізує інший підхід: маршрутизація здійснюється на стороні клієнта шляхом зміни стану застосунку та оновлення відповідних компонентів інтерфейсу. Це значно підвищує швидкість роботи системи та покращує досвід користувачів (UX).

Основу React Router складає концепція маршруту (Route). Кожен маршрут пов’язує певний URL-шлях із конкретним React-компонентом. Коли користувач переходить за визначеним шляхом, бібліотека автоматично відображає відповідний компонент без необхідності перезавантаження сторінки. Для організації маршрутів використовуються компоненти BrowserRouter, Routes та Route.

Компонент BrowserRouter відповідає за інтеграцію React Router із History API браузера. Завдяки цьому зміна URL відбувається без повного оновлення сторінки, а навігація між маршрутами працює подібно до традиційних веб-сайтів. Компонент Routes містить набір маршрутів, а Route визначає відповідність між URL та компонентом інтерфейсу.

Однією з важливих можливостей React Router є підтримка динамічних маршрутів. Це дозволяє передавати параметри через URL, наприклад ідентифікатор користувача або товару. Такі параметри використовуються для створення динамічного контенту та забезпечують гнучкість інформаційних систем. Отримання параметрів здійснюється за допомогою хука useParams.

React Router також підтримує вкладені маршрути, які дозволяють будувати багаторівневу структуру інтерфейсу. Вкладена маршрутизація особливо корисна для адміністративних панелей, систем керування контентом та великих інформаційних платформ. Такий підхід забезпечує повторне використання компонентів і спрощує організацію навігації.

Для програмного переходу між сторінками використовується хук useNavigate, який дозволяє виконувати навігацію в коді JavaScript. Це важливо для реалізації авторизації, перенаправлень після надсилання форм або обробки

помилок. Крім того, React Router підтримує компонент Link, який використовується для створення навігаційних посилань без перезавантаження сторінки.

У сучасних версіях бібліотеки значну увагу приділено роботі з асинхронними даними. React Router підтримує механізми loaders та actions, які дозволяють отримувати або змінювати дані ще до рендерингу компонента. Це наближає бібліотеку до функціональності повноцінних фреймворків і покращує керування станом навігації.

Також у React Router наявна підтримка захищених маршрутів (ProtectedRoute). Вони використовуються для обмеження доступу до певних сторінок системи лише авторизованим користувачам. Такий механізм є базовим для забезпечення безпеки у системі та є необхідним для веб-застосунків, що містять персональні дані або адміністративні функції.

React Router також забезпечує підтримку lazy loading та code splitting. Це дозволяє завантажувати компоненти лише тоді, коли вони потрібні користувачу, зменшуючи початковий розмір застосунку та покращуючи продуктивність. Для цього використовуються механізми React.lazy() та Suspense.

Важливою перевагою React Router є його тісна інтеграція з екосистемою React. Бібліотека підтримує сучасні можливості React, включаючи Hooks, Suspense та Concurrent Rendering. Це дозволяє створювати масштабовані та високопродуктивні веб-застосунки з гнучкою системою навігації.

Таким чином, React Router є ключовим інструментом для реалізації маршрутизації в React-застосунках. Бібліотека забезпечує швидку навігацію, підтримку складних маршрутів, динамічних URL та асинхронної роботи з даними, що робить її важливим компонентом сучасних інформаційних систем і SPA-застосунків.

## 2.7 Інструмент розгортання Vite

Vite є сучасним інструментом для розробки та збірки веб-застосунків, орієнтованим на високу швидкість роботи та оптимізацію процесу frontend-розробки. Інструмент був створений автором фреймворку Vue.js як альтернатива традиційним системам збірки, які з часом стали недостатньо ефективними для сучасних масштабних JavaScript-застосунків.

Основною метою Vite є забезпечення максимально швидкого запуску середовища розробки та оптимізованої збірки проєкту. На відміну від класичних інструментів, таких як Webpack, Vite використовує нативну підтримку ES-модулів (ECMAScript Modules, ESM) у сучасних браузерах. Завдяки цьому під час запуску застосунку в режимі розробки немає необхідності попередньо збирати весь проєкт. Браузер завантажує лише ті модулі, які необхідні для поточної сторінки, що значно скорочує час старту сервера розробки.

Архітектура Vite складається з двох основних частин: сервера розробки та системи production-збірки. У режимі розробки Vite використовує ES-модулі та технологію Hot Module Replacement (HMR), яка забезпечує миттєве оновлення компонентів після зміни коду без повного перезавантаження сторінки. Це позитивно впливає на продуктивність праці розробника, оскільки зміни інтерфейсу відображаються практично миттєво.

Для production-збірки Vite використовує інструмент Rollup, який забезпечує оптимізацію JavaScript-коду, мінімізацію файлів та розділення коду на окремі фрагменти. Такий підхід дозволяє створювати високопродуктивні веб-застосунки з мінімальним кінцевим розміром. Крім того, Vite підтримує автоматичну оптимізацію залежностей, що додатково підвищує швидкість компіляції.

Важливою перевагою Vite є підтримка сучасних фреймворків та бібліотек, серед яких React, Vue, Svelte, Preact та інші. Для інтеграції з конкретними технологіями використовуються офіційні плагіни, які забезпечують коректну обробку JSX, TypeScript, CSS-модулів та інших особливостей середовища

розробки. У випадку React застосунок зазвичай створюється за допомогою шаблону create-vite, який автоматично налаштовує базову структуру проєкту.

Vite підтримує широкий набір сучасних веб-технологій, включаючи TypeScript, PostCSS, Sass, CSS- та JSON-модулі. Також інструмент має вбудовану підтримку змінних середовища (.env), що дозволяє гнучко налаштовувати поведінку застосунку залежно від середовища виконання. Конфігурація Vite здійснюється через файл vite.config.js (або .ts для TypeScript), у якому визначаються параметри збірки, плагіни, проксі-сервери та інші налаштування.

Однією з важливих характеристик Vite є низьке споживання ресурсів під час розробки. Завдяки використанню нативних ES-модулів і пакетувальника esbuild система забезпечує значно вищу продуктивність у порівнянні з багатьма традиційними інструментами збірки. Esbuild написаний мовою Go та виконує трансформацію коду в десятки разів швидше, ніж Babel або TypeScript Compiler.

Vite також активно використовується в сучасних практиках DevOps та CI/CD. Його проста конфігурація та висока швидкість збірки дозволяють ефективно інтегрувати інструмент у системи автоматичного тестування та розгортання. Завдяки підтримці модульної архітектури Vite легко масштабується для великих проєктів і підтримує розділення коду між командами розробників.

У сучасній веб-розробці Vite став одним із найбільш популярних інструментів frontend-розробки завдяки простоті використання, високій продуктивності та підтримці сучасних стандартів JavaScript. Його застосування дозволяє суттєво скоротити час розробки та покращити продуктивність веб-застосунків.

### 2.8 Хмарна платформа Vercel

Vercel є хмарною платформою для розгортання, хостингу та масштабування веб-застосунків, орієнтованою насамперед на frontend-технології. Платформа була створена компанією Vercel та стала широко відомою завдяки тісній інтеграції з

фреймворком Next.js. Основною метою Vercel є автоматизація процесів розгортання, оптимізації та доставки веб-застосунків кінцевим користувачам.

Однією з ключових особливостей Vercel є підтримка концепції безперервного розгортання. Платформа інтегрується з популярними системами контролю версій, такими як GitHub, GitLab та Bitbucket. Після внесення змін до репозиторію Vercel автоматично виконує збірку проєкту та розгортає нову версію застосунку без необхідності ручного налаштування серверів.

Архітектура Vercel побудована на принципах безсерверних обчислень. Це означає, що розробнику не потрібно безпосередньо керувати фізичними або віртуальними серверами. Платформа автоматично масштабує ресурси залежно від навантаження, забезпечуючи високу доступність та стабільність веб-застосунків. Такий підхід значно спрощує процес адміністрування системи та дозволяє зосередитися безпосередньо на розробці програмного забезпечення.

Важливою складовою Vercel є глобальна CDN-мережа (Content Delivery Network), яка забезпечує швидку доставку статичних ресурсів користувачам із різних регіонів світу. Контент кешується на периферійних серверах, що мінімізує затримки та покращує продуктивність веб-ресурсів. Це особливо важливо для сучасних SPA-застосунків та інформаційних систем із великою кількістю мультимедійного контенту.

Платформа підтримує широкий спектр сучасних frontend-технологій, серед яких React, Vue, Svelte, Angular та інші. Особливо глибока інтеграція реалізована для Next.js, оскільки Vercel є офіційною платформою цього фреймворку. Завдяки цьому автоматично підтримуються механізми SSR, Static Site Generation (SSG), Incremental Static Regeneration (ISR) та Edge Functions.

Однією з важливих функцій Vercel є підтримка preview deployments. Для кожного pull request або окремої гілки система автоматично створює тимчасове середовище розгортання, що дозволяє тестувати нові функції до їх інтеграції в основну версію застосунку. Це значно покращує процес командної розробки та забезпечує контроль якості програмного забезпечення.

Vercel також підтримує безсерверні (serverless) функції, які дозволяють реалізовувати серверну логіку без створення окремого backend-сервера. Такі функції можуть використовуватися для обробки API-запитів, авторизації, взаємодії з базами даних та інших серверних операцій. Безсерверний підхід забезпечує автоматичне масштабування та ефективне використання ресурсів.

Ще однією перевагою платформи є підтримка Edge Middleware та Edge Functions. Ці механізми дозволяють виконувати логіку ближче до користувача на периферійних серверах CDN, що суттєво зменшує затримки під час виконання запитів. Edge-архітектура є важливим напрямом розвитку сучасних хмарних платформ та веб-інфраструктури.

Vercel забезпечує інтеграцію з системами аналітики, моніторингу та управління продуктивністю. Платформа підтримує автоматичне налаштування HTTPS, оптимізацію зображень, кешування ресурсів та моніторинг швидкодії веб-застосунків. Це дозволяє підвищити ефективність роботи інформаційних систем та покращити UX.

У сучасній веброзробці Vercel активно використовується як платформа для розгортання React- та Next.js-застосунків завдяки простоті використання, автоматизації DevOps-процесів і високій продуктивності. Хмарна модель дозволяє значно скоротити витрати на адміністрування серверної інфраструктури та прискорити процес доставки програмного забезпечення.

У даній кваліфікаційній роботі клієнтська частина системи аналітики успішності студентів розміщена на хостингу Vercel, де вона збирається автоматично з відповідного Git-репозиторію.

### 2.9 Система контролю версій GitHub

Git є розподіленою системою контролю версій, яка використовується для відстеження змін у програмному коді та організації спільної роботи над проєктами між багатьма розробниками [15]. GitHub є хмарною платформою для збереження Git-репозиторіїв та управління процесом розробки програмного забезпечення.

Під час створення клієнтської частини системи GitHub використовувався для контролю змін у вихідному коді проєкту. Кожна нова функція або виправлення помилок фіксувалися окремими комітами, що дозволяло відстежувати історію розробки та за потреби повертатися до попередніх версій коду.

GitHub використовувався як основне сховище репозиторію проєкту та забезпечував централізоване збереження вихідного коду. Також, напряму з цього сховища будувався та запускався інтерфейс клієнтської частини на хмарній платформі Vercel, що у свою чергу забезпечило легкий процес модифікації коду та його швидке оновлення у GitHub-сховищі, а зі сховища – на платформі Vercel.

Використання Git та GitHub дозволило спростити процес розробки, тестування та оновлення системи, а також забезпечило надійне збереження змін у проєкті.

# 3 ПРОЄКТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ

## 3.1 Архітектура системи

Система аналізу успішності студентів спроєктована на основі клієнт-серверної архітектури з поділом на три рівні: клієнтський рівень, серверний рівень та рівень збереження даних. Такий підхід дозволяє розділити відповідальність між компонентами системи, спростити підтримку застосунку та забезпечити можливість подальшого розширення функціональності.

На рисунках 3.1-3.2 зображено загальну архітектуру клієнтської частини системи, яка демонструє взаємодію між компонентами інтерфейсу, серверною частиною та базою даних системи.

Клієнтський рівень реалізовано як односторінковий веб-застосунок (SPA), що взаємодіє із сервером через HTTP-запити до REST API. Користувач (адміністратор) працює з інтерфейсом системи, надсилає запити на отримання або зміну даних, а також переглядає результати аналітичної обробки у вигляді таблиць і статистичних показників.

Клієнтську частину реалізовано на основі JavaScript-бібліотек React та React Router. Ця частина містить у собі користувацький інтерфейс системи:

- екран авторизації у системі;
- головний дашборд з основними даними;
- сторінку зі списком студентів, збережених у системі;
- модальні вікна для додавання нових студентів та редагування даних про вже наявних студентів;
- профіль кожного окремого студента з його даними та останніми оцінками;
- сторінку з динамікою навчальних груп, окремих предметів, та розподілом студентів за їх середніми оцінками;
- сторінку порівняння успішності навчальних груп.

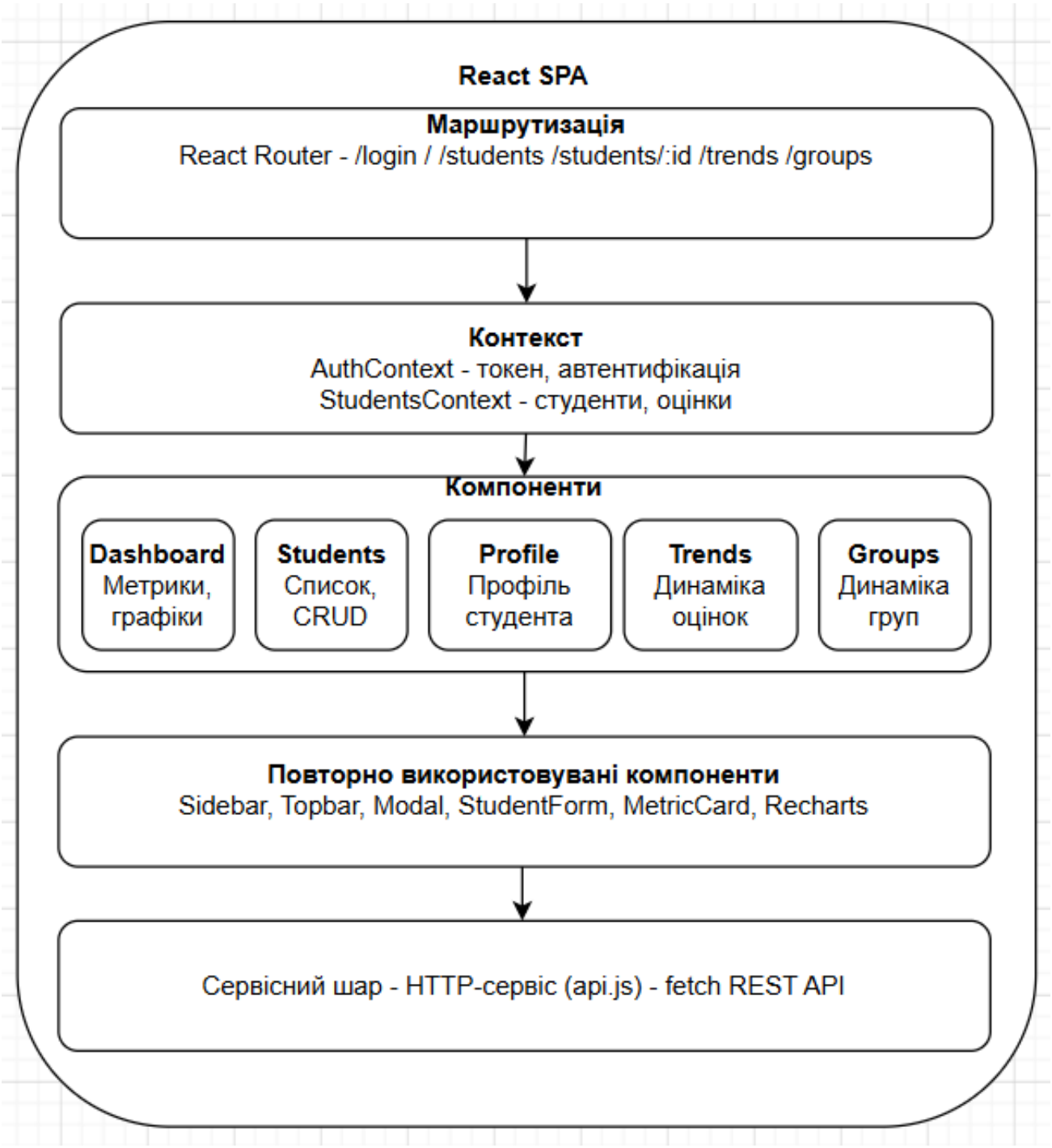



Рис. 3.1 Архітектура системи аналітики успішності студентів

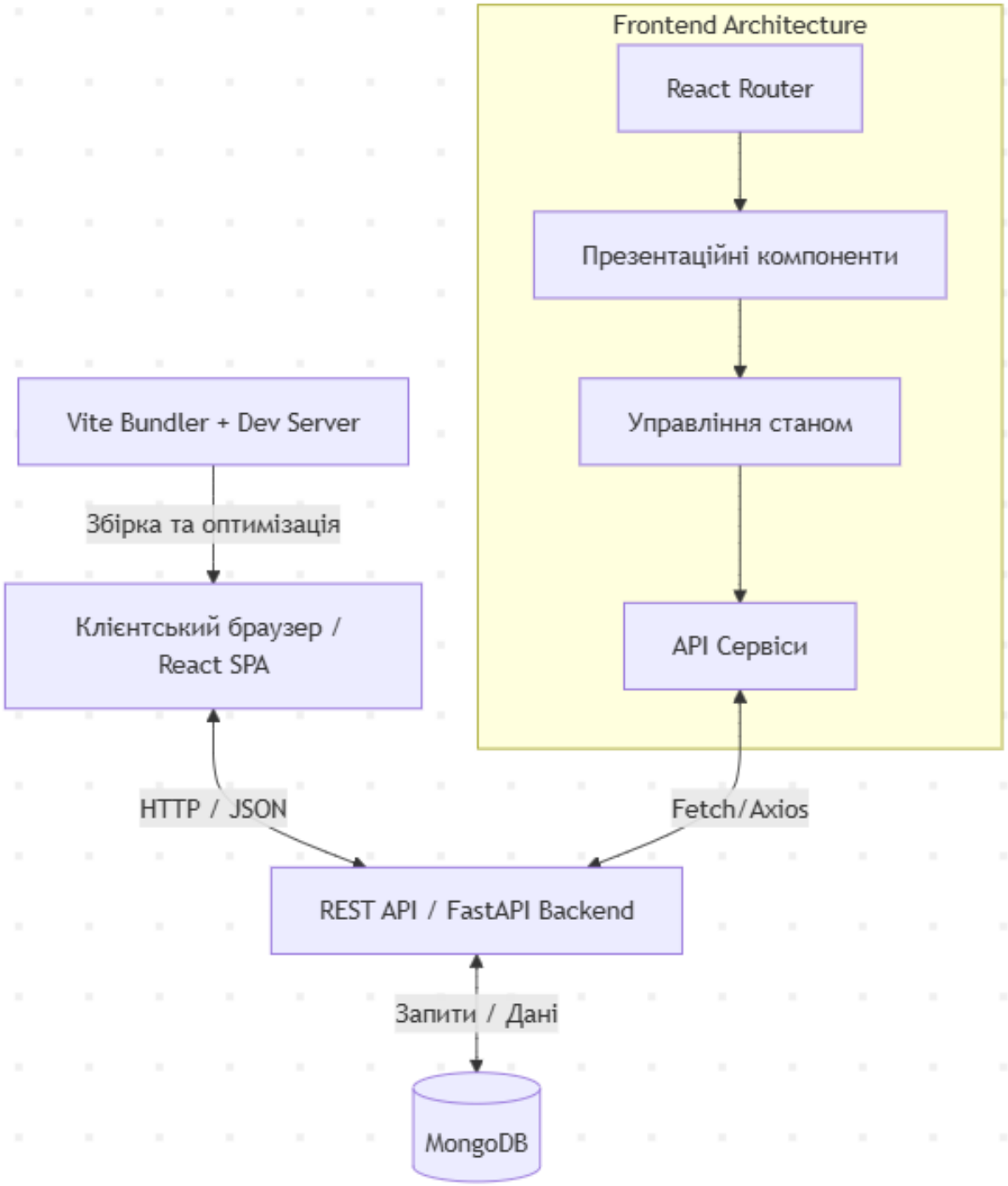



Рис. 3.2 Діаграма архітектури клієнтської частини та взаємодії з серверною частиною

### 3.2 Мапи застосунку

Навігаційна структура системи реалізована за допомогою бібліотеки React Router, яка дозволяє прив’язувати визначені URL-адреси до відповідних компонентів інтерфейсу. Карта застосунку розроблена з урахуванням логіки роботи користувача й розділена на дві основні зони: публічну та приватну (авторизовану). Це гарантує, що неавторизовані користувачі не матимуть доступу до системи.

Публічна частина містить маршрут автентифікації (/login), де користувач вводить свої облікові дані. Після успішної перевірки токена система перенаправляє користувача до приватної зони, яка обгорнута у компонент захищеного маршруту (Protected Route). Базовим маршрутом приватної зони є головний дашборд (/), де відображається зведена статистика та ключові метрики системи аналітики.

Для управління об’єктами системи виділено окремі гілки маршрутів. Маршрут /students веде до загального списку студентів у системі, тоді як динамічний маршрут /students/:id дозволяє переглядати детальну аналітику конкретного студента за його унікальним ідентифікатором, завантажуючи відповідні дані з сервера. За маршрутами /trends та /groups відповідно представлено сторінки динаміки навчальних груп за наявними в системі даними та загальний список груп із їх середніми балами. Мапу системи наведено на рисунку 3.3; розроблювана система має назву “SDF”.

Така структура маршрутизації робить застосунок інтуїтивно зрозумілим, дозволяє користувачам ділитися посиланнями на конкретні сторінки аналітики, а також сприяє ефективному розподілу коду. Завдяки лінивому завантаженню маршрутів скрипти для певних сторінок завантажуються лише тоді, коли користувач фактично переходить на них, що суттєво зменшує початковий час завантаження системи.

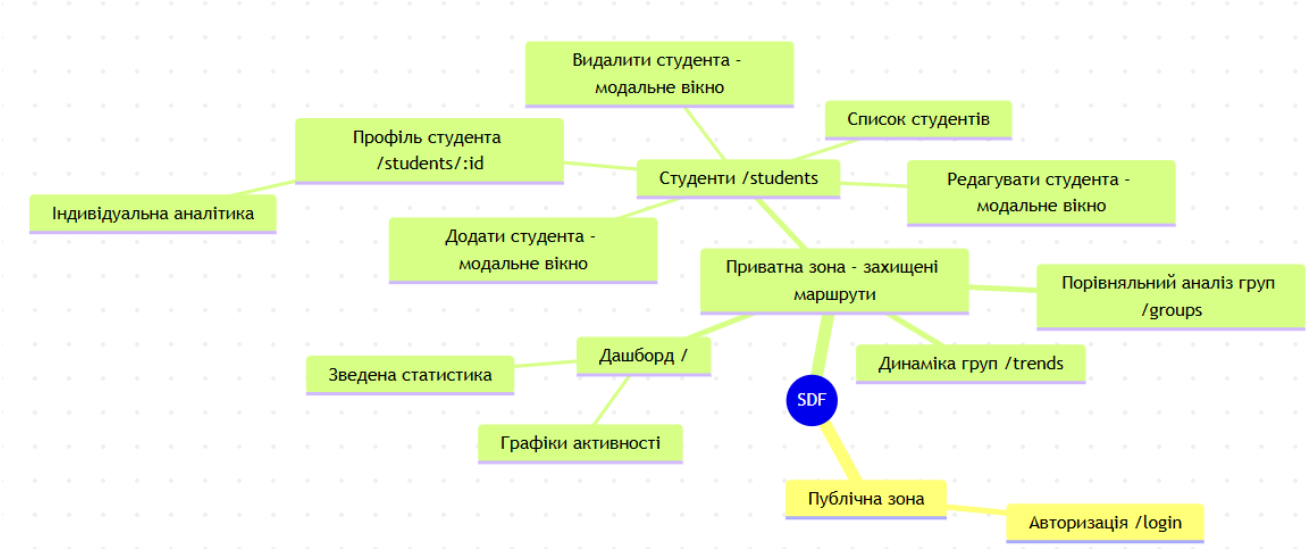



Рис. 3.3 Мапа застосунку

### 3.3 Діаграма компонентів системи

Компонентна модель системи базується на принципах модульності та повторного використання коду, що є фундаментальними для React. Увесь користувацький інтерфейс розбито на ієрархічне дерево незалежних блоків-компонентів, кожен з яких відповідає за відображення певної частини екрану та обробку відповідних подій користувача. Діаграму компонентів наведено на рисунку 3.4.

Кореневим компонентом є App, який слугує точкою входу та обгорткою для базових провайдерів контексту та системи маршрутизації. Нижче за ієрархією розташовуються макетні компоненти (Layouts), які визначають загальну структуру сторінок, включаючи навігаційну панель (Topbar) та бічне меню (Sidebar). Вони дозволяють уникнути дублювання коду при переході між різними розділами системи аналітики.

Функціональні компоненти розділені за логічними доменами. Наприклад, модуль аналітики включає такі компоненти, як Dashboard, PerformanceChart (для візуалізації графіків успішності) та StatisticsCard. Модуль управління студентами складається з StudentList (список), StudentProfile та StudentForm (форми для

додавання або редагування даних). Кожен з цих великих компонентів складається з менших, атомарних елементів інтерфейсу: кнопок, полів введення, випадаючих списків.

Така декомпозиція значно спрощує підтримку та масштабування системи. Ізольованість компонентів дозволяє вносити зміни у візуальне представлення, не порушуючи при цьому логіку інших частин застосунку. Крім того, це сприяє ефективній розробці та дозволяє легко інтегрувати сторонні UI-бібліотеки за необхідності.

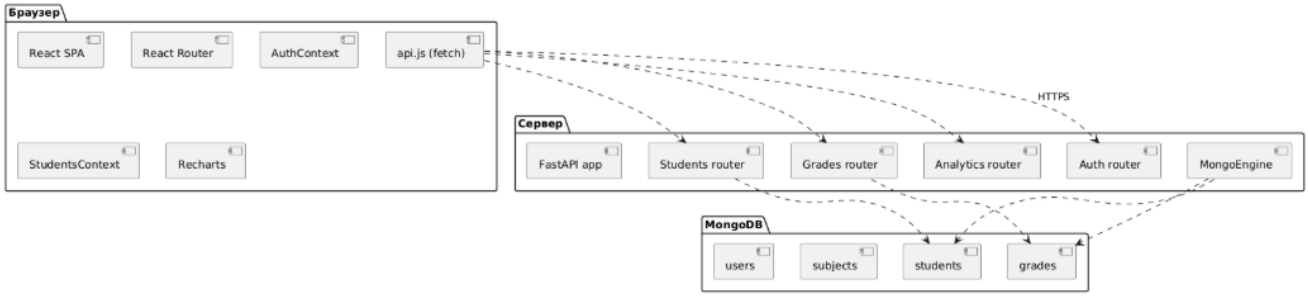



Рис. 3.4 Діаграма компонентів системи

### 3.4 Діаграма варіантів використання

Діаграма варіантів використання описує функціональні вимоги до клієнтської частини системи з точки зору взаємодії кінцевих користувачів з інтерфейсом. Основним актором у цій системі виступає «Адміністратор», який ініціює усі процеси. Система надає йому інструментарій для повноцінного ведення та аналізу освітніх даних.

Першим ключовим прецедентом є «Авторизація», який є обов’язковою умовою (include) для доступу до всіх інших функцій системи. Після входу в систему актор автоматично направляється на головний дашборд та отримує можливість взаємодіяти з усіма частинами системи. Ці частини представляють

собою прецеденти: «Управління студентами», «Динаміка груп», «Порівняння груп».

Окремим вагомим блоком є аналітичні можливості системи. Актор може ініціювати прецедент "Перегляд індивідуальної аналітики успішності", який система обробляє шляхом агрегації даних та побудови візуальних графіків. Також передбачено прецедент "Генерація зведеного звіту по групі", що дозволяє викладачам швидко оцінювати загальну успішність колективу та виявляти тенденції у навчальному процесі.

Усі ці дії виконуються через інтуїтивний веб-інтерфейс, спроєктований таким чином, щоб мінімізувати навантаження на користувача. Візуальний зворотний зв’язок (повідомлення про успішне збереження, індикатори завантаження) є невід’ємною частиною кожного прецеденту, забезпечуючи прозорість процесів та інформування користувача про поточний стан системи.

Діаграму варіантів використання представлено на рисунку 3.5.

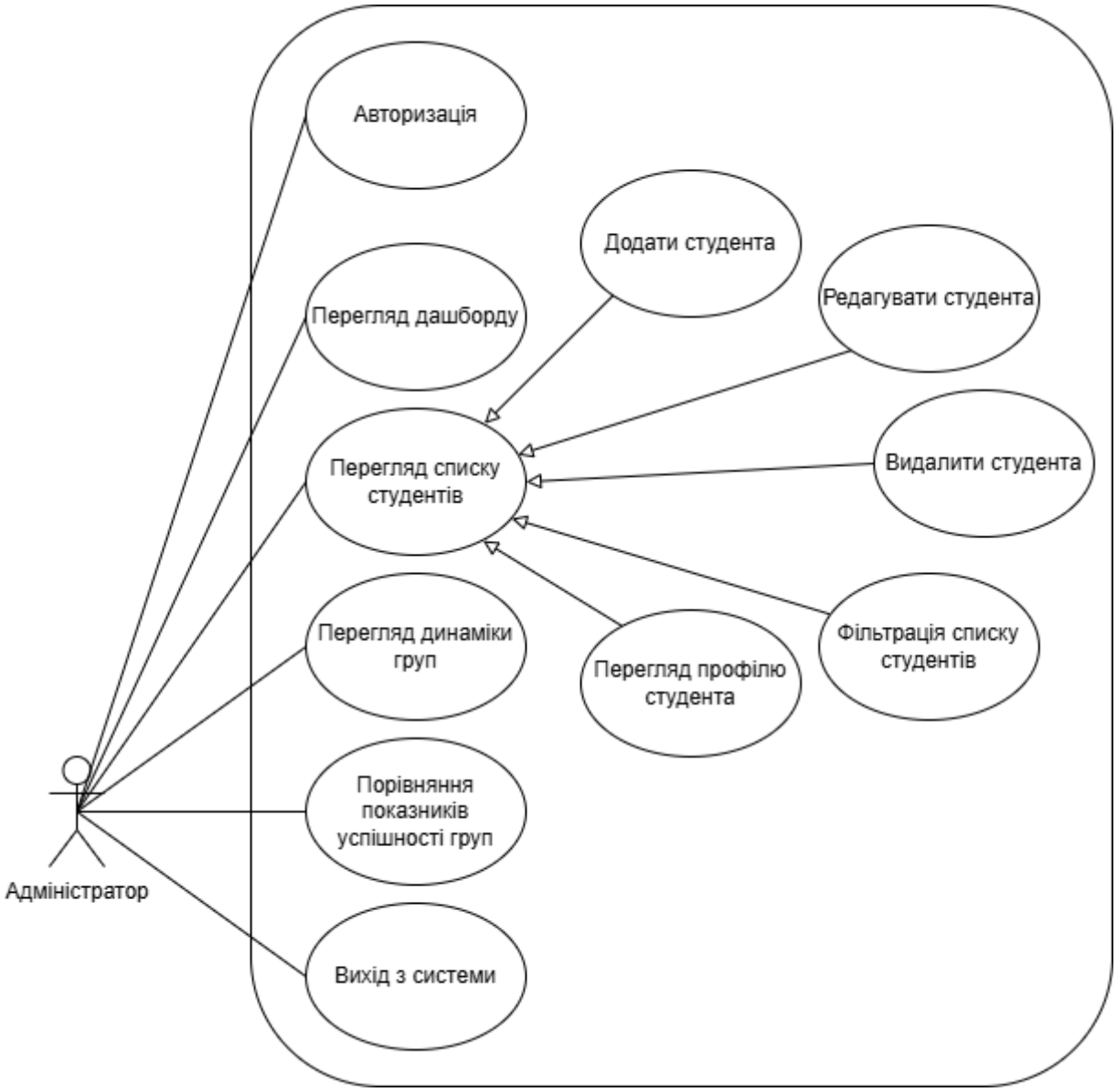



Рис. 3.5 Діаграма варіантів використання

### 3.5 Діаграма послідовності роботи системи

Діаграма послідовності загального рівня демонструє процес ініціалізації клієнтського застосунку та первинного завантаження даних. Процес починається, коли користувач відкриває застосунок у браузері та авторизується у системі. React-додаток ініціалізується, після чого компонент маршрутизатора перевіряє наявність дійсного токена сесії у локальному сховищі (Local Storage).

Якщо сесія дійсна, клієнт формує асинхронний HTTP GET запит до FastAPI сервера для отримання початкових даних дашборду (загальна кількість студентів, середній бал по курсу тощо). Сервер приймає запит, перевіряє авторизацію та звертається до MongoDB для вибірки та агрегації відповідних документів.

Після формування відповіді сервер повертає клієнту дані у форматі JSON. React-компонент приймає ці дані, оновлює свій внутрішній стан (state) та ініціює процес перемальовування інтерфейсу. Користувач бачить на екрані актуальні графіки та статистику. Протягом усього процесу обміну даними клієнт відображає індикатори завантаження, що забезпечує неперервність UX. Цей процес узагальнено у діаграмі послідовності на рисунку 3.6.

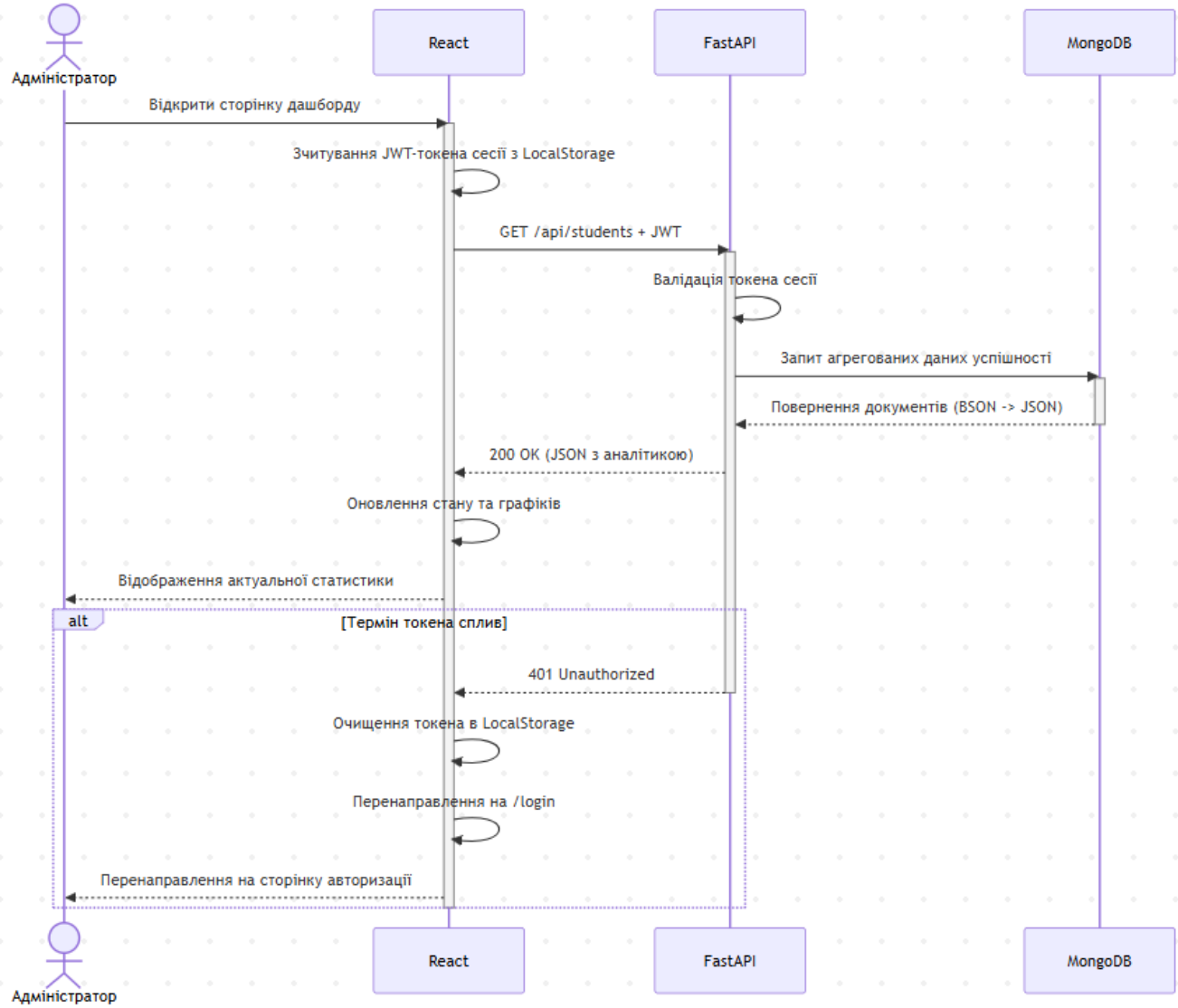



Рис. 3.6 Діаграма послідовності роботи системи

### 3.6 Діаграма послідовності додавання студента

Процес додавання нового студента відображає взаємодію форм вводу на клієнті з серверною частиною. Користувач переходить на сторінку списку студентів, натискає кнопку «Додати студента» та заповнює необхідні поля у спливаючому модальному вікні. Під час введення клієнтська частина виконує попередню валідацію даних (перевірка формату email, обов’язкових полів), що дозволяє зменшити кількість хибних запитів до сервера.

Після натискання кнопки «Зберегти», клієнтська бізнес-логіка (API Service) формує HTTP POST запит, передаючи об’єкт з даними студента у тілі запиту (payload). Серверна частина приймає запит, проводить остаточну серверну валідацію за допомогою схем Pydantic та створює новий запис у відповідній колекції MongoDB.

У разі успішного збереження сервер повертає статус 200 OK та дані створеного об’єкта. Клієнтська частина обробляє цю відповідь, виводить спливаюче повідомлення про успішне додавання та автоматично оновлює список студентів, додаючи туди новий запис без перезавантаження сторінки. Якщо ж виникає помилка (наприклад, дублювання даних), клієнт її перехоплює й відображає відповідне повідомлення в інтерфейсі форми.

Діаграму послідовності додавання студента наведено на рисунку 3.7.

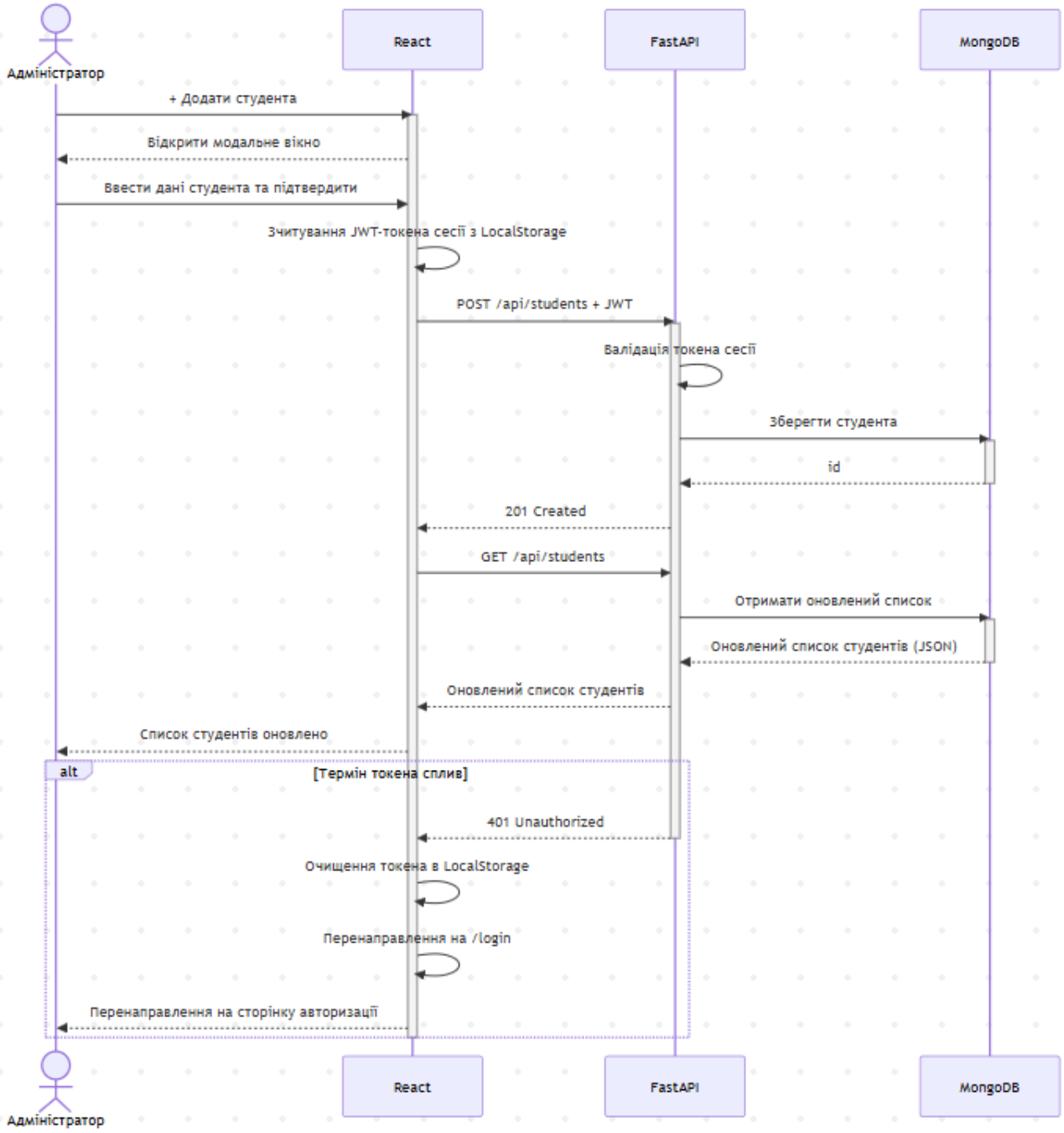



Рис. 3.7 Діаграма послідовності додавання нового студента

# 4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ

## 4.1 Структура проєкту

Реалізація клієнтської частини розробленої системи базується на сучасному стеку технологій веб-розробки. Основу проєкту складає JavaScript-бібліотека React у поєднанні з інструментом збірки Vite, що забезпечило високу швидкість компіляції та ефективне HMR під час розробки. Безпосереднє написання коду здійснювалося у середовищі розробки Cursor, яке надає розширені можливості для рефакторингу та інтелектуального автодоповнення коду, що значно пришвидшило процес розробки ПЗ.

Файлова структура проєкту побудована за модульним принципом, що забезпечує масштабованість та легкість навігації по кодовій базі:

- Директорія src/components/ містить у собі логіку перенаправлення неавторизованих користувачів та UI-елементи, що використовуються на усіх сторінках застосунку: модальні вікна для роботи зі студентами, модулі метрик, бічна та навігаційна панелі.
- Директорія src/pages/ включає опис власне сторінок застосунку, які об’єднують у собі дрібніші елементи та відповідають за відображення конкретних маршрутів (/students, /trends тощо).
- Директорія src/context/ містить провайдери контексту для авторизації та списку студентів.
- Директорія src/data/ містить тестувальні (mock) дані, які використовувалися для тестування роботи клієнтської частини до її підключення до серверної частини та БД з реальними даними.

- Файл src/api.js представляє собою централізований сервісний шар, який обробляє усі HTTP-запити та контролює обмін даними із серверною частиною.

Такий поділ дозволив чітко розмежувати відповідальність між презентаційною логікою та логікою отримання даних. Структуру проєкту графічно наведено на рисунку 4.1.

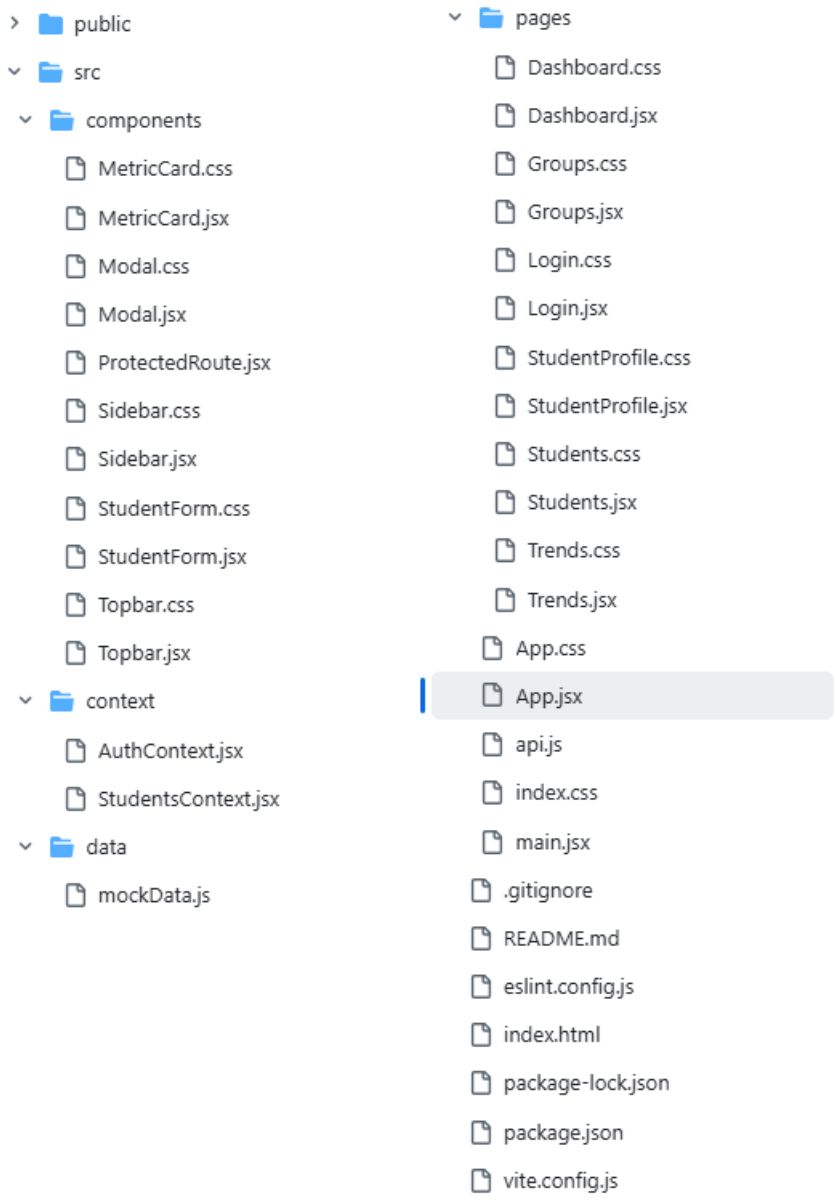



Рис. 4.1 Структура проєкту

### 4.2 Ключові модулі системи

Одним із центральних елементів є модуль маршрутизації, реалізований за допомогою бібліотеки React Router. У кореневому компоненті App.jsx визначено дерево маршрутів. Доступ до усіх маршрутів, окрім авторизації (/login), заборонено для неавторизованих користувачів за допомогою компонента ProtectedRoute. Цей компонент перевіряє наявність дійсного токена авторизації у локальному сховищі браузера та, у разі його відсутності, перенаправляє користувача на сторінку входу.

Модуль відображення списку студентів та їх успішності побудований з використанням хуків стану useState та життєвого циклу useEffect (рис. 4.2). Під час рендерингу компонента здійснюється асинхронний виклик до сервісного шару для отримання актуальних даних.

```
22  export default function Students() {
23      const navigate = useNavigate()
24      const { students, loading, error, addStudent, editStudent, removeStudent } = useStudents()
25
26      const [search, setSearch] = useState('')
27      const [statusFilter, setStatus] = useState('all')
28      const [groupFilter, setGroup] = useState('all')
29      const [showAdd, setShowAdd] = useState(false)
30      const [editTarget, setEditTarget] = useState(null)
31      const [deleteTarget, setDeleteTarget] = useState(null)
32      const [credentials, setCredentials] = useState(null)
33      const [saving, setSaving] = useState(false)
34      const [saveError, setSaveError] = useState('')
```

Рис. 4.2 Фрагмент модуля відображення списку студентів

Для управління формами - наприклад, додавання нового студента або редагування оцінок) - реалізовано контрольовані компоненти (рис. 4.3). Кожна зміна в полі вводу автоматично оновлює відповідний стан у React, що дозволяє виконувати миттєву валідацію даних на стороні клієнта ще до їх відправки на сервер. Це мінімізує кількість хибних запитів до бази даних та покращує UX завдяки швидкому зворотному зв’язку (наприклад, підсвічування некоректно введеного імені чи прізвища студента).

```
async function handleAdd(data) {
  setSaving(true)
  setSaveError('')
  console.log('handleAdd called, data:', data, 'addStudent:', typeof addStudent)
  //if (!showAdd) return
  //addStudent(data)
  //setShowAdd(false)
  try {
    const res = await addStudent(data)
    setShowAdd(false)
    if (res.credentials) setCredentials(res.credentials)
  } catch (e) {
    setSaveError(e.message)
  } finally {
    setSaving(false)
  }
}
```

```
async function handleEdit(data) {
    if (!editTarget) return
    setSaving(true)
    setSaveError('')
    try {
      await editStudent(editTarget.id, data)
      setEditTarget(null)
    } catch (e) {
      setSaveError(e.message)
    } finally {
      setSaving(false)
    }
}

async function handleDelete() {
  if (!deleteTarget) return
  setSaving(true)
  try {
    await removeStudent(deleteTarget.id)
    setDeleteTarget(null)
  } catch (e) {
    setSaveError(e.message)
  } finally {
    setSaving(false)
  }
}
```

Рис. 4.3 Функції роботи з даними студентів

### 4.3 Реалізація зв’язку з серверною частиною системи

Для забезпечення стабільної та безпечної комунікації з сервером, реалізованим на базі FastAPI, у клієнті наявний окремий сервісний шар. Увесь код, відповідальний за HTTP-запити, реалізовано у файлі api.js.

Основою цього шару є налаштований екземпляр клієнта (за допомогою fetch), який містить базову конфігурацію: базову URL-адресу сервера, стандартні заголовки (authHeaders) та правила обробки таймаутів. Якщо сервер повертає статус 401 Unauthorized (термін дії токена закінчився), сервісний шар автоматично ініціює процес виходу з системи (logout) та перенаправляє користувача на екран авторизації, очищуючи локальне сховище.

Реалізація конкретних функцій, в тому числі CRUD-циклу управління даними, представляє собою набір експортованих асинхронних функцій (рис. 4.4). Наприклад:

- fetchStudents() - виконує fetch-запит для отримання списку студентів у системі.
- createStudent(data) - виконує POST-запит, передаючи об’єкт даних, перетворений на формат JSON.
- fetchGrades(), fetchSubjects() – виконують fetch-запити для отримання оцінок студентів та відповідних їм навчальних дисциплін.
- fetchRiskGroup() – виконує fetch-запит, який отримує з сервера список студентів у групі ризику, тобто таких студентів, чий середній бал є нижчим за 60.

Ця абстракція робить React-компоненти незалежними від деталей реалізації мережевого протоколу.

```
46  export async function createStudent(data) {
47    const res = await fetch(`${BASE}/api/students`, {
48      method: 'POST',
49      headers: authHeaders(),
50      body: JSON.stringify(data),
51    })
52    return handleResponse(res)
53  }
54
55  export async function updateStudent(id, data) {
56      const res = await fetch(`${BASE}/api/students/${id}`, {
57          method: 'PUT',
58          headers: authHeaders(),
59          body: JSON.stringify(data),
60      })
61      return handleResponse(res)
62  }
63
64  export async function deleteStudent(id) {
65    const res = await fetch(`${BASE}/api/students/${id}`, {
66      method: 'DELETE',
67      headers: authHeaders(),
68    })
69    return handleResponse(res)
70  }
```

Рис. 4.4 Фрагмент коду сервісного шару – управління даними студентів

### 4.4 Тестування функціональності

Тестування клієнтської частини проводилося методом ручного функціонального тестування через створений інтерфейс. Це дозволило перевірити коректність роботи бізнес-логіки системи та зручність інтерфейсу.

Перевірено логіку авторизації у системі. Неавторизовані користувачі при спробі переходу на головну сторінку системи (дашборд) автоматично перенаправляються на сторінку входу (рис. 4.5).

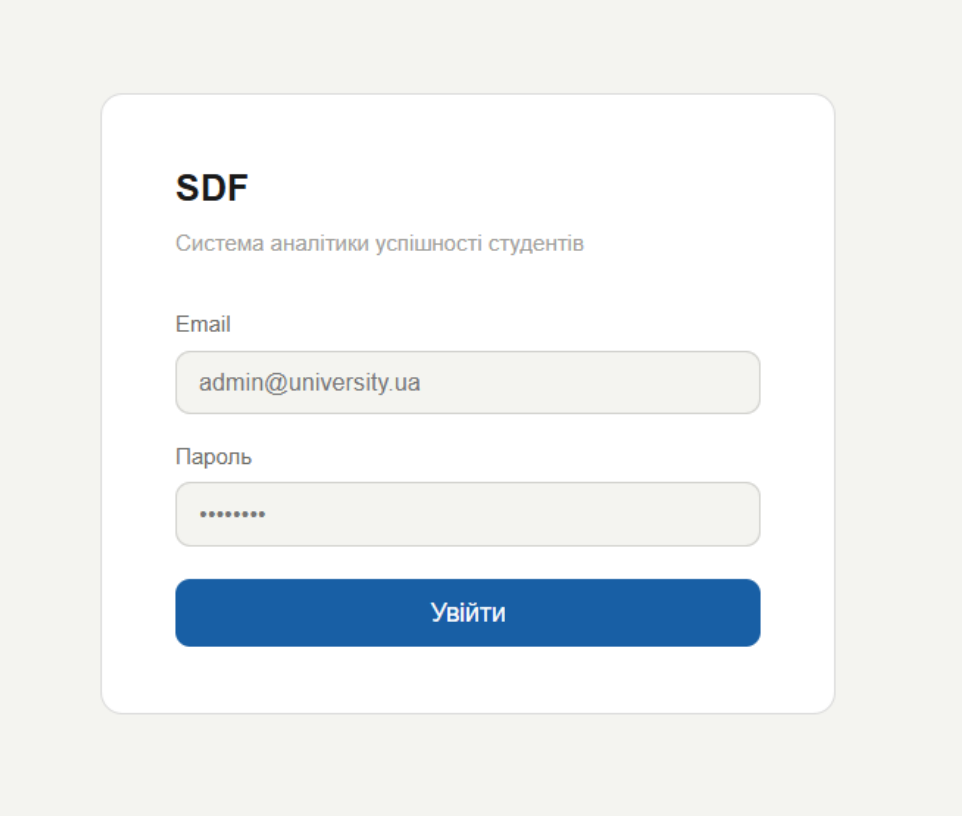



Рис. 4.5 Авторизація в системі

Перевірено коректність відображення списку студентів та його фільтрацію за такими параметрами, як номер групи та статус студента за його оцінками (рис. 4.6); роботу модальних вікон та CRUD-циклу управління даними студентів (рис. 4.7).

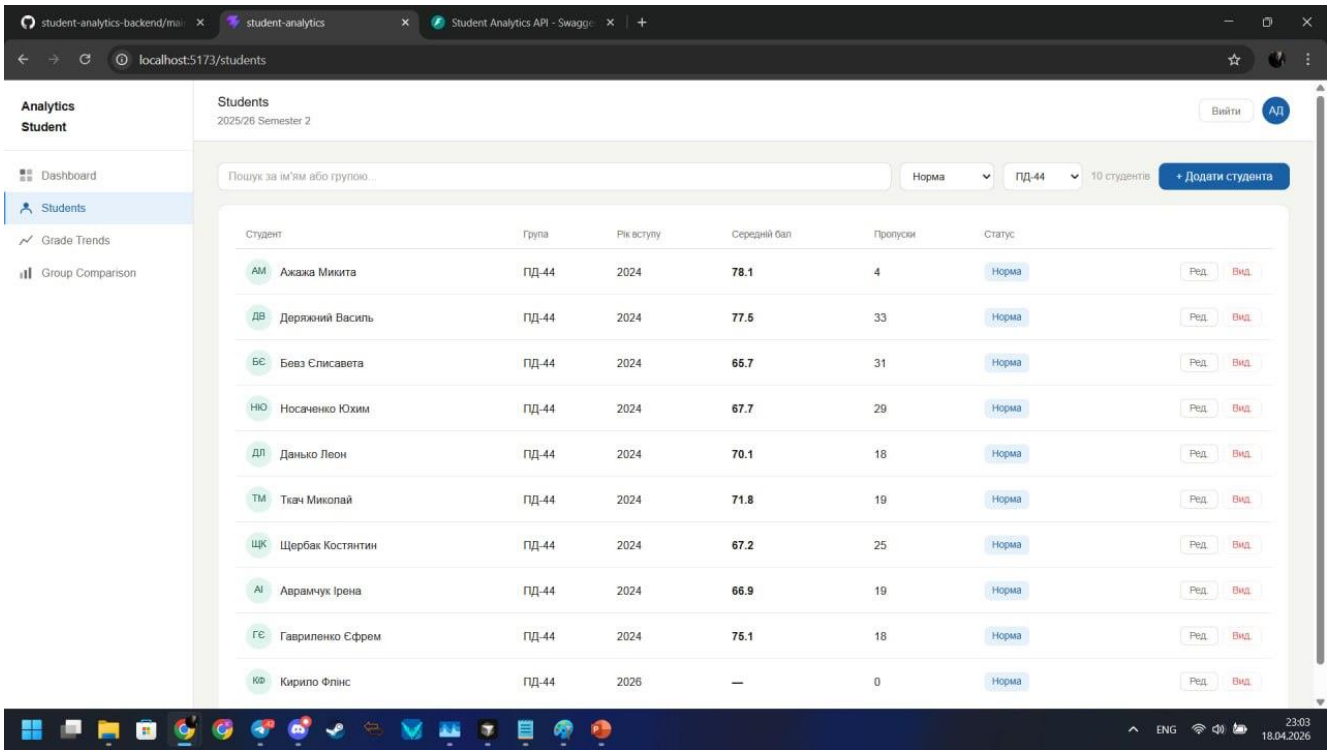



Рис. 4.6 Список студентів з реальними даними із сервера та з застосованими фільтрами

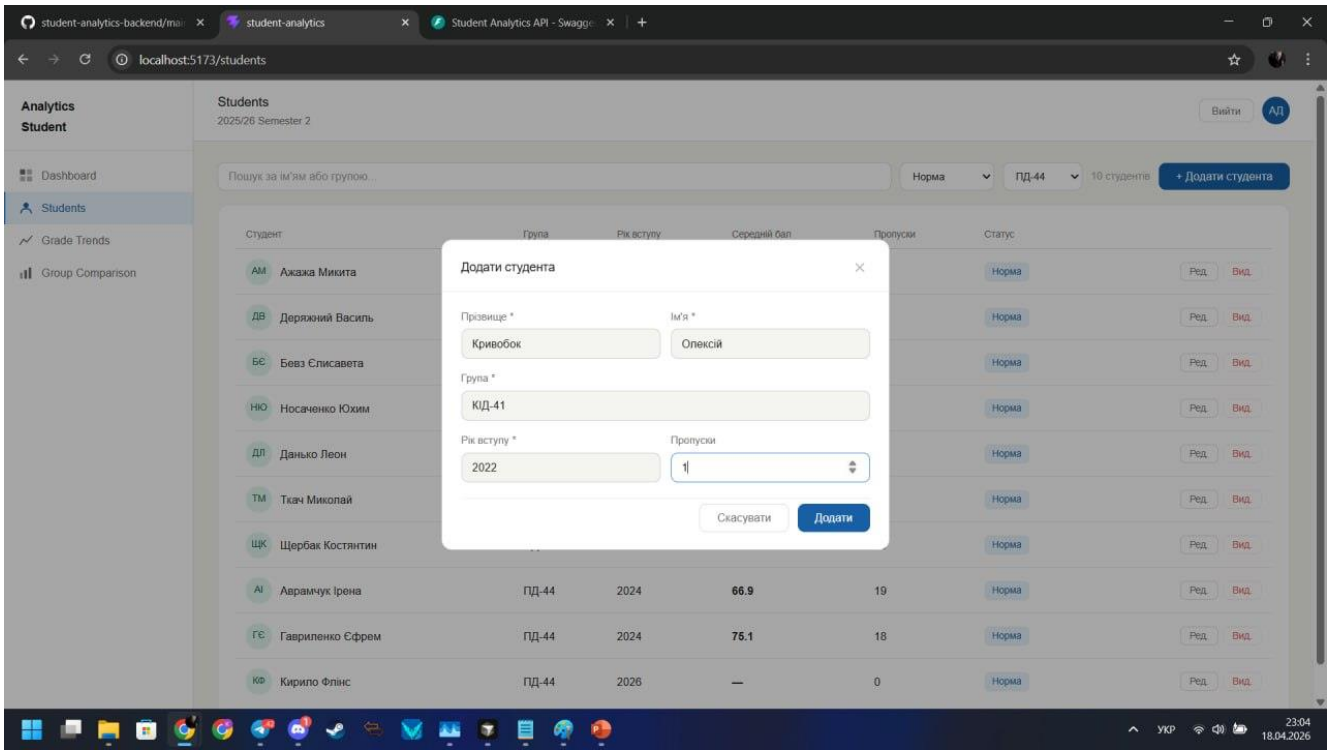



Рис. 4.7 Інтерфейс додавання нового студента

Перевірено коректність завантаження даних з сервера та їх візуалізацію у вигляді списків та діаграм (рис. 4.8-4.9).

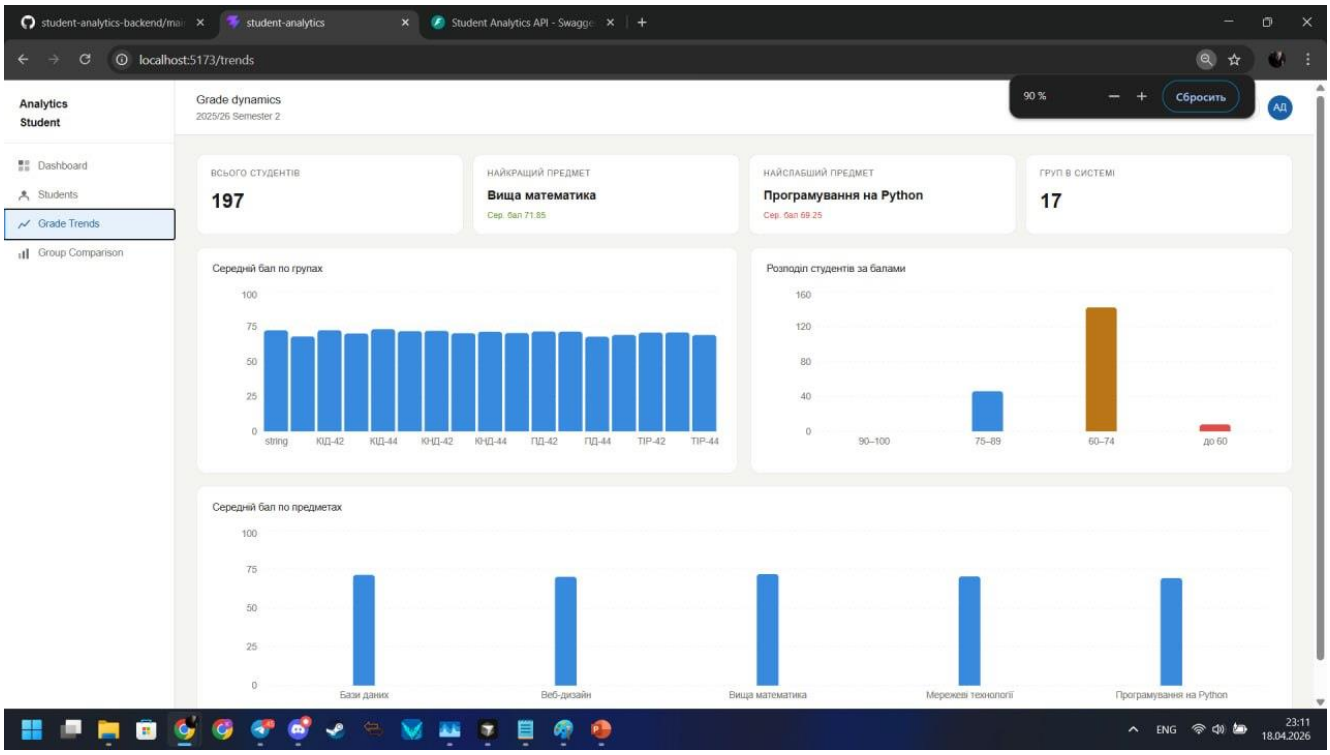



Рис. 4.8 Діаграми динаміки успішності за групами та предметами

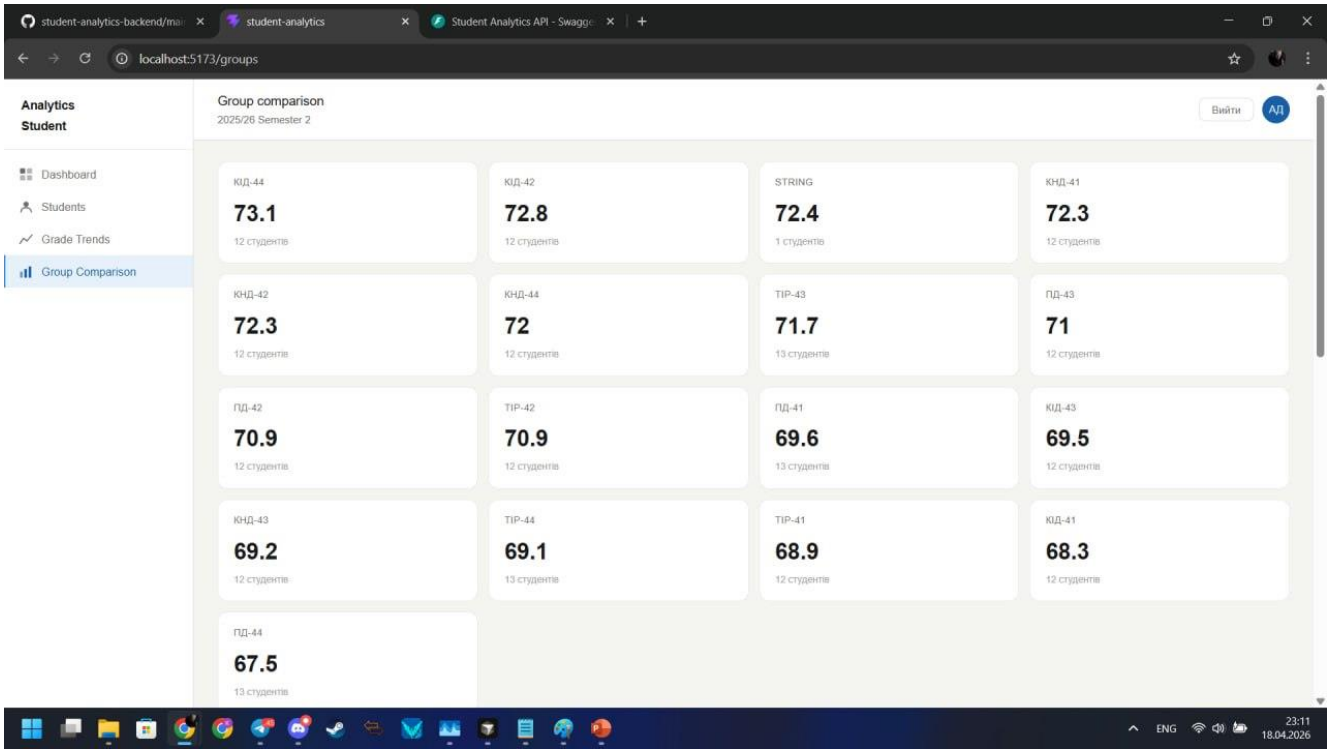



Рис. 4.9 Порівняння усіх наявних у системі груп за їх середніми балами

# ВИСНОВКИ

У процесі виконання кваліфікаційної роботи розроблено клієнтську (front-end) частину системи аналітики успішності студентів. Ця частина реалізує користувацький інтерфейс, побудований на базі бібліотеки React, модуль аналітики та візуалізації освітніх даних, та механізм безпеки – авторизацію адміністратора та відхилення доступу для неавторизованих користувачів.

1. Проведено аналіз предметної галузі систем моніторингу та аналітики успішності студентів. Визначено ключові проблеми звичних та подекуди застарілих методів обробки великих об’ємів даних, які є притаманними для вищих навчальних закладів.

2. Проведено порівняльний аналіз деяких зі вже існуючих програмних рішень. Для аналізу обрано системи Canvas LMS, Moodle та Blackboard Predict. Визначено, що розглянуті системи є здебільшого орієнтованими на конкретно ведення електронних журналів та організації навчальних процесів, без можливостей наочної аналітики наявних даних. Остання з вищенаведених систем фокусується на передбаченні майбутньої успішності студента за його попередніми даними, але ця система має значні недоліки, такі як виключно корпоративна основа, висока вартість, та відсутність пояснення знаходження того чи іншого студента у групі ризику.

3. На основі аналізу існуючих систем визначено функціональні та нефункціональні вимоги до розроблюваної системи. До таких вимог, з-поміж інших, належать: автентифікація адміністратора, повний цикл управління даними студентів, візуалізація сирих даних за допомогою різних діаграм та списків, визначення студентів у групі ризику та пояснення причини цього; безпека, продуктивність, навігація без перезавантаження сторінки, взаємодія з сервером через централізований сервісний шар.

4. Обґрунтовано вибір програмних засобів для розробки клієнтської частини. Було використано комплекс веб-технологій HTML, CSS, JavaScript у

фреймворках React, React Router та інструмент розгортання Vite. Написання програмної частини відбувалося у інтегрованому середовищі розробки Cursor, а хостинг клієнтської частини забезпечувався хмарною платформою Vercel з інтеграцією з Git-репозиторієм системи.

5. Розроблено діаграми архітектури та компонентів системи; діаграми послідовності роботи системи; мапу веб-застосунку; діаграму варіантів використання (прецедентів) системи. Архітектура системи представляє собою односторінковий SPA-застосунок.

6. Клієнтська частина реалізована із застосуванням принципів безпеки: неавторизовані користувачі мають доступ лише до сторінки входу в систему, а авторизований адміністратор має повний доступ до усієї системи та даних у ній. Модуль візуалізації освітніх даних реалізований за допомогою вбудованої у React бібліотеки Recharts. Цей модуль перетворює дані на наочні діаграми, графіки та списки для легшого засвоєння інформації. Це, нарівні з визначенням та поясненням групи ризику, є основною перевагою розробленої системи над вже існуючими.

7. Тестування функціональності системи проведено методом ручного функціонального тестування. Перевірено роботу авторизації та перенаправлення неавторизованих користувачів, URL-маршрутів, циклу управління студентами та їх даними, завантаження та відображення усіх наявних діаграм та списків. При відсутності чи нестачі будь-яких даних, або при виникненні помилок, відображаються відповідні повідомлення. Тестування підтвердило відповідність розробленої системи визначеним функціональним та нефункціональним вимогам.

Розроблена система може використовуватися у закладах вищої освіти для автоматизації аналізу освітніх даних студентів та виявлення груп ризику, що надає можливість завчасного вжиття необхідних заходів. Подальший розвиток та масштабування системи можуть включати в себе інтеграцію з електронними журналами, реалізацію доступу викладачів з обмеженими правами (лише перегляд оцінок студентів); додавання механізмів машинного навчання та/або нейронних мереж/штучного інтелекту для реалізації механізму передбачення майбутньої успішності студентів.

Кваліфікаційна робота пройшла апробацію, за результатами якої було опубліковано тези доповідей:

1. Кривобок О.А., Цапро І.В. Розробка користувацької частини веб-застосунку для аналітики успішності студентів. Матеріали ІІ Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025. С.150-152.

2. Кривобок О.А., Цапро І.В. Аналітика та візуалізація даних у клієнтській частині системи аналітики успішності студентів. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2026. С.84-85.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Сотніченко Ю. О. Роль цифрових платформ в підвищенні якості освіти фахівців з телекомунікації. Педагогічна Академія: наукові записки. 2024. № 9. URL: https://pedagogical-academy.com/index.php/journal/article/view/273/156

2. Поліщук Н. В., Бугаєнко Т. І., Лемешева Н. В. Підвищення якості вищої освіти за допомогою цифрових технологій та дистанційного навчання для здобувачів вищої освіти в Україні. Академічні візії. 2024. № 38. URL: https://www.academy-vision.org/index.php/av/article/view/1561/1444

3. Закрижевська І. В., Овод Л. В. Роль освіти у формуванні цифрової етики та підтримці академічної доброчесності. 2024. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/6a8d42ac-c8f9-4500-a10d-dc5f92638ac7/content

4. Геревенко А. М., Ільїна Т. В., Ібрагімова Л. А. Використання цифрових платформ для підвищення якості професійної освіти. Академічні візії. 2024. № 31. URL: https://academy-vision.org/index.php/av/article/view/1149/2536

5. Антоненко Г. М., Терменжи Д. Є., Різак Г. В. Дослідження ролі дистанційних технологій у забезпеченні стійкості української вищої освіти. Педагогічна Академія: наукові записки. 2025. № 19. URL: https://pedagogical-academy.com/index.php/journal/article/view/1059/937

6. Роль цифрових технологій навчання в епоху цивілізаційних змін / Р. Гуревич та ін. Modern Information Technologies and Innovation Methodologies of Education in Professional Training Methodology Theory Experience Problems. 2021. С. 28–38. URL: https://vspu.net/sit/index.php/sit/article/view/3742/3146

7. Родінова Н. Л. Використання цифрових платформ для формування медіаграмотності у здобувачів вищої освіти. Педагогічна академія: наукові записки. 2024. № 13. URL: https://pedagogical-academy.com/index.php/journal/article/view/454/334

8. Oliveira P. C., Cunha C. J. C. A., Nakayama M. K. Learning Management Systems (LMS) and e-learning management: an integrative review and research agenda.

JISTEM-Journal of Information Systems and Technology Management. 2016. Vol. 13. P. 157–180. URL: https://doi.org/10.4301/S1807-17752016000200001

9. Turnbull D., Chugh R., Luck J. Learning management systems, an overview. Encyclopedia of education and information technologies. 2020. P. 1052–1058. URL: https://link.springer.com/rwe/10.1007/978-3-030-10576-1_248

10. Mahnegar F. Learning management system. International Journal of Business and Social Science. 2012. Vol. 3, no. 12. URL: https://d1wqtxts1xzle7.cloudfront.net/90409043/14-libre.pdf

11. Rhode J. Understanding faculty use of the learning management system. Online Learning. 2017. Vol. 21, no. 3. P. 68. URL: https://doi.org/10.24059/olj.v21i3.1217

12. Kasim N. N. M., Khalid F. Choosing the right learning management system (LMS) for the higher education institution context: A systematic review. International Journal of Emerging Technologies in Learning. 2016. Vol. 11, no. 6. URL: https://d1wqtxts1xzle7.cloudfront.net/46940438/5644-18966-1-PB-libre.pdf

13. Berking P., Gallagher S. Choosing a learning management system. Advanced Distributed Learning (ADL) Co-Laboratories. 2013. Vol. 14, no. 4. P. 40–62. URL: https://www.adlnet.gov/assets/uploads/ChoosingAnLMS.pdf

14. Coates H., James R., Baldwin G. A critical examination of the effects of learning management systems on university teaching and learning. Tertiary education and management. 2005. Vol. 11, no. 1. P. 19–36. URL: https://link.springer.com/article/10.1007/s11233-004-3567-9

15. Клименко Є., Глазунова О. Архітектура інформаційної технології освітньої аналітики з використанням інтелектуального аналізу даних. Інформаційні технології та суспільство. 2025. № 2 (17). С. 69–75. URL: https://journals.maup.com.ua/index.php/it/article/view/4907/5231

16. Дужич Н. В., Мялюк О. П., Марущак М. І. Аналіз чинників, які впливають на академічну успішність. Медична освіта. 2023. № 3. С. 54–61. URL: https://ojs.tdmu.edu.ua/index.php/med_osvita/article/view/14050/13148

17. Géron A. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., 2022. 856 p. URL:

https://books.google.com.ua/books?id=X5ySEAAAQBAJ&lpg=PT17&ots=yC4wsf03AH&dq=Scikit-learn&lr&hl=uk&pg=PT17#v=onepage&q=Scikit-learn&f=false

18. Gavrilă V., Băjenaru L., Dobre C. Modern single page application architecture: a case study. Studies in Informatics and Control. 2019. Vol. 28, no. 2. P. 231–238. URL: https://sic.ici.ro/documents/239/Art.-11-Issue-2-SIC-2019.pdf

19. Ruoxuan W., Uehara M. A Multitenant Single-Page Application for Programming Education. 2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW). IEEE, 2023. DOI: https://doi.org/10.1109/CANDARW60564.2023.00038.

20. Aponte M. Create Your Single-Page Application. Building Single Page Applications in .NET Core 3: Jumpstart Coding Using Blazor and C#. Berkeley, CA: Apress, 2020. P. 33–71. URL: https://link.springer.com/chapter/10.1007/978-1-4842-5747-0_3

21. Majeed A., Rauf I. MVC architecture: a detailed insight to the modern web applications development. Peer Review Journal of Solar & Photoenergy Systems. 2018. Vol. 1, no. 1. P. 1–7. URL: https://d1wqtxts1xzle7.cloudfront.net/125256717/PRSP.000505-libre.pdf

22. Necula S. Exploring the model-view-controller (MVC) architecture: A broad analysis of market and technological applications. Preprints. 2024. URL: https://www.preprints.org/frontend/manuscript/00899780442be70e42e7cf39910ad427/download_pub

23. Mufid M. R. et al. Design an mvc model using python for flask framework development. 2019 International Electronics Symposium (IES). IEEE, 2019. URL: https://ieeexplore.ieee.org/abstract/document/8901656

24. Ahmad S. I., Rana T., Maqbool A. A model-driven framework for the development of MVC-based (Web) application. Arabian Journal for Science and Engineering. 2022. Vol. 47, no. 2. P. 1733–1747. DOI: https://doi.org/10.1007/s13369-021-06087-4.

25. Kukla M., Wieczorek B., Warguła Ł. Development of methods for performing the maximum voluntary contraction (MVC) test. MATEC Web of Conferences. 2018. Vol. 157. DOI: https://doi.org/10.1051/matecconf/201815705015.

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Система аналітики успішності студентів. Спец-частина: розробка Frontend-частини системи

Виконав студент 4 курсу
групи ПД-44
Кривобок Олексій Анатолійович
Керівник роботи
викладач кафедри ІПЗ Цапро Ігор Вікторович

Київ – 2026

## МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Мета роботи**: удосконалення процесу управління аналізом успішності у закладах вищої освіти за рахунок створення та впровадження комплексної системи для безпечного збереження, агрегації та візуалізації освітніх даних.

**Об’єкт дослідження:** процеси аналізу успішності та інтерактивного представлення аналітичних даних у веб-середовищі.

**Предмет дослідження:** методи, засоби та технології розробки систем аналітики освітніх даних у закладах вищої освіти.

# ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної галузі щодо питань та проблем у цифровізації вищої освіти та автоматизації створення аналітики результатів студентів.
2. Провести огляд та аналіз вже існуючих програм та систем для аналітики успішності студентів, визначити їх переваги та недоліки.
3. Сформувати функціональні та нефункціональні вимоги до розроблюваної системи.
4. Виконати огляд програмних засобів для розробки клієнтської (front-end) частини системи аналітики успішності студентів.
5. Спроєктувати архітектуру клієнтської частини системи.
6. Розробити front-end частину системи аналітики успішності студентів: інтерфейс, модуль аналітики та візуалізації даних.
7. Налаштувати зв’язок front-end частини з серверною (back-end) частиною.
8. Провести функціональне тестування розробленого програмного забезпечення.

3

# АНАЛІЗ АНАЛОГІВ

| Параметр | Canvas LMS | Moodle | Blackboard Predict | “SDF” |
|---|---|---|---|---|
| **Тип рішення** | Хмарна LMS | Гібридна LMS | Система передбачення успішності | Веб-система аналітики та візуалізації даних |
| **Складність впровадження** | Середня (час на інтеграцію та навчання) | Висока (повне самостійне налаштування) | Висока (попередня робота з вхідними даними) | Низька (Docker-контейнеризація) |
| **Гнучкість UI** | Висока (редактор тем) | Низька (застарілий інтерфейс) | Низька (лише налаштування фільтрів) | Спеціалізована (під конкретні потреби) |
| **Розширюваність** | Середня (зовнішні сервіси) | Висока (відкритий код) | Відсутня | Висока |
| **Підтримувані БД** | PostgreSQL | PostgreSQL, MySQL, Oracle | Інтегрована | MongoDB |

4

# ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

1. Автентифікація адміністратора за поштою та паролем.
2. Візуалізація результатів моніторингу на інтерактивних дашбордах.
3. Інтерактивна фільтрація даних.
4. Повний CRUD-цикл управління даними.

Нефункціональні:

1. Час первинного завантаження сторінки та побудови графіків <= 2 с.
2. Взаємодія з серверною частиною через централізований сервісний шар.
3. Кросбраузерність (Chrome, Opera, Microsoft Edge).
4. Навігація без перезавантаження сторінки шляхом використання концепції SPA.

5

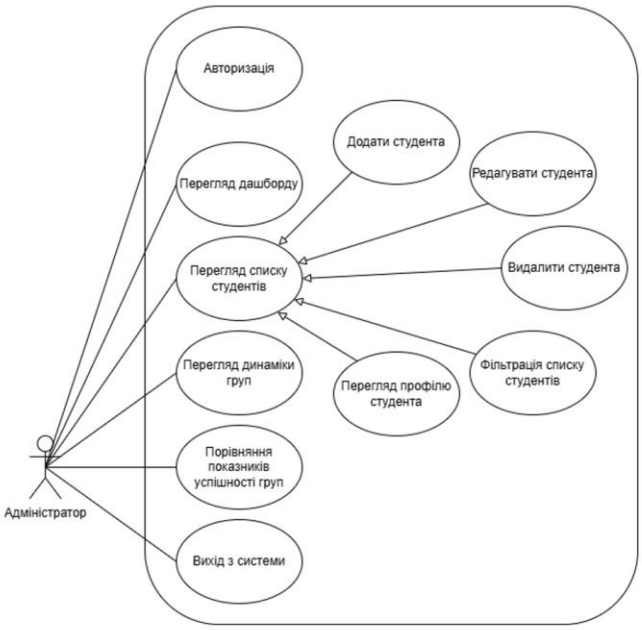
# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ





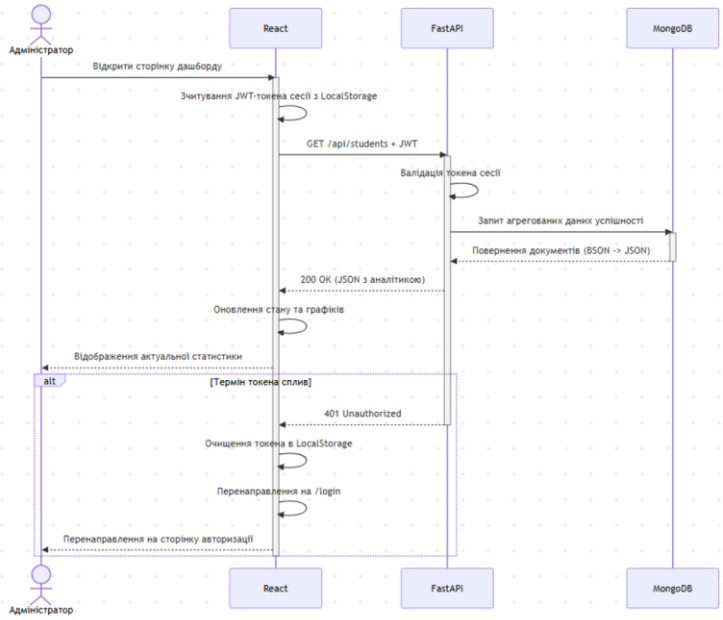

6

# ДІАГРАМА ПРЕЦЕДЕНТІВ



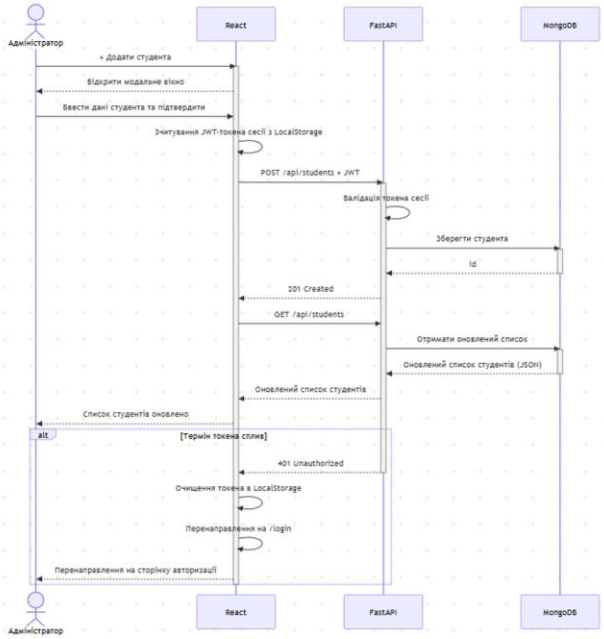

7

# ДІАГРАМА ПОСЛІДОВНОСТІ ЗАВАНТАЖЕННЯ ГОЛОВНОЇ СТОРІНКИ



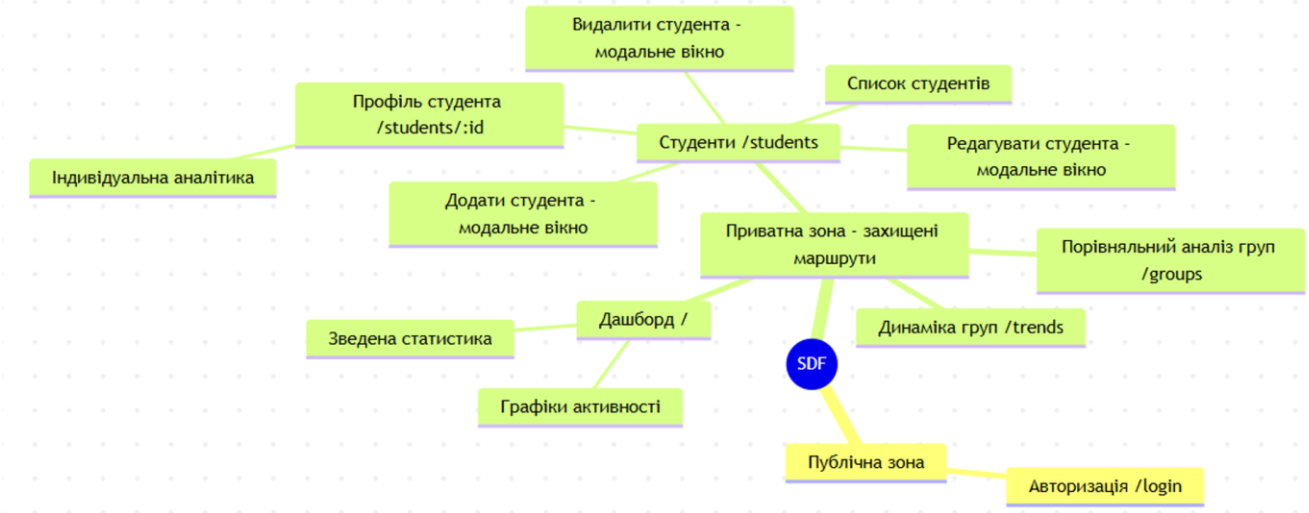

8

# ДІАГРАМА ПОСЛІДОВНОСТІ ДОДАВАННЯ СТУДЕНТА



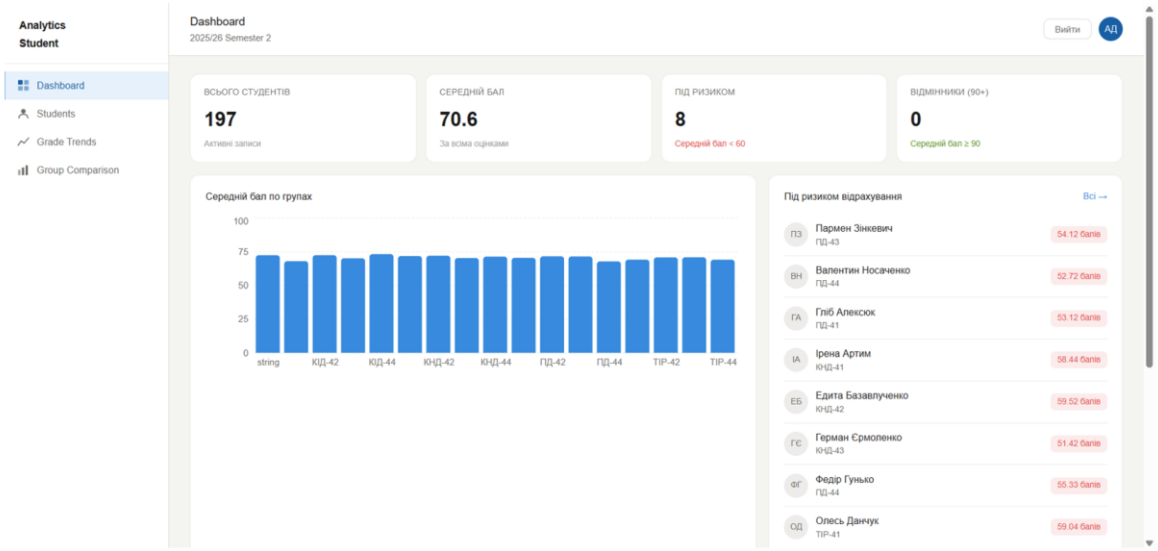

9

# МАПА СИСТЕМИ «SDF»




10

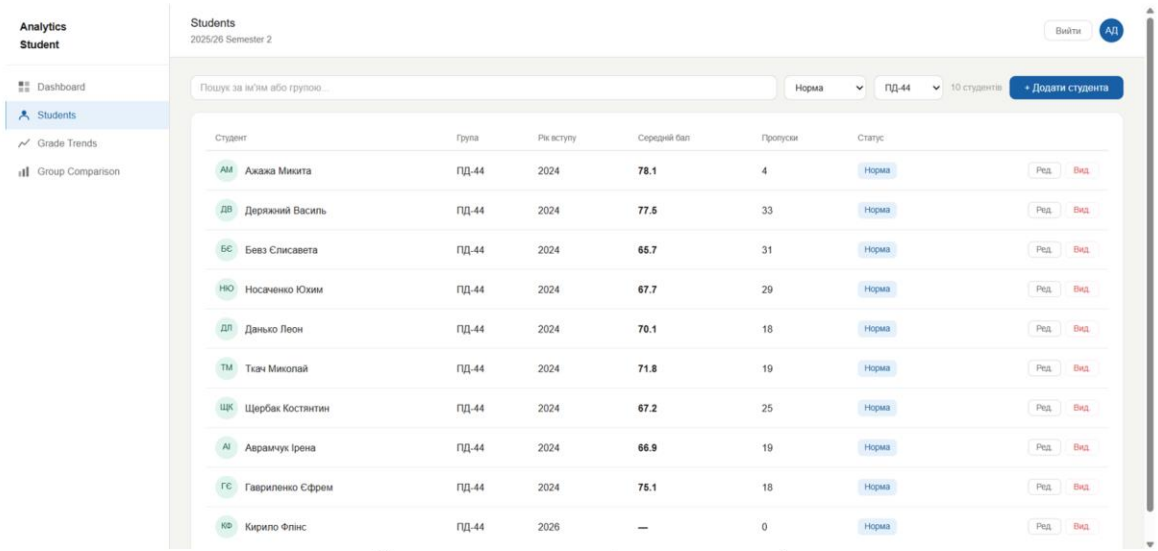
# ЕКРАННІ ФОРМИ




Головний дашборд системи

11

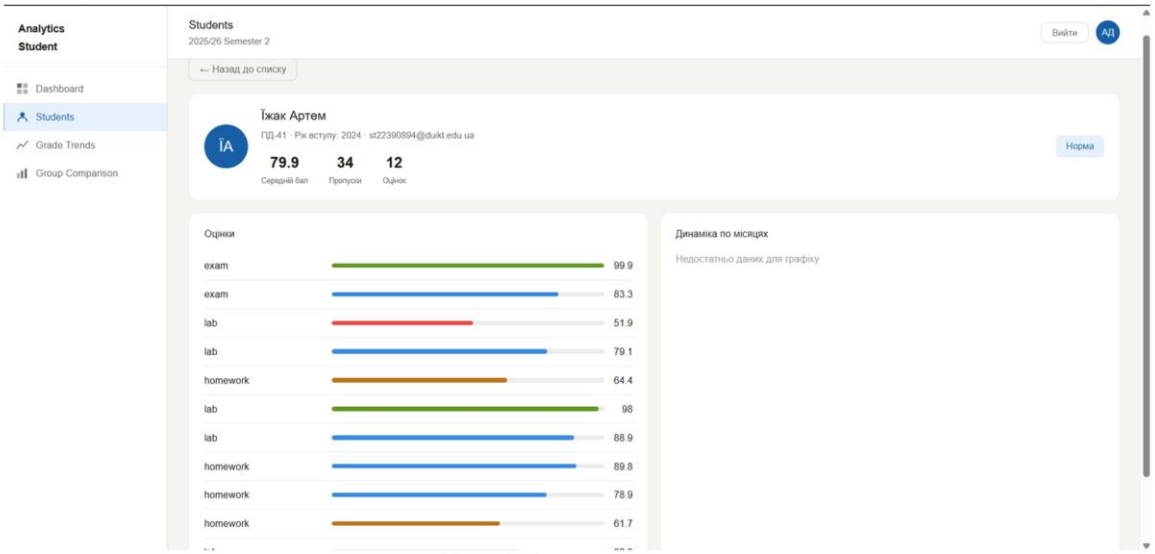
# ЕКРАННІ ФОРМИ




Список студентів у системі

12

# ЕКРАННІ ФОРМИ




Профіль студента



# АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Кривобок О.А., Цапро І.В. Розробка користувацької частини веб-застосунку для аналітики успішності студентів. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025. С.150-152.

2. Кривобок О.А., Цапро І.В. Аналітика та візуалізація даних у клієнтській частині системи аналітики успішності студентів. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2026. С.84-85.

# ВИСНОВКИ

1. Проведено аналіз предметної галузі щодо проблем цифровізації вищої освіти.
2. Проведено аналіз існуючих систем, що виконують функції аналітики успішності студентів. Визначено їх переваги та недоліки.
3. Визначено функціональні та нефункціональні вимоги до розроблюваної системи.
4. Проведено огляд програмних засобів реалізації front-end частини системи аналітики “SDF”, обрано веб-фреймворки React, React Router, Recharts.
5. Спроєктовано архітектуру системи та розроблено односторінковий веб-застосунок (SPA) з головним дашбордом, списком студентів із фільтрами, профілями окремих студентів, сторінками динаміки окремих груп та навчальних дисциплін.
6. Проведено функціональне тестування роботи клієнтської частини, підтверджено відповідність розробленого функціоналу визначеним вимогам.

# ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

**main.jsx**

```
import { StrictMode } from "react";
import { createRoot } from 'react-dom/client'
import { AuthProvider } from "./context/AuthContext";
import { StudentsProvider } from "./context/StudentsContext";
import App from './App.jsx'
import './index.css'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <AuthProvider>
      <StudentsProvider>
        <App />
      </StudentsProvider>
    </AuthProvider>
  </StrictMode>
)
```

**App.jsx**

```
import { BrowserRouter, Routes, Route } from "react-router-dom"
import Sidebar from './components/Sidebar'
import Topbar from './components/Topbar'
import ProtectedRoute from "./components/ProtectedRoute"
import Dashboard from "./pages/Dashboard"
import Students from "./pages/Students"
import StudentProfile from "./pages/StudentProfile"
import Trends from "./pages/Trends"
import Groups from "./pages/Groups"
import Login from "./pages/Login"
import './App.css'

function Layout({title, children}) {
  return (
    <div className="app-layout">
      <Sidebar />
      <div className="app-main">
        <Topbar
          title={title}
          subtitle="2025/26 Semester 2"
        />
        <main className="app-content">{children}</main>
      </div>
    </div>
  )
}

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/login" element={<Login />} />
        <Route path="/" element={
          <ProtectedRoute><Layout
title="Dashboard"><Dashboard
/></Layout></ProtectedRoute>
        } />
        <Route path="/students" element={
          <ProtectedRoute><Layout     title="Students"><Students
/></Layout></ProtectedRoute>
        } />
        <Route path="/students/:id" element={
          <ProtectedRoute><Layout
title="Students"><StudentProfile
/></Layout></ProtectedRoute>
        } />
        <Route path="/trends" element={
          <ProtectedRoute><Layout                title="Grade
dynamics"><Trends /></Layout></ProtectedRoute>
        } />
        <Route path="/groups" element={
          <ProtectedRoute><Layout                title="Group
comparison"><Groups /></Layout></ProtectedRoute>
        } />
      </Routes>
    </BrowserRouter>
  )
}
```

**api.js**

```
const BASE = 'https://student-analytics-backend.onrender.com'

function getToken() {
  return localStorage.getItem('auth_token')
}

function authHeaders() {
  return {
    'Authorization': `Bearer ${getToken()}`,
    'Content-Type': 'application/json',
  }
}

async function handleResponse(res) {
  if (!res.ok) {
    const err = await res.json().catch(() => ({}))
    throw new Error(err.detail || `HTTP ${res.status}`)
  }
  return res.json()
}

export async function login(email, password) {
  const body = new URLSearchParams({ username: email,
password })
  const res = await fetch(`${BASE}/api/auth/login`, {
    method: 'POST',
    headers:    {    'Content-Type':    'application/x-www-form-
urlencoded' },
    body,
  })
  return handleResponse(res)
}

export async function register(email, password) {
  const res = await fetch(`${BASE}/api/auth/register`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ email, password }),
  })
  return handleResponse(res)
}

export async function fetchStudents() {
  const res = await fetch(`${BASE}/api/students`)
  return handleResponse(res)
}

export async function createStudent(data) {
  const res = await fetch(`${BASE}/api/students`, {
    method: 'POST',
    headers: authHeaders(),
    body: JSON.stringify(data),
  })
  return handleResponse(res)
}
```

```
export async function updateStudent(id, data) {
  const res = await fetch(`${BASE}/api/students/${id}`, {
    method: 'PUT',
    headers: authHeaders(),
    body: JSON.stringify(data),
  })
  return handleResponse(res)
}

export async function deleteStudent(id) {
 const res = await fetch(`${BASE}/api/students/${id}`, {
  method: 'DELETE',
  headers: authHeaders(),
 })
 return handleResponse(res)
}

export async function fetchGrades() {
 const res = await fetch(`${BASE}/api/grades`)
 return handleResponse(res)
}

export async function addGrade(data) {
 const res = await fetch(`${BASE}/api/grades`, {
  method: 'POST',
  headers: authHeaders(),
  body: JSON.stringify(data),
 })
 return handleResponse(res)
}

export async function fetchSubjects() {
 const res = await fetch(`${BASE}/api/subjects`)
 return handleResponse(res)
}

export async function createSubject(data) {
 const res = await fetch(`${BASE}/api/subjects`, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(data),
 })
 return handleResponse(res)
}

export async function fetchDashboard() {
 const res = await fetch(`${BASE}/api/analytics/dashboard`)
 return handleResponse(res)
}

export async function fetchRiskGroup() {
 const res = await fetch(`${BASE}/api/analytics/risk-group`)
 return handleResponse(res)
}
```

**pages/Dashboard.jsx**

```
import { useEffect, useState } from 'react'
import {useNavigate} from 'react-router-dom'
import {BarChart, Bar, RadarChart, Radar, PolarGrid, PolarAngleAxis, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer} from 'recharts'
import MetricCard from '../components/MetricCard'
import { fetchDashboard, fetchRiskGroup } from '../api'
import { useStudents } from '../context/StudentsContext'
//import { dashboardStats, groups } from '../data/mockData'
import './Dashboard.css'

export default function Dashboard() {
  const navigate = useNavigate()
  const {students} = useStudents()
  const [analytics, setAnalytics] = useState(null)
  const [riskGroup, setRiskGroup] = useState(null)
  const [loading, setLoading] = useState(true)
  useEffect(() => {
    Promise.all([fetchDashboard(), fetchRiskGroup()])
      .then(([a, r]) => { setAnalytics(a); setRiskGroup(r) })
      .catch(() => {})
      .finally(() => setLoading(false))
  }, [])

  const totalStudents = students.length
  const gpas          = students.filter(s => s.gpa !== null).map(s => s.gpa)
  const averageGpa    = gpas.length ? (gpas.reduce((a, b) => a + b, 0) / gpas.length).toFixed(1) : '—'
  const atRiskCount   = students.filter(s => s.status === 'risk').length
  const excellentCount= students.filter(s => s.status === 'excellent').length

  const groupChartData = analytics?.avarege_score_by_group
    ? Object.entries(analytics.avarege_score_by_group).map(([name, gpa]) => ({ name, gpa }))
    : []

  const radarData = analytics?.average_score_by_subject
    ? Object.entries(analytics.average_score_by_subject).map(([subject, score]) => ({ subject, score }))
    : []

  const atRiskStudents = riskGroup?.students_at_risk || []

  if (loading) return <div className="page-loading">Завантаження аналітики...</div>

  return (
    <div className="dashboard">
      <div className="metrics-row">
        <MetricCard label="Всього студентів" value={totalStudents} change="Активні записи" changeType="neutral" />
        <MetricCard label="Середній бал" value={averageGpa} change="За всіма оцінками" changeType="neutral" />
        <MetricCard label="Під ризиком" value={atRiskCount} change="Середній бал < 60" changeType={atRiskCount > 0 ? 'down' : 'up'} />
        <MetricCard label="Відмінники (90+)" value={excellentCount} change="Середній бал ≥ 90" changeType="up" />
      </div>

      <div className="dashboard-grid">
        <div className="card">
          <div className="card-header">
            <span className="card-title">Середній бал по групах</span>
          </div>
          {groupChartData.length === 0
            ? <p style={{ color: 'var(--text-hint)', fontSize: 13 }}>Немає даних - додайте оцінки</p>
            : <ResponsiveContainer width="100%" height={220}>
                <BarChart data={groupChartData} barSize={32}>
                  <CartesianGrid strokeDasharray="3 3" stroke="rgba(0,0,0,0.06)" vertical={false} />
```

```
                <XAxis dataKey="name" tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <YAxis domain={[0, 100]} tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <Tooltip contentStyle={{ fontSize: 12, borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
                  formatter={v => [v, 'Середній бал']} />
                <Bar dataKey="gpa" fill="#378ADD" radius={[4, 4, 0, 0]} />
              </BarChart>
            </ResponsiveContainer>
          }
        </div>

        <div className="card">
          <div className="card-header">
            <span className="card-title">Під ризиком відрахування</span>
            <span className="card-action" onClick={() => navigate('/students')}>Всі →</span>
          </div>
          {atRiskStudents.length === 0
            ? <p style={{ color: 'var(--text-hint)', fontSize: 13 }}>Студентів під ризиком немає</p>
            : <div className="at-risk-list">
                {atRiskStudents.map(s => (
                  <div key={s.student_id} className="at-risk-item"
                    onClick={() => navigate(`/students/${s.student_id}`)}>
                    <div className="ari-avatar">{(s.full_name || '??').split(' ').map(w => w[0]).join('').toUpperCase()}</div>
                    <div className="ari-info">
                      <div className="ari-name">{s.full_name}</div>
                      <div className="ari-meta">{s.group_code}</div>
                    </div>
                    <span className="badge badge-danger">{s.score} балів</span>
                  </div>
                ))}
              </div>
          }
        </div>
      </div>

      {radarData.length > 0 && (
        <div className="card">
          <div className="card-header">
            <span className="card-title">Успішність по предметах (загальна)</span>
          </div>
          <ResponsiveContainer width="100%" height={220}>
            <RadarChart data={radarData}>
              <PolarGrid stroke="rgba(0,0,0,0.08)" />
              <PolarAngleAxis dataKey="subject" tick={{ fontSize: 12, fill: '#6B6B67' }} />
              <Radar dataKey="score" stroke="#378ADD" fill="#378ADD" fillOpacity={0.15} strokeWidth={2} />
              <Tooltip contentStyle={{ fontSize: 12, borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
                formatter={v => [v, 'Бал']} />
            </RadarChart>
          </ResponsiveContainer>
        </div>
      )}
    </div>
  )
}
```

**pages/Groups.jsx**

```
import {useEffect, useState} from 'react'
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from 'recharts'
//import MetricCard from '../components/MetricCard'
//import { groups } from '../data/mockData'
import { useStudents } from '../context/StudentsContext'
import { fetchDashboard, fetchRiskGroup } from '../api'
import './Groups.css'

const GROUP_COLORS = ['#378ADD', '#BA7517', '#639922', '#7F77DD', '#E24B4A']

const badgeClass = (n, type) => {
  if (type === 'risk')      return n >= 5 ? 'badge-danger' : 'badge-warning'
  if (type === 'excellent') return n >= 5 ? 'badge-success' : 'badge-info'
  return ''
}

export default function Groups() {
  const { students } = useStudents()
  const [analytics, setAnalytics] = useState(null)
  const [riskGroup, setRiskGroup] = useState(null)

  useEffect(() => {
    Promise.all([fetchDashboard(), fetchRiskGroup()])
      .then(([a, r]) => { setAnalytics(a); setRiskGroup(r) })
      .catch(() => {})
  }, [])
  const groupMap = {}
  students.forEach(s => {
    if (!groupMap[s.group_code]) groupMap[s.group_code] = { name: s.group_code, students: [] }
    groupMap[s.group_code].students.push(s)
  })

  const groupStats = Object.values(groupMap).map(g => {
    const withGpa  = g.students.filter(s => s.gpa !== null)
    const gpa      = withGpa.length ? withGpa.reduce((a, s) => a + s.gpa, 0) / withGpa.length : 0
    const risk     = g.students.filter(s => s.status === 'risk').length
    const excellent = g.students.filter(s => s.status === 'excellent').length
    const absences = g.students.reduce((a, s) => a + (s.absences || 0), 0) / g.students.length
    return { name: g.name, count: g.students.length, gpa: Math.round(gpa * 10) / 10, risk, excellent, absences: Math.round(absences * 10) / 10 }
  }).sort((a, b) => b.gpa - a.gpa)
  const subjectData = analytics?.average_score_by_subject
    ? Object.entries(analytics.average_score_by_subject).map(([subject, score]) => ({ subject, score }))
    : []

  const best  = groupStats[0]
  const worst = groupStats[groupStats.length - 1]

  return (
    <div className="groups-page">
      <div className="groups-metrics">
        {groupStats.map(g => (
          <div key={g.name} className="metric-card">
            <div className="metric-label">{g.name}</div>
            <div className="metric-value">{g.gpa || '—'}</div>
```

```
          <div className="metric-change neutral">{g.count} студентів</div>
        </div>
      ))}
    </div>

    <div className="groups-grid">
      <div className="card">
        <div className="card-header"><span className="card-title">Середній бал по предметах</span></div>
        {subjectData.length === 0
          ? <p style={{ color: 'var(--text-hint)', fontSize: 13 }}>Немає даних - додайте оцінки</p>
          : <ResponsiveContainer width="100%" height={260}>
              <BarChart data={subjectData} barSize={28}>
                <CartesianGrid strokeDasharray="3 3" stroke="rgba(0,0,0,0.06)" vertical={false} />
                <XAxis dataKey="subject" tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <YAxis domain={[0, 100]} tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <Tooltip contentStyle={{ fontSize: 12, borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
                  formatter={v => [v, 'Середній бал']} />
                <Bar dataKey="score" radius={[4, 4, 0, 0]}>
                  {subjectData.map((_, i) => (
                    <rect key={i} fill={GROUP_COLORS[i % GROUP_COLORS.length]} />
                  ))}
                </Bar>
              </BarChart>
            </ResponsiveContainer>
        }
      </div>

      <div className="card">
        <div className="card-header"><span className="card-title">Зведена таблиця груп</span></div>
        {groupStats.length === 0
          ? <p style={{ color: 'var(--text-hint)', fontSize: 13 }}>Немає студентів</p>
          : <>
              <table className="groups-table">
                <thead>
                  <tr><th>Група</th><th>Сер. бал</th><th>Ризик</th><th>Відм.</th><th>Пропуски</th></tr>
                </thead>
                <tbody>
                  {groupStats.map(g => (
                    <tr key={g.name}
                      className={g.name === best?.name ? 'row-best' : g.name === worst?.name && groupStats.length > 1 ? 'row-worst' : ''}>
                      <td><strong>{g.name}</strong></td>
                      <td>{g.gpa}</td>
                      <td><span className={`badge ${badgeClass(g.risk, 'risk')}`}>{g.risk}</span></td>
                      <td><span className={`badge ${badgeClass(g.excellent, 'excellent')}`}>{g.excellent}</span></td>
                      <td>{g.absences}</td>
                    </tr>
                  ))}
                </tbody>
              </table>
              {groupStats.length > 1 && (
                <div className="groups-table-legend">
                  <span className="legend-item"><span className="legend-dot" style={{ background: 'var(--green-light)', border: '1px solid var(--green)' }} />Найкраща</span>
                  <span className="legend-item"><span className="legend-dot" style={{ background: 'var(--red-light)', border: '1px solid var(--red)' }} />Найслабша</span>
                </div>
              )}
            </>
        }
      </div>
    </div>
  )
}
```

**pages/Login.jsx**

```
import { useState } from "react";
import {useNavigate} from 'react-router-dom'
import {useAuth} from '../context/AuthContext'
import './Login.css'

export default function Login() {
  const { login } = useAuth()
  const navigate = useNavigate()
  const [email, setEmail]       = useState('')
  const [password, setPassword] = useState('')
  const [error, setError]       = useState('')
  const [loading, setLoading]   = useState(false)

  async function handleSubmit() {
    if (!email || !password) { setError('Заповніть всі поля'); return }
    setLoading(true)
    setError('')
    try {
      await login(email, password)
      navigate('/')
    } catch (e) {
      setError(e.message || 'Невірний email або пароль')
    } finally {
      setLoading(false)
    }
  }

  return (
    <div className="login-page">
      <div className="login-card">
        <div className="login-logo">SDF</div>
        <div className="login-sub">Система аналітики успішності студентів</div>

        <div className="login-form">
          <div className="form-row">
            <label className="form-label">Email</label>
            <input
              className="form-input"
              type="email"
              placeholder="admin@university.ua"
              value={email}
              onChange={e => setEmail(e.target.value)}
              onKeyDown={e => e.key === 'Enter' && handleSubmit()}
            />
          </div>
          <div className="form-row">
            <label className="form-label">Пароль</label>
            <input
              className="form-input"
              type="password"
```

```
            placeholder="••••••••"
            value={password}
            onChange={e => setPassword(e.target.value)}
            onKeyDown={e => e.key === 'Enter' &&
handleSubmit()}
           />
          </div>
          {error && <div className="login-
error">{error}</div>}
          <button
           className="btn-primary login-btn"
           onClick={handleSubmit}
           disabled={loading}
          >
           {loading ? 'Вхід...' : 'Увійти'}
          </button>
         </div>
        </div>
      </div>
    )
}
```

**pages/StudentProfile.jsx**

```
import { useParams, useNavigate } from 'react-router-dom'
import { LineChart, Line, XAxis, YAxis, CartesianGrid,
Tooltip, ResponsiveContainer } from 'recharts'
//import { studentDetails } from '../data/mockData'
import { useStudents } from '../context/StudentsContext'
import './StudentProfile.css'

const statusLabel = {
 excellent: { text: 'Відмінник', cls: 'badge-success' },
 risk: { text: 'Ризик', cls: 'badge-danger' },
 weak: { text: 'Слабкий', cls: 'badge-warning' },
 normal: { text: 'Норма', cls: 'badge-info' },
}

export default function StudentProfile() {
  const { id } = useParams()
  const navigate = useNavigate()
  const { students, grades, loading } = useStudents()
  if (loading) return <div className="page-
loading">Loading...</div>
  const student = students.find(s => s.id === id)
  //const details = studentDetails[Number(id)]

  if (!student) return (
    <div className="profile-not-found">
     <p>Студента не знайдено.</p>
     <button onClick={() => navigate('/students')}>← Назад
до списку</button>
    </div>
  )

  const studentGrades = grades.filter(g => g.student_id === id)
  const byMonth = {}
  studentGrades.forEach(g => {
   const month = g.date.slice(0, 7)
   if (!byMonth[month]) byMonth[month] = []
   byMonth[month].push(g.score)
  })
  const trendData = Object.entries(byMonth)
   .sort(([a], [b]) => a.localeCompare(b))
   .map(([month, scores]) => ({
    month,
    gpa: Math.round((scores.reduce((s, v) => s + v, 0) /
scores.length) * 10) / 10,
   }))
   const lineColor = student.gpa >= 75 ? '#378ADD' :
student.gpa >= 60 ? '#BA7517' : '#E24B4A'
   const st = statusLabel[student.status] || statusLabel.normal

   return (
    <div className="profile-page">
     <button className="back-btn" onClick={() =>
navigate('/students')}>← Назад до списку</button>

     <div className="card profile-header-card">
      <div                                  className="profile-
avatar">{student.initials}</div>
      <div className="profile-info">
       <div                                  className="profile-
name">{student.name}</div>
       <div className="profile-meta">
        {student.group_code} · Рік вступу:
{student.enrollment_year} · {student.email}
       </div>
       <div className="profile-stats">
        <div className="profile-stat">
         <div  className="ps-value">{student.gpa  ??  '—
'}</div>
         <div className="ps-label">Середній бал</div>
        </div>
        <div className="profile-stat">
         <div                                  className="ps-
value">{student.absences}</div>
         <div className="ps-label">Пропуски</div>
        </div>
        <div className="profile-stat">
         <div                                  className="ps-
value">{studentGrades.length}</div>
         <div className="ps-label">Оцінок</div>
        </div>
       </div>
      </div>
      <span    className={`badge    ${st.cls}`}    style={{
fontSize: 12, padding: '6px 14px' }}>
       {st.text}
      </span>
     </div>

     <div className="profile-grid">
      <div className="card">
       <div                  className="card-header"><span
className="card-title">Оцінки</span></div>
       {studentGrades.length === 0
        ? <p style={{ color: 'var(--text-hint)', fontSize: 13
}}>Оцінок ще немає</p>
        : studentGrades.map((g, i) => {
          const barColor = g.score >= 90 ? 'var(--green)' :
g.score >= 75 ? 'var(--blue)' : g.score >= 60 ? 'var(--amber)' :
'var(--red)'
          return (
           <div key={i} className="subject-row">
            <span                                  className="subject-
name">{g.grade_type}</span>
            <div className="subject-bar-wrap">
             <div className="subject-bar" style={{ width:
`${g.score}%`, background: barColor }} />
            </div>
            <span                                  className="subject-
score">{g.score}</span>
           </div>
          )
         })
       }
      </div>

      <div className="card">
```

```
          <div                    className="card-header"><span
className="card-title">Динаміка по місяцях</span></div>
          {trendData.length < 2
           ? <p style={{ color: 'var(--text-hint)', fontSize: 13
}}>Недостатньо даних для графіку</p>
           :          <ResponsiveContainer          width="100%"
height={220}>
              <LineChart data={trendData}>
                <CartesianGrid       strokeDasharray="3       3"
stroke="rgba(0,0,0,0.06)" vertical={false} />
                <XAxis dataKey="month" tick={{ fontSize: 11,
fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <YAxis domain={[0, 100]} tick={{ fontSize: 12,
fill: '#6B6B67' }} axisLine={false} tickLine={false} />
                <Tooltip      contentStyle={{      fontSize:      12,
borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
                  formatter={v => [v, 'Сер. бал']} />
                <Line      dataKey="gpa"      stroke={lineColor}
strokeWidth={2}
                  dot={{ r: 4, fill: lineColor }} activeDot={{ r: 5
}} />
              </LineChart>
            </ResponsiveContainer>
          }
         </div>
        </div>
       </div>
     )
}
```

**pages/Students.jsx**

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { useStudents } from '../context/StudentsContext';
import Modal from '../components/Modal';
import StudentForm from '../components/StudentForm';
import './Students.css'

const statusLabel = {
   excellent: { text: 'Відмінник', cls: 'badge-success' },
   risk: { text: 'Ризик', cls: 'badge-danger' },
   weak: { text: 'Слабкий', cls: 'badge-warning' },
   normal: { text: 'Норма', cls: 'badge-info' },
 }

const avatarColors = {
   excellent: { bg: '#E6F1FB', color: '#0C447C' },
   risk: { bg: '#FCEBEB', color: '#A32D2D' },
   weak: { bg: '#FAEEDA', color: '#854F0B' },
   normal: { bg: '#E1F5EE', color: '#085041' },
}

export default function Students() {
 const navigate = useNavigate()
 const { students, loading, error, addStudent, editStudent,
removeStudent } = useStudents()

 const [search, setSearch] = useState('')
 const [statusFilter, setStatus] = useState('all')
 const [groupFilter, setGroup] = useState('all')
 const [showAdd, setShowAdd] = useState(false)
 const [editTarget, setEditTarget] = useState(null)
 const [deleteTarget, setDeleteTarget] = useState(null)
 const [credentials, setCredentials] = useState(null)
 const [saving, setSaving] = useState(false)
 const [saveError, setSaveError] = useState('')

 const      groups      =      [...new      Set(students.map(s      =>
s.group_code))].sort()

 const filtered = students.filter(s => {
  const                matchSearch                =
s.name.toLowerCase().includes(search.toLowerCase()) ||

s.group_code.toLowerCase().includes(search.toLowerCase())
  const matchStatus = statusFilter === 'all' || s.status ===
statusFilter
  const matchGroup  = groupFilter  === 'all' || s.group_code
=== groupFilter
  return matchSearch && matchStatus && matchGroup
 })

 async function handleAdd(data) {
  setSaving(true)
  setSaveError('')
  console.log('handleAdd called, data:', data, 'addStudent:',
typeof addStudent)
  try {
   const res = await addStudent(data)
   setShowAdd(false)
   if (res.credentials) setCredentials(res.credentials)
  } catch (e) {
   setSaveError(e.message)
  } finally {
   setSaving(false)
  }
 }

 async function handleEdit(data) {
   if (!editTarget) return
   setSaving(true)
   setSaveError('')
   try {
    await editStudent(editTarget.id, data)
    setEditTarget(null)
   } catch (e) {
    setSaveError(e.message)
   } finally {
    setSaving(false)
   }
 }

 async function handleDelete() {
  if (!deleteTarget) return
  setSaving(true)
  try {
   await removeStudent(deleteTarget.id)
   setDeleteTarget(null)
  } catch (e) {
   setSaveError(e.message)
  } finally {
   setSaving(false)
  }
 }

  if (loading) return <div className="page-loading">Loading
students...</div>
  if (error) return <div className="page-error">Error:
{error}</div>

  return (
   <div className="students-page">
    <div className="students-toolbar">
     <input className="search-input" type="text"
      placeholder="Пошук за ім'ям або групою..."
      value={search}              onChange={e              =>
setSearch(e.target.value)} />
     <select  className="toolbar-select"  value={statusFilter}
onChange={e => setStatus(e.target.value)}>
```

```
          <option value="all">Всі статуси</option>
          <option value="excellent">Відмінники</option>
          <option value="normal">Норма</option>
          <option value="weak">Слабкі</option>
          <option value="risk">Під ризиком</option>
        </select>
        <select className="toolbar-select" value={groupFilter}
onChange={e => setGroup(e.target.value)}>
          <option value="all">Всі групи</option>
          {groups.map(g => <option key={g}
value={g}>{g}</option>)}
        </select>
        <span className="students-count">{filtered.length}
студентів</span>
        <button className="btn-primary" style={{ marginLeft:
'auto' }} onClick={() => { setSaveError(''); setShowAdd(true)
}}>
          + Додати студента
        </button>
      </div>

      <div className="card">
        <table className="students-table">
          <thead>
            <tr>
              <th>Студент</th>
              <th>Група</th>
              <th>Рік вступу</th>
              <th>Середній бал</th>
              <th>Пропуски</th>
              <th>Статус</th>
              <th></th>
            </tr>
          </thead>
          <tbody>
            {filtered.length === 0
              ? <tr><td colSpan={7} className="table-
empty">Нічого не знайдено</td></tr>
              : filtered.map(s => {
                  const av = avatarColors[s.status] ||
avatarColors.normal
                  return (
                    <tr key={s.id} onClick={() =>
navigate(`/students/${s.id}`)}>
                      <td>
                        <div className="student-name-cell">
                          <div className="st-avatar" style={{
background: av.bg, color: av.color }}>{s.initials}</div>
                          {s.name}
                        </div>
                      </td>
                      <td>{s.group_code}</td>
                      <td>{s.enrollment_year}</td>
                      <td><strong>{s.gpa ?? '—'}</strong></td>
                      <td>{s.absences}</td>
                      <td><span className={`badge
${statusLabel[s.status].cls}`}>{statusLabel[s.status].text}</spa
n></td>
                      <td>
                        <div className="row-actions" onClick={e =>
e.stopPropagation()}>
                          <button className="row-btn" onClick={()
=> { setSaveError(''); setEditTarget(s) }}>Ред.</button>
                          <button className="row-btn row-btn-
danger" onClick={() => setDeleteTarget(s)}>Вид.</button>
                        </div>
                      </td>
                    </tr>
                  )
                })
            }
          </tbody>
        </table>
      </div>

      {showAdd && (
        <Modal title="Додати студента" onClose={() =>
setShowAdd(false)}>
          {saveError && <div className="login-error" style={{
marginBottom: 0 }}>{saveError}</div>}
          <StudentForm onSubmit={handleAdd} onCancel={()
=> setShowAdd(false)}
            submitLabel={saving ? 'Збереження...' : 'Додати'} />
        </Modal>
      )}

      {editTarget && (
        <Modal title="Редагувати студента" onClose={() =>
setEditTarget(null)}>
          {saveError && <div className="login-error" style={{
marginBottom: 0 }}>{saveError}</div>}
          <StudentForm initial={editTarget}
onSubmit={handleEdit} onCancel={() => setEditTarget(null)}
            submitLabel={saving ? 'Збереження...' : 'Зберегти
зміни'} />
        </Modal>
      )}

      {deleteTarget && (
        <Modal title="Видалити студента" onClose={() =>
setDeleteTarget(null)}>
          <p style={{ fontSize: 13, color: 'var(--text-secondary)',
lineHeight: 1.6 }}>
            Ви впевнені, що хочете видалити
<strong>{deleteTarget.name}</strong>? Цю дію неможливо
скасувати.
          </p>
          <div className="form-actions" style={{ marginTop: 8
}}>
            <button className="btn-secondary" onClick={() =>
setDeleteTarget(null)}>Скасувати</button>
            <button className="btn-danger"
onClick={handleDelete} disabled={saving}>
              {saving ? 'Видалення...' : 'Видалити'}
            </button>
          </div>
        </Modal>
      )}

      {credentials && (
        <Modal title="Студента створено - доступи"
onClose={() => setCredentials(null)}>
          <p style={{ fontSize: 13, color: 'var(--text-secondary)',
lineHeight: 1.6 }}>
            Збережіть ці дані - пароль більше не буде показано.
          </p>
          <div className="credentials-box">
            <div className="cred-
row"><span>Email</span><strong>{credentials.email}</stron
g></div>
            <div className="cred-
row"><span>Пароль</span><strong>{credentials.password}<
/strong></div>
          </div>
          <div className="form-actions" style={{ marginTop: 8
}}>
            <button className="btn-primary" onClick={() =>
setCredentials(null)}>Зрозуміло</button>
```

```
        </div>
      </Modal>
    )}
  </div>
 )
}
```

**pages/Trends.jsx**

```
import {useEffect, useState} from 'react'
import { LineChart, Line, BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer, Cell } from 'recharts'
//import MetricCard from '../components/MetricCard'
//import { semesterTrend, subjectComparison, scoreDistribution } from '../data/mockData'
import { fetchDashboard } from '../api'
import { useStudents } from '../context/StudentsContext'
import './Trends.css'

const distColors = ['var(--green)', 'var(--blue)', 'var(--amber)', 'var(--red)']

export default function Trends() {
  const { students } = useStudents()
  const [analytics, setAnalytics] = useState(null)

  useEffect(() => {
   fetchDashboard().then(setAnalytics).catch(() => {})
  }, [])

  const dist = [
   { range: '90–100', count: students.filter(s => s.gpa >= 90).length },
   { range: '75–89', count: students.filter(s => s.gpa >= 75 && s.gpa < 90).length },
   { range: '60–74', count: students.filter(s => s.gpa >= 60 && s.gpa < 75).length },
   { range: 'до 60', count: students.filter(s => s.gpa !== null && s.gpa < 60).length },
  ]

  const subjectData = analytics?.average_score_by_subject
   ? Object.entries(analytics.average_score_by_subject).map(([subject, score]) => ({ subject, score }))
   : []

  const groupData = analytics?.avarege_score_by_group
   ? Object.entries(analytics.avarege_score_by_group).map(([name, gpa]) => ({ name, gpa }))
   : []

  const bestSubject = subjectData.length ? subjectData.reduce((a, b) => a.score > b.score ? a : b) : null
  const worstSubject = subjectData.length ? subjectData.reduce((a, b) => a.score < b.score ? a : b) : null

  return (
   <div className="trends-page">
    <div className="trends-metrics">
     <div className="metric-card">
      <div className="metric-label">Всього студентів</div>
      <div className="metric-value">{students.length}</div>
     </div>
     <div className="metric-card">
      <div className="metric-label">Найкращий предмет</div>
      <div className="metric-value" style={{ fontSize: 16 }}>{bestSubject?.subject ?? '—'}</div>
      {bestSubject && <div className="metric-change up">Сер. бал {bestSubject.score}</div>}
     </div>
     <div className="metric-card">
      <div className="metric-label">Найслабший предмет</div>
      <div className="metric-value" style={{ fontSize: 16 }}>{worstSubject?.subject ?? '—'}</div>
      {worstSubject && <div className="metric-change down">Сер. бал {worstSubject.score}</div>}
     </div>
     <div className="metric-card">
      <div className="metric-label">Груп в системі</div>
      <div className="metric-value">{groupData.length}</div>
     </div>
    </div>

    <div className="trends-grid-top">
     <div className="card">
      <div className="card-header"><span className="card-title">Середній бал по групах</span></div>
      {groupData.length === 0
       ? <p style={{ color: 'var(--text-hint)', fontSize: 13 }}>Немає даних</p>
       : <ResponsiveContainer width="100%" height={220}>
          <BarChart data={groupData} barSize={36}>
           <CartesianGrid strokeDasharray="3 3" stroke="rgba(0,0,0,0.06)" vertical={false} />
           <XAxis dataKey="name" tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
           <YAxis domain={[0, 100]} tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
           <Tooltip contentStyle={{ fontSize: 12, borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
            formatter={v => [v, 'Середній бал']} />
           <Bar dataKey="gpa" fill="#378ADD" radius={[4, 4, 0, 0]} />
          </BarChart>
         </ResponsiveContainer>
       }
     </div>

     <div className="card">
      <div className="card-header"><span className="card-title">Розподіл студентів за балами</span></div>
      <ResponsiveContainer width="100%" height={220}>
       <BarChart data={dist} barSize={40}>
        <CartesianGrid strokeDasharray="3 3" stroke="rgba(0,0,0,0.06)" vertical={false} />
        <XAxis dataKey="range" tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
        <YAxis tick={{ fontSize: 12, fill: '#6B6B67' }} axisLine={false} tickLine={false} />
        <Tooltip contentStyle={{ fontSize: 12, borderRadius: 8, border: '0.5px solid rgba(0,0,0,0.1)' }}
         formatter={v => [v, 'Студентів']} />
        <Bar dataKey="count" radius={[4, 4, 0, 0]}>
         {dist.map((_, i) => <Cell key={i} fill={distColors[i]} />)}
        </Bar>
       </BarChart>
      </ResponsiveContainer>
     </div>
    </div>
```

```
      {subjectData.length > 0 && (
        <div className="card">
          <div                    className="card-header"><span
className="card-title">Середній          бал          по
предметах</span></div>
          <ResponsiveContainer width="100%" height={240}>
            <BarChart data={subjectData} barSize={28}>
              <CartesianGrid          strokeDasharray="3          3"
stroke="rgba(0,0,0,0.06)" vertical={false} />
              <XAxis dataKey="subject" tick={{ fontSize: 12, fill:
'#6B6B67' }} axisLine={false} tickLine={false} />
              <YAxis domain={[0, 100]} tick={{ fontSize: 12, fill:
'#6B6B67' }} axisLine={false} tickLine={false} />
              <Tooltip contentStyle={{ fontSize: 12, borderRadius:
8, border: '0.5px solid rgba(0,0,0,0.1)' }}
                formatter={v => [v, 'Середній бал']} />
              <Bar  dataKey="score"  fill="#378ADD"  radius={[4,
4, 0, 0]} />
            </BarChart>
          </ResponsiveContainer>
        </div>
      )}
    </div>
  )
}
```

**data/mockData.js**

```
export const students = [
  { id: 1, name: 'Шевченко Олена',   initials: 'ШО', group: 'КН-
31', gpa: 88, absences: 2,  status: 'excellent' },
  { id: 2, name: 'Даниленко Михайло', initials: 'ДМ', group:
'КН-31', gpa: 52, absences: 14, status: 'risk' },
  { id: 3, name: 'Іваненко Андрій',   initials: 'ІА', group: 'КН-
32', gpa: 76, absences: 3,  status: 'normal' },
  { id: 4, name: 'Мороз Наталія',     initials: 'МН', group: 'МТ-
21', gpa: 91, absences: 1,  status: 'excellent' },
  { id: 5, name: 'Кравченко Аліна',   initials: 'КА', group: 'МТ-
21', gpa: 58, absences: 9,  status: 'weak' },
  { id: 6, name: 'Бондаренко Василь', initials: 'БВ', group: 'КН-
31', gpa: 70, absences: 5,  status: 'normal' },
  { id: 7, name: 'Петренко Сергій',   initials: 'ПС', group: 'КН-
32', gpa: 55, absences: 11, status: 'risk' },
  { id: 8, name: 'Левченко Вікторія', initials: 'ЛВ', group: 'КН-
31', gpa: 61, absences: 7,  status: 'weak' },
  { id: 9, name: 'Ткаченко Роман',    initials: 'ТР', group: 'МТ-
21', gpa: 83, absences: 2,  status: 'normal' },
  { id: 10, name: 'Коваль Юлія',      initials: 'КЮ', group: 'КН-
32', gpa: 94, absences: 0,  status: 'excellent' },
]

export const studentDetails = {
  1: {
    fullName: 'Шевченко Олена Василівна',
    group: 'КН-31',
    specialty: "Комп'ютерні науки",
    year: 3,
    studentId: '2022-0147',
    rank: 'Топ 12%',
    subjects: [
      { name: 'Алгоритми і структури', score: 92 },
      { name: 'Бази даних',           score: 88 },
      { name: 'Вища математика',      score: 79 },
      { name: 'Операційні системи',   score: 91 },
      { name: "Комп'ютерні мережі",   score: 85 },
      { name: 'Англійська мова',      score: 95 },
    ],
    semesterGpa: [81, 84, 87, 88, 88],
  },
  2: {
    fullName: 'Даниленко Михайло Олегович',
    group: 'КН-31',
    specialty: "Комп'ютерні науки",
    year: 3,
    studentId: '2022-0134',
    rank: 'Низ 15%',
    subjects: [
      { name: 'Алгоритми і структури', score: 48 },
      { name: 'Бази даних',           score: 55 },
      { name: 'Вища математика',      score: 51 },
      { name: 'Операційні системи',   score: 60 },
      { name: "Комп'ютерні мережі",   score: 49 },
      { name: 'Англійська мова',      score: 52 },
    ],
    semesterGpa: [61, 58, 55, 54, 52],
  },
}

export const groups = [
  { name: 'КН-31', gpa: 78.3, count: 32, risk: 4, excellent: 7,
absences: 3.1 },
  { name: 'КН-32', gpa: 71.1, count: 29, risk: 6, excellent: 3,
absences: 5.4 },
  { name: 'МТ-21', gpa: 74.4, count: 28, risk: 3, excellent: 5,
absences: 4.0 },
  { name: 'МТ-22', gpa: 69.8, count: 31, risk: 5, excellent: 2,
absences: 6.2 },
]

export const subjectComparison = [
  { subject: 'Алгоритми', current: 74, previous: 71 },
  { subject: 'БД',        current: 79, previous: 76 },
  { subject: 'Математика',current: 68, previous: 65 },
  { subject: 'Мережі',    current: 75, previous: 73 },
  { subject: 'ОС',        current: 70, previous: 68 },
]

export const semesterTrend = [
  { semester: 'Сем 1', gpa: 70 },
  { semester: 'Сем 2', gpa: 72 },
  { semester: 'Сем 3', gpa: 71 },
  { semester: 'Сем 4', gpa: 74 },
  { semester: 'Сем 5', gpa: 74 },
]

export const scoreDistribution = [
  { range: '90–100', count: 19 },
  { range: '75–89',  count: 48 },
  { range: '60–74',  count: 37 },
  { range: 'до 60',  count: 23 },
]

export const dashboardStats = {
  totalStudents: 127,
  averageGpa: 74.2,
  atRisk: 11,
  excellent: 19,
}
```

**context/AuthContext.jsx**

```
import { createContext, useContext, useState } from "react";
import {login as apiLogin} from '../api'

const AuthContext = createContext(null)
export function AuthProvider({ children }) {
  const      [token,      setToken]      =      useState(()      =>
localStorage.getItem('auth_token'))

  async function login(email, password) {
    const data = await apiLogin(email, password)
```

```
    const t = data.access_token
    localStorage.setItem('auth_token', t)
    setToken(t)
    return data
  }

  function logout() {
    localStorage.removeItem('auth_token')
    setToken(null)
  }

  const isAuthenticated = !!token

  return (
    <AuthContext.Provider value={{ token, login, logout,
isAuthenticated }}>
      {children}
    </AuthContext.Provider>
  )
}

export function useAuth() {
  return useContext(AuthContext)
}
```

**context/StudentsContext.jsx**

```
import { createContext, useContext, useState, useEffect,
useCallback } from "react";
//import {students as initialStudents} from '../data/mockData'
import * as api from '../api'

const StudentsContext = createContext(null)

export function StudentsProvider({ children }) {
 const [students, setStudents] = useState([])
 const [grades, setGrades]     = useState([])
 const [loading, setLoading]   = useState(true)
 const [error, setError]       = useState(null)

 const load = useCallback(async () => {
  setLoading(true)
  setError(null)
  try {
    const [s, g] = await Promise.all([api.fetchStudents(),
api.fetchGrades()])
    setStudents(s)
    setGrades(g)
  } catch (e) {
    setError(e.message)
  } finally {
    setLoading(false)
  }
 }, [])

 useEffect(() => { load() }, [load])

 function getStudentGpa(studentId) {
  const sg = grades.filter(g => g.student_id === studentId)
  if (!sg.length) return null
  const avg = sg.reduce((sum, g) => sum + g.score, 0) /
sg.length
  return Math.round(avg * 10) / 10
 }

 function getStatus(gpa) {
  if (gpa === null) return 'normal'
  if (gpa >= 90) return 'excellent'
  if (gpa >= 65) return 'normal'
  if (gpa >= 60) return 'weak'
  return 'risk'
 }

 const enriched = students.map(s => {
  const gpa = getStudentGpa(s.id)
  const name = `${s.last_name} ${s.first_name}`
  const initials = (s.last_name[0] +
s.first_name[0]).toUpperCase()
  return {
    ...s,
    name,
    initials,
    gpa,
    status: getStatus(gpa),
  }
 })

async function addStudent(data) {
  const res = await api.createStudent(data)
  await load()
  return res
}

async function editStudent(id, data) {
  await api.updateStudent(id, data)
  await load()
}

async function removeStudent(id) {
  await api.deleteStudent(id)
  await load()
}

return (
  <StudentsContext.Provider value={{
    students: enriched,
    grades,
    loading,
    error,
    reload: load,
    addStudent,
    editStudent,
    removeStudent,
  }}>
    {children}
  </StudentsContext.Provider>
 )
}

export function useStudents() {
 return useContext(StudentsContext)
}
```

**components/ProtectedRoute.jsx**

```
import { Navigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
export default function ProtectedRoute({children}) {
  const { isAuthenticated } = useAuth()
  return isAuthenticated ? children : <Navigate to="/login"
replace />
}
```

**components/StudentForm.jsx**

```
import { useState } from "react";
import './StudentForm.css'
const emptyForm = {
  first_name: '',
  last_name: '',
  group_code: '',
  enrollment_year: new Date().getFullYear(),
  absences: '',
}
```

```
export default function StudentForm({ initial, onSubmit, onCancel, submitLabel }) {
  const [form, setForm] = useState(initial || emptyForm)
  const [errors, setErrors] = useState({})

  function set(field, value) {
    setForm(prev => ({ ...prev, [field]: value}))
    setErrors(prev => ({ ...prev, [field]: undefined }))
  }

  function validate() {
    const e = {}
    if (!form.first_name.trim()) e.first_name = "Введіть ім'я"
    if (!form.last_name.trim()) e.last_name = "Введіть прізвище"
    if (!form.group_code.trim()) e.group_code = "Введіть групу"
    const year = Number(form.enrollment_year)
    if (!year || year < 2000 || year > 2100) e.enrollment_year = "Введіть рік"
    const abs = Number(form.absences)
    if (isNaN(abs) || abs < 0) e.absences = "Невід\'ємне число"
    return e
  }

  function handleSubmit() {
    const e = validate()
    if (Object.keys(e).length) { setErrors(e); return }
    onSubmit({
      first_name: form.first_name.trim(),
      last_name: form.last_name.trim(),
      group_code: form.group_code.trim(),
      enrollment_year: Number(form.enrollment_year),
      absences: Number(form.absences),
    })
  }

  return (
    <>
      <div className="form-2col">
        <div className="form-row">
          <label className="form-label">Прізвище *</label>
          <input  className={`form-input ${errors.last_name ? 'input-error' : ''}`}
            value={form.last_name}  onChange={e => set('last_name', e.target.value)}
            placeholder="Шевченко" />
          {errors.last_name && <span className="form-error">{errors.last_name}</span>}
        </div>
        <div className="form-row">
          <label className="form-label">Ім'я *</label>
          <input  className={`form-input ${errors.first_name ? 'input-error' : ''}`}
            value={form.first_name}  onChange={e => set('first_name', e.target.value)}
            placeholder="Олена" />
          {errors.first_name && <span className="form-error">{errors.first_name}</span>}
        </div>
      </div>

      <div className="form-row">
        <label className="form-label">Група *</label>
        <input  className={`form-input ${errors.group_code ? 'input-error' : ''}`}
          value={form.group_code}  onChange={e => set('group_code', e.target.value)}
          placeholder="напр. КН-31" />
        {errors.group_code && <span className="form-error">{errors.group_code}</span>}
      </div>

      <div className="form-2col">
        <div className="form-row">
          <label className="form-label">Рік вступу *</label>
          <input  className={`form-input ${errors.enrollment_year ? 'input-error' : ''}`}
            type="number" min="2000" max="2100"
            value={form.enrollment_year}  onChange={e => set('enrollment_year', e.target.value)}
            placeholder="2022" />
          {errors.enrollment_year && <span className="form-error">{errors.enrollment_year}</span>}
        </div>
        <div className="form-row">
          <label className="form-label">Пропуски</label>
          <input  className={`form-input ${errors.absences ? 'input-error' : ''}`}
            type="number" min="0"
            value={form.absences}  onChange={e => set('absences', e.target.value)}
            placeholder="0" />
          {errors.absences && <span className="form-error">{errors.absences}</span>}
        </div>
      </div>

      <div className="form-actions">
        <button  className="btn-secondary" onClick={onCancel}>Скасувати</button>
        <button  className="btn-primary" onClick={handleSubmit}>{submitLabel || 'Зберегти'}</button>
      </div>
    </>
  )
}
```