Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

першого (балакаврського) рівня освіти
(назва освітнього ступеня)

На тему: Розробка та впровадження програмного забезпечення для моніторингу та аналітики навчальних курсів

Виконав: студент IV курсу, групи СПз-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Пилипчук В. В.
(прізвище та ініціали)

Керівник (підпис) Пастух О. А.
(прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р
(прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня першого (бакалаврського) рівня освіти
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Пилипчуку Володимиру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та впровадження програмного забезпечення для моніторингу та аналітики навчальних курсів

Керівник роботи проф. Пастух Олег Анатолійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*6*» *04* 2026 року № *4/9-173*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Технічна документація, теоретичні дані

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз існуючих підходів до реалізації програмного забезпечення для моніторингу та аналітики, розробка та впровадження програмного забезпечення для моніторингу та аналітики навчальних курсів, прикладні аспекти застосування програмного забезпечення для моніторингу та аналітики навчальних курсів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 6 квітня 2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Розробка технічного завдання	11.05 – 12.05	*Виконано*
2	Підбір джерел	13.05 – 14.05	*Виконано*
3	Опрацювання літературних джерел	15.05 – 17.05	*Виконано*
4	Виконання дослідження	18.05 – 20.05	*Виконано*
5	Написання програмного коду	21.05 – 22.05	*Виконано*
6	Оформлення розділу 1. Аналіз існуючих підходів до реалізації програмного забезпечення для моніторингу та аналітики	23.05 – 24.05	*Виконано*
7	Оформлення розділу 2. Розробка та впровадження програмного забезпечення для моніторингу та аналітики навчальних курсів	25.05 – 26.05	*Виконано*
8	Оформлення розділу 3. Прикладне використання програмного забезпечення для моніторингу та аналітики навчальних курсів	27.05 – 28.05	*Виконано*
9	Виконання завдання до підрозділу «БЖД, основи охорони праці»	26.05 – 30.05	*Виконано*
10	Оформлення пояснювальної записки	01.06 – 02.06	*Виконано*
11	Перевірка на академічний плагіат, перевірка керівником та консультантами	02.06 – 06.06	*Виконано*
12	Попередній захист кваліфікаційної роботи бакалавра	09.06 – 13.06	*Виконано*
13	Захист кваліфікаційної роботи	18.06	

Студент _____ Пилипчук В. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пастух О. А.

АНОТАЦІЯ

У кваліфікаційній роботі розроблено програмне забезпечення для моніторингу та збору аналітичних даних для університету. Окрім збору аналітики, цілю програмного забезпечення є систематизація та обробка інформації щодо якості освітнього процесу у зрозумілому відображенні даних на графіках. Запроповане рішення дозволяє покращити систему оцінювання ефективності робочого процесу та виявляти проблемні аспекти у ньому.

Програмна частина реалізована з врахуванням архітектури клієнт-сервер та сучасному стеку технологій: React та Node.js, які окремо реалізують frontend та backend частини застосунку, надалі поєднуючи їх та комунікуючи один з одним. Система написана згідно правил написання чистого та безпечного коду, що підтверджено тестуванням та QA-роботою після розробки.

Ключові слова: моніторинг, освітня аналітики, візуалізація даних, аналіз ефективності навчання.

ABSTRACT

This thesis involves the development of software for monitoring and collecting analytical data for a university. In addition to gathering analytics, the software is designed to organise and process information regarding the quality of the educational process, presenting the data in clear graphical form. The proposed solution enables the improvement of the system for evaluating the effectiveness of the workflow and the identification of problematic aspects within it.

The software component has been implemented using a client-server architecture and a modern technology stack: React and Node.js, which implement the frontend and backend parts of the application separately, subsequently combining them and enabling them to communicate with one another. The system is written in accordance with the principles of clean and secure coding, as confirmed by testing and QA work following development.

Keywords: monitoring, educational analytics, data visualisation, analysis of learning effectiveness.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

JWT – JSON Web Token

REST API – Representational State Transfer Application Programming Interface

CRUD – Create, Read, Update, Delete

UI – User Interface

UX – User Experience

PDF – Portable Document Format

JSON – JavaScript Object Notation

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

Node.js – Node JavaScript Runtime Environment

Express.js – Express JavaScript Framework

React.js – React JavaScript Library

Chart.js – JavaScript Charting Library

bcrypt – Password Hashing Function

CORS – Cross-Origin Resource Sharing

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

URL – Uniform Resource Locator

LMS – Learning Management System

Git – Distributed Version Control System

GitHub – Web-based Git Repository Hosting Service

npm – Node Package Manager

IDE – Integrated Development Environment

SPA – Single Page Application

RBAC – Role-Based Access Control

KPI – Key Performance Indicator

DevOps – Development and Operations

JWT Auth – JSON Web Token Authentication

Audit Log – System Activity Change History

ВСТУП

У сучасних умовах розвитку системи вищої освіти особливої важливості набуває постійний контроль якості навчальних курсів та окремих освітніх компонентів. Ефективність освітнього процесу значною мірою залежить від своєчасного виявлення проблемних аспектів у змісті дисциплін, методах викладання та рівні задоволеності здобувачів освіти. Одним із найбільш цінних джерел інформації для вдосконалення навчального процесу виступають відгуки студентів, випускників та інших учасників освітнього середовища.

Сучасні університети активно впроваджують внутрішні механізми контролю якості освіти, що передбачають регулярне проведення опитувань, анкетувань та аналізу результатів навчання. Однак традиційні методи збору й обробки такої інформації, зокрема паперові анкети або ручне опрацювання результатів, характеризуються низькою ефективністю, значними витратами часу та високою ймовірністю втрати важливих аналітичних даних. Це ускладнює прийняття своєчасних управлінських рішень та знижує загальну якість освітнього моніторингу.

Актуальність обраної теми посилюється зростанням ролі дистанційного та змішаного навчання, що особливо стало помітним в умовах воєнного стану та цифрової трансформації освітнього середовища. За таких обставин університети потребують більш гнучких, надійних та автоматизованих інструментів для оцінювання ефективності навчальних курсів, контролю академічних результатів та формування об’єктивної аналітичної картини освітнього процесу.

Саме тому розробка програмного забезпечення для моніторингу та аналітики навчальних курсів є актуальним і практично значущим завданням. Використання автоматизованої інформаційної системи дозволяє значно прискорити процес збору даних, спростити їх обробку, підвищити точність оцінювання та забезпечити наочне представлення результатів у вигляді звітів, статистичних показників і графічної візуалізації.

Запропонований підхід передбачає комплексний аналіз ефективності навчальних курсів на основі багатофакторної оцінки, що враховує академічну успішність студентів, рівень їхньої залученості до навчального процесу, задоволеність змістом дисциплін, якість викладання та відповідність навчальних компонентів сучасним вимогам ринку праці. Така система дозволяє виявляти сильні та слабкі сторони освітніх програм, визначати напрями їх удосконалення та формувати обґрунтовані управлінські рішення.

Метою дипломного проєкту є підвищення ефективності контролю, оперативності аналізу та достовірності оцінювання навчальних курсів шляхом розробки програмного забезпечення для автоматизованого моніторингу освітнього процесу. Практична реалізація роботи спрямована на створення системи, яка забезпечить швидке отримання аналітичної інформації щодо якості освітніх програм та окремих навчальних дисциплін. Об’єктом дипломного проєкту є процес моніторингу, оцінювання та аналітичного супроводу навчальних курсів і освітніх компонентів у закладах вищої освіти. Предметом дипломного проєкту є методи автоматизованого аналізу ефективності навчальних курсів, інструменти збору та обробки освітніх даних, а також способи представлення результатів аналітики для підтримки управлінських рішень в університетському середовищі.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІТИКИ

У першому розділі було досліджено існуючі підходи та програмні рішення для моніторингу й аналітики навчальних курсів. Було розглянуто такі системи, як Moodle, EvaP та Qualtrics XM, визначено їхні переваги та недоліки. На основі проведеного аналізу зроблено висновок, що доцільно розробити власне програмне забезпечення, яке буде адаптоване під потреби університету.

1.1 Аналіз існуючих рішень

Для розробки ефективного програмного забезпечення для моніторингу та аналітики навчальних курсів необхідно насамперед проаналізувати сучасні технологічні рішення, які вже застосовуються у сфері контролю якості освіти. У світовій практиці одним із найбільш поширених інструментів оцінювання освітнього процесу є систематичне опитування студентів щодо якості викладання, змісту дисциплін та загальної ефективності навчальних курсів. Такі механізми відомі як Student Course Evaluations [1, 3] і широко використовуються в університетах Європи, США та інших країн як складова внутрішньої системи забезпечення якості освіти.

Основною метою подібних опитувань є отримання об’єктивного зворотного зв’язку від здобувачів освіти, які безпосередньо взаємодіють із навчальним процесом. Саме студенти можуть найточніше оцінити практичну корисність дисциплін, актуальність навчальних матеріалів, якість викладання та відповідність освітньої програми сучасним професійним вимогам. Отримана інформація дозволяє адміністрації університету оперативно реагувати на проблемні аспекти та приймати обґрунтовані управлінські рішення.

У більшості випадків такі опитування будуються на основі шкали Лікерта, де респонденту пропонується оцінити певне твердження за шкалою від повної незгоди до повної згоди. Наприклад, студент може оцінювати твердження типу: «Навчальний матеріал подавався зрозуміло», «Дисципліна відповідає сучасним вимогам ринку праці», «Викладач забезпечував достатній рівень зворотного зв’язку», «Структура курсу була логічною та послідовною». Додатково часто включаються загальні питання щодо рівня задоволеності курсом, якості викладання та відкриті поля для текстових коментарів, які дозволяють отримати більш глибоку якісну оцінку.

Результати таких анкетувань використовуються не лише для оцінки окремих дисциплін чи викладачів, а й для комплексного аналізу ефективності навчальних курсів загалом. На основі цих даних можуть прийматися рішення щодо перегляду навчальних планів, оновлення змісту курсів, підвищення кваліфікації викладацького складу, зміни методик викладання та вдосконалення освітньої стратегії університету. Саме тому студентські оцінки сьогодні вважаються одним із ключових інструментів внутрішнього аудиту якості освіти.

З розвитком цифрових технологій процес збору, зберігання та аналізу таких даних поступово перейшов у цифровий формат. Сучасні університети дедалі частіше відмовляються від паперових анкет на користь електронних платформ, які дозволяють автоматизувати весь цикл роботи з освітньою аналітикою. Це значно знижує адміністративне навантаження, мінімізує людський фактор та підвищує достовірність результатів.

Сьогодні існує велика кількість спеціалізованих програмних продуктів, призначених для організації освітнього моніторингу. Частина закладів вищої освіти використовує вбудовані можливості систем управління навчанням, таких як Moodle, де модулі опитування інтегруються безпосередньо в навчальне середовище. Інші університети впроваджують окремі професійні рішення, серед яких особливо поширеними є EvaP та Qualtrics XM, [10].

Рисунок 1.1 – EvaP (Evaluation Platform)

Платформа EvaP є прикладом сучасного відкритого програмного рішення для організації процесу оцінювання навчальних курсів. Її головною перевагою є поєднання зручного інтерфейсу, адаптивного дизайну та можливості повної автоматизації анкетування. Система дозволяє формувати опитувальники, збирати відповіді студентів, здійснювати аналітичну обробку результатів та формувати зрозумілі звіти для адміністрації. Завдяки відкритому вихідному коду такі системи можуть адаптуватися під потреби конкретного університету, що є важливою перевагою для закладів освіти.

Рисунок 1.2 – Qualtrics XM

Іншим прикладом є Qualtrics XM — потужна комерційна хмарна платформа, яка працює за моделлю SaaS (Software as a Service). Вона орієнтована на професійне управління досвідом користувачів (Experience Management) і використовується не

лише в освіті, а й у сфері бізнес-аналітики, маркетингових досліджень, HR-процесів та UX-досліджень. Платформа надає розширені можливості побудови складних анкет, глибокої статистичної аналітики, інтеграції з корпоративними системами та автоматичного формування звітності, проте потребує фінансових витрат на ліцензування та підтримку.

Аналіз існуючих технологічних рішень показує, що найбільш важливими характеристиками сучасних систем освітньої аналітики є веб-орієнтована архітектура, можливість дистанційного доступу для всіх учасників освітнього процесу, підтримка анонімності респондентів, автоматичне обчислення показників ефективності та зручне представлення результатів у вигляді графіків, діаграм і статистичних звітів.

Саме тому для реалізації власного програмного забезпечення доцільно використовувати клієнт-серверну архітектуру, у якій фронтенд відповідає за взаємодію з користувачем через веб-інтерфейс, а серверна частина забезпечує логіку обробки даних, аналітичні розрахунки та збереження інформації у базі даних. Такий підхід дозволяє забезпечити масштабованість системи, її стабільність, безпеку та можливість подальшої інтеграції в існуючу цифрову інфраструктуру університету як окремого модуля або складової загальної LMS-системи.

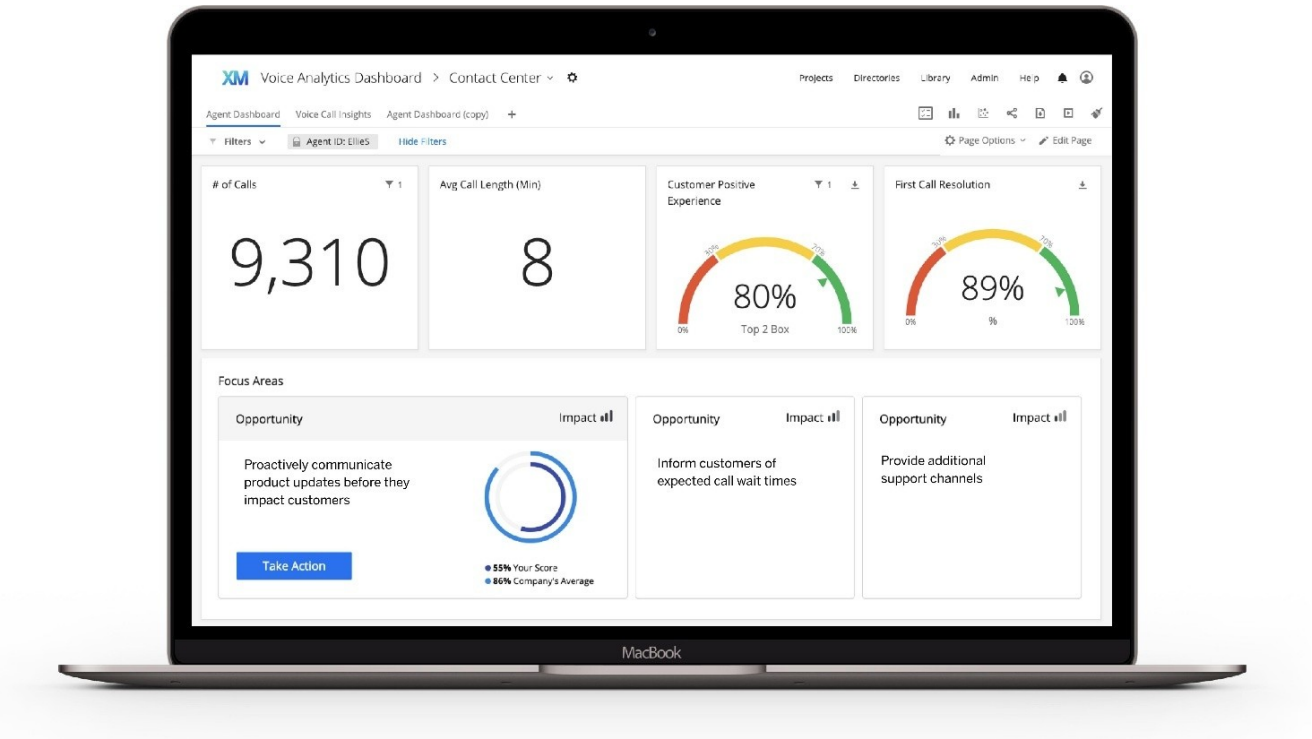


Рисунок 1.3 – Qualtrics XM, зображення забезпечення

1.2 Аналіз існуючих методологій

Багатокритеріальний аналіз є одним із найбільш ефективних підходів для оцінювання складних систем, у яких загальний результат формується під впливом великої кількості взаємопов’язаних факторів. У сфері вищої освіти така необхідність особливо актуальна, оскільки якість навчальної програми неможливо визначити лише за одним показником. Ефективність освітнього процесу залежить від багатьох параметрів: академічної успішності студентів, рівня їхньої задоволеності навчанням, якості викладання, актуальності змісту дисциплін, доступності навчальних матеріалів, організації практичної підготовки та рівня зворотного зв’язку між викладачем і студентом, [2, 4].

Саме тому для побудови сучасної системи моніторингу навчальних курсів доцільно використовувати багатофакторний підхід, який дозволяє одночасно враховувати декілька критеріїв та формувати більш об’єктивну аналітичну оцінку. Такий метод дає змогу не лише визначити загальний рівень ефективності освітньої

програми, а й виявити конкретні проблемні зони, що потребують коригування або вдосконалення.

У спрощеному вигляді багатокритеріальна оцінка може бути представлена як сума окремих показників, де кожен критерій має власну вагу залежно від його значущості. Наприклад, академічна успішність може мати більший вплив на підсумковий результат, ніж окремі організаційні аспекти курсу, тоді як показники задоволеності студентів дозволяють врахувати якісну складову освітнього процесу. Такий підхід забезпечує більш гнучке та реалістичне оцінювання порівняно з простими середніми значеннями.

Важливим доповненням до числового аналізу є робота з неструктурованими даними, зокрема з відкритими текстовими коментарями студентів. Саме вони часто містять найбільш цінну інформацію про реальні проблеми навчального процесу, які складно виявити за допомогою лише формалізованих анкет. У текстових відгуках студенти можуть детально описувати труднощі, позитивні враження, недоліки організації занять або проблеми взаємодії з викладачами, що робить такі дані надзвичайно важливими для глибокої аналітики.

Одним із перспективних напрямів обробки таких даних є тематичне моделювання, зокрема метод LDA (Latent Dirichlet Allocation), [5]. Його основне призначення полягає в автоматичному групуванні великих масивів тексту за змістовими темами. Це дозволяє системі самостійно визначати ключові напрями студентського фідбеку без необхідності ручного сортування коментарів.

Наприклад, за допомогою LDA можна виділити такі типові теми, як «організація розкладу», «якість навчальних матеріалів», «практична підготовка», «об’єктивність оцінювання», «зворотний зв’язок від викладача», «доступ до електронних ресурсів» та інші. Такий підхід значно спрощує аналіз великої кількості відгуків та дозволяє швидко визначити, які саме аспекти навчального процесу викликають найбільшу увагу студентів.

Після визначення основних тематичних груп доцільним стає застосування аналізу тональності (Sentiment Analysis), який дозволяє оцінити емоційне забарвлення текстових повідомлень. Основна задача цього підходу полягає у

класифікації відгуків на позитивні, нейтральні або негативні, а також у встановленні того, до якого саме аспекту навчального процесу належить відповідна оцінка.

Наприклад, якщо система виявляє, що тема «організація практичних занять» переважно супроводжується негативними коментарями, це може свідчити про необхідність перегляду методики проведення лабораторних робіт або змін у структурі практичної підготовки. Якщо ж позитивні оцінки переважають у темі «якість навчальних матеріалів», це може підтверджувати ефективність уже впроваджених рішень.

Таким чином, поєднання LDA та Sentiment Analysis дозволяє проводити багатовимірний аналіз: не лише визначати загальні теми студентського фідбеку, а й оцінювати емоційне ставлення до кожної з них окремо. Це значно підвищує точність управлінських рішень та дозволяє здійснювати адресне вдосконалення освітніх програм.

Сучасні методи машинного навчання значно підвищують якість такої аналітики. Особливу роль у цьому відіграють трансформерні моделі, зокрема BERT та RoBERTa, які дозволяють не лише точніше класифікувати текстові дані, а й виконувати автоматичне узагальнення великої кількості студентських коментарів.

Завдяки таким моделям система може формувати короткі аналітичні резюме на основі сотень або навіть тисяч відгуків, що значно спрощує роботу адміністрації університету. Замість ручного перегляду великого масиву текстів керівництво отримує структуровану інформацію про основні проблеми, позитивні тенденції та потенційні ризики в організації навчального процесу.

У межах даного дипломного проєкту основна увага зосереджується насамперед на кількісному аналізі числових показників, отриманих у результаті студентських опитувань, що дозволяє формувати об’єктивну оцінку ефективності навчальних курсів. Водночас перспективним напрямом подальшого розвитку системи є інтеграція модулів інтелектуального аналізу текстових відгуків, що значно розширить аналітичні можливості програмного забезпечення.

Таким чином, поєднання багатокритеріального аналізу структурованих даних із сучасними методами обробки неструктурованої текстової інформації формує комплексний підхід до моніторингу навчальних курсів. Така система має не лише прикладне значення для оцінювання якості освіти, а й стратегічну цінність для довгострокового розвитку закладу вищої освіти в умовах цифрової трансформації.

1.3. Визначення задач програмного забезпечення

На основі проведеного аналізу сучасних технологічних і методологічних підходів у сфері контролю якості освіти було встановлено доцільність створення комплексного програмного забезпечення для моніторингу та аналітики навчальних курсів. Сучасні заклади вищої освіти потребують не лише інструментів збору зворотного зв’язку від студентів, а й повноцінної інформаційної системи, яка дозволяє автоматизувати процес накопичення, обробки, аналізу та візуалізації освітніх даних для подальшого прийняття управлінських рішень.

В умовах цифровізації освітнього середовища та постійного оновлення вимог до якості підготовки фахівців особливо важливим стає формалізований підхід до побудови таких систем. Це передбачає чітке визначення функціональних можливостей програмного забезпечення, вимог до його продуктивності, безпеки, масштабованості, а також розробку математичних моделей аналізу освітніх показників і побудову раціональної архітектури системи.

Проєктована система повинна забезпечувати повний цикл роботи з освітньою аналітикою — від збору первинної інформації до формування підсумкових аналітичних звітів. Однією з ключових функцій є створення структурованих опитувальників, які дозволяють оцінювати різні аспекти навчального процесу. До таких категорій можуть належати якість викладання, актуальність змісту дисциплін, рівень практичної підготовки, інтерактивність занять, складність навчального матеріалу, доступність навчальних ресурсів, ефективність комунікації

з викладачем та загальна оцінка курсу. Важливістю системи є динамічне формування анкет. Такий підхід значно підвищує гнучкість програмного забезпечення та дозволяє адаптувати систему до специфіки різних факультетів, кафедр або освітніх програм. Анкети можуть містити як кількісні питання зі шкалою оцінювання (наприклад, від 1 до 5), так і текстові поля для розгорнутих коментарів студентів.

Наступним важливим компонентом є організація анонімного проходження опитування через веб-інтерфейс із можливістю ідентифікації навчальної дисципліни, семестру, освітньої програми та інших метаданих. Анонімність є критично важливою умовою отримання об’єктивного зворотного зв’язку, оскільки саме вона сприяє більш відкритому та чесному висловленню думок студентів щодо якості навчального процесу.

Для забезпечення ефективного управління системою необхідно реалізувати багаторівневу модель доступу. Усі отримані відповіді повинні зберігатися у базі даних та бути доступними на огляд лише для адміністратора, щоб забезпечити безпеку тих, хто проходить опитування та прозорості. Студенти повинні мати можливість проходити опитування у зручному та зрозумілому інтерфейсі без складних процедур реєстрації, за аналогією з роботою електронних освітніх платформ типу Moodle або внутрішніх університетських кампусів. Викладачі отримують доступ до перегляду узагальнених результатів без персоніфікації респондентів, а адміністративний персонал — до створення навчальних курсів, налаштування опитувань, управління структурою системи та генерації офіційної звітності.

Окрему увагу необхідно приділити побудові модуля візуалізації аналітичних результатів. Графічне представлення даних значно спрощує сприйняття інформації та дозволяє швидко виявляти проблемні тенденції. Для цього доцільно реалізувати аналітичну панель (dashboard), яка відображатиме результати у форматі графіків, гістограм, кругових діаграм, порівняльних таблиць та інтерактивних звітів.

На основі вищезазначених вимог можна сформулювати основні часткові задачі проєктування системи, вирішення яких забезпечить створення повнофункціонального програмного продукту.

Першим етапом є дослідження методів аналітичного оцінювання навчальних курсів та визначення ключових критеріїв, за якими здійснюється моніторинг ефективності освітнього процесу. Необхідно обґрунтувати значущість кожного показника та встановити їх вплив на загальний рівень якості навчання.

Другим етапом є розробка структури анкетування з урахуванням сучасних UX-принципів, що забезпечують простоту, зрозумілість та швидкість проходження опитування. Оптимальна тривалість заповнення анкети не повинна перевищувати 5–10 хвилин, що сприятиме підвищенню активності студентів та якості отриманих відповідей.

Третім важливим завданням є побудова логічної моделі бази даних, які повинні забезпечувати збереження результатів опитувань, інформації про дисципліни, семестри, категорії запитань та взаємозв’язків між усіма сутностями системи.

Наступним етапом є реалізація математичних алгоритмів для обробки даних, що включають статистичні розрахунки, багатокритеріальне оцінювання, кореляційний аналіз та побудову аналітичних прогнозів.

Також необхідно реалізувати систему авторизації та розмежування прав доступу, що дозволить забезпечити безпечну роботу з даними та чіткий розподіл функціональних можливостей між різними категоріями користувачів.

Завершальним етапом є створення сучасного інтерфейсу аналітики та візуалізації результатів, який забезпечить зручне представлення звітності для адміністрації університету та сприятиме прийняттю ефективних управлінських рішень.

Виконання зазначених завдань дозволить створити сучасне, масштабоване та практично корисне програмне забезпечення, яке відповідатиме актуальним вимогам цифрової трансформації вищої освіти та сприятиме підвищенню якості навчального процесу в університетському середовищі.

1.4. Висновки до розділу 1

Отже, у першому розділі було проведено комплексний аналіз предметної області та сформовано теоретичне підґрунтя для подальшого проєктування програмного забезпечення моніторингу та аналітики навчальних курсів. Дослідження існуючих рішень показало, що сучасні системи оцінювання якості освіти орієнтуються на автоматизацію збору студентського зворотного зв’язку, підтримку анонімності респондентів, зручне представлення результатів та інтеграцію аналітичних інструментів у єдине цифрове середовище. Аналіз платформ EvaP, Qualtrics XM та LMS-рішень підтвердив доцільність використання веб-орієнтованої клієнт-серверної архітектури як найбільш ефективного підходу для реалізації подібних систем.

У межах аналізу методологічних підходів було встановлено, що найбільш обґрунтованим для оцінювання ефективності навчальних курсів є багатокритеріальний аналіз, який дозволяє враховувати як кількісні показники, так і якісні характеристики освітнього процесу. Додатково розглянуто перспективи використання сучасних методів інтелектуального аналізу текстових відгуків студентів, зокрема тематичного моделювання LDA, аналізу тональності та трансформерних моделей типу BERT і RoBERTa. Це дозволяє розширити можливості системи не лише до статистичного оцінювання, а й до глибокої інтерпретації неструктурованих даних, що суттєво підвищує якість управлінських рішень.

На основі проведеного аналізу було сформульовано основні задачі майбутнього програмного забезпечення, яке має забезпечувати повний цикл роботи з освітньою аналітикою: від формування анкет і збору відповідей до математичної обробки результатів, побудови аналітичних звітів та візуалізації ключових показників. Визначено необхідність реалізації гнучкої структури опитувань, багаторівневої системи доступу, захищеного зберігання даних, математичних алгоритмів оцінювання та інтерактивної аналітичної панелі.

РОЗДІЛ 2
РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІТИКИ НАВЧАЛЬНИХ КУРСІВ

У другому розділі описано процес проєктування та реалізації системи. Було обрано клієнт-серверну архітектуру, яка дозволяє розділити клієнтську та серверну частини. Для frontend частини використано React, а для backend — Node.js та Express.js. Також було визначено основні функціональні вимоги, структуру системи та її ключові модулі.

2.1 Розробка архітектури програмного забезпечення

Програмне забезпечення системи побудоване на фундаментальній архітектурі клієнт-сервер, яка на сьогодні вважається одним із найбільш ефективних та універсальних підходів до побудови сучасних інформаційних систем. Саме така модель широко застосовується у веб-платформах, корпоративних інформаційних системах, електронних освітніх середовищах, CRM-системах, банківських сервісах та інших програмних продуктах, де необхідна централізована обробка великого обсягу даних і одночасна робота значної кількості користувачів [7, 8].

Суть клієнт-серверної архітектури полягає у чіткому розподілі функціональних обов’язків між двома основними складовими — клієнтом і сервером. Клієнтська частина відповідає за безпосередню взаємодію з користувачем, відображення інтерфейсу, введення даних та відправлення запитів до серверної частини. Сервер, у свою чергу, виступає центральним елементом системи, який приймає ці запити, виконує їх обробку, реалізує бізнес-логіку, забезпечує доступ до бази даних, виконує аналітичні розрахунки та повертає готовий результат користувачу у зручному форматі.

Для системи моніторингу та аналітики навчальних курсів така архітектура є особливо доцільною, оскільки дозволяє централізовано накопичувати великі масиви інформації про результати студентських опитувань, забезпечувати контроль доступу до даних, підтримувати анонімність респондентів та реалізовувати складні алгоритми статистичного аналізу без перевантаження клієнтських пристроїв. Усі обчислення виконуються на сервері, тоді як користувач взаємодіє лише з веб-інтерфейсом, що значно спрощує використання системи.

Центральним елементом архітектури виступає серверна частина програмного забезпечення, яка може бути реалізована як на фізичному сервері, розміщеному безпосередньо в межах університетської інфраструктури, так і у вигляді віртуального сервера в хмарному середовищі. Використання хмарних технологій сьогодні є особливо актуальним, оскільки вони дозволяють забезпечити високу доступність системи, резервне копіювання даних, гнучке масштабування ресурсів та зниження витрат на технічне обслуговування обладнання.

Сервер додатка виконує функції веб-сервера, приймає HTTP-запити від користувачів, проводить перевірку прав доступу, реалізує логіку роботи з анкетами, виконує математичну обробку результатів, формує аналітичні показники та забезпечує взаємодію з базою даних. Саме на цьому рівні реалізується основна бізнес-логіка системи: створення опитувальників, збереження відповідей, розрахунок середніх оцінок, побудова рейтингів, визначення проблемних зон освітнього процесу та генерація звітів для адміністрації університету.

Окрему важливу роль відіграє система управління базами даних, яка забезпечує структуроване збереження всієї інформації. У базі даних зберігаються відомості про дисципліни, викладачів, навчальні програми, семестри, категорії запитань, результати анкетування, статистичні показники та історія змін. Централізоване збереження даних дозволяє уникнути дублювання інформації, забезпечує її цілісність, спрощує резервне копіювання та підвищує рівень інформаційної безпеки.

Клієнтська частина системи представлена веб-інтерфейсом, доступ до якого здійснюється через звичайний браузер без необхідності встановлення додаткового

програмного забезпечення. Це є важливою перевагою для закладів вищої освіти, оскільки дозволяє забезпечити доступ до системи для великої кількості користувачів незалежно від типу пристрою чи операційної системи. Студенти можуть проходити анкетування з ноутбуків, планшетів або смартфонів, а викладачі та адміністрація — переглядати аналітичні звіти та статистичні панелі з персональних комп'ютерів.

Особлива увага приділяється адаптивності інтерфейсу, тобто його здатності автоматично підлаштовуватися під різні розміри екранів та формати пристроїв. Це забезпечує комфортну роботу як на великих моніторах, так і на мобільних телефонах, що суттєво підвищує рівень залучення студентів до процесу анкетування та покращує загальний користувацький досвід.

Мережеве з'єднання виступає ще одним важливим компонентом архітектури. Система підтримує як внутрішню роботу в локальній мережі університету, так і повноцінний віддалений доступ через мережу Інтернет. Такий підхід особливо важливий в умовах дистанційного навчання, змішаних форматів освіти та необхідності забезпечення безперервного доступу до освітніх сервісів незалежно від фізичного місця перебування користувача.

Для захисту інформації та забезпечення конфіденційності даних використовується захищений протокол HTTPS, який реалізує шифрування переданої інформації між клієнтом і сервером. Це особливо важливо при роботі з персональними даними, результатами опитувань та адміністративною звітністю. Додатково можуть застосовуватися механізми авторизації, рольового розмежування доступу, журналювання дій користувачів та резервного копіювання інформації.

Процес взаємодії між клієнтом і сервером відбувається за стандартною схемою. Користувач через веб-інтерфейс виконує певну дію — наприклад, відкриває анкету, надсилає відповідь або переглядає результати оцінювання. Сформований запит передається мережею до серверної частини, де проходить обробку. За необхідності сервер звертається до бази даних для отримання або запису інформації, після чого формує відповідь та повертає її клієнту. На стороні

користувача ця інформація відображається у вигляді таблиць, графіків, діаграм або інтерактивних аналітичних панелей.

Важливою перевагою клієнт-серверної моделі є її масштабованість. У разі збільшення кількості користувачів система може бути розширена шляхом додавання нових серверів, балансування навантаження або переходу до кластерної інфраструктури. Це дозволяє підтримувати стабільну роботу навіть при значному навантаженні під час масового проходження анкетування наприкінці семестру.

Таким чином, використання клієнт-серверної архітектури дозволяє забезпечити високу надійність, безпеку, централізоване управління даними, зручність користування та можливість подальшого масштабування системи. Саме така архітектурна модель найбільш повно відповідає вимогам сучасної цифрової трансформації вищої освіти та створює міцну технологічну основу для побудови ефективної системи моніторингу якості навчальних курсів.

Архітектура клієнт-сервер представлена на рис. 2.1.

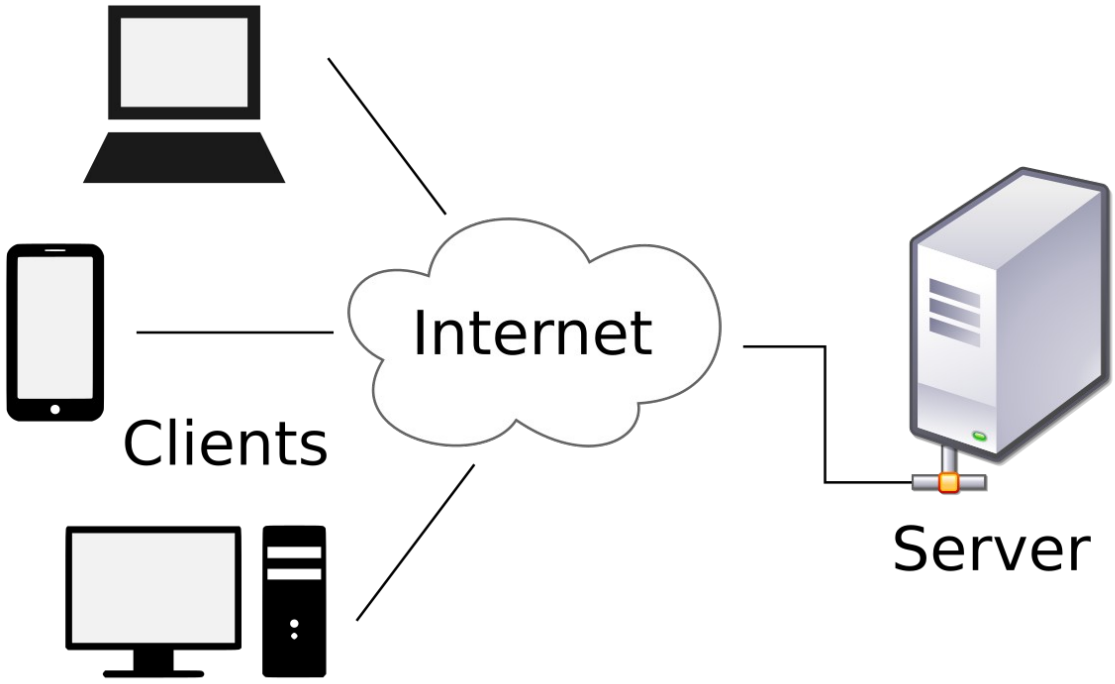


Рисунок 2.1 – Архітектура клієнт-сервер

2.2 Інженерія вимог

У процесі проєктування аналітичної платформи для моніторингу якості навчальних курсів одним із ключових етапів стало формування повного переліку функціональних і нефункціональних вимог до системи. Саме на цьому етапі визначаються не лише основні можливості програмного забезпечення, але й критерії його ефективності, зручності використання, надійності та придатності до реального впровадження в освітньому середовищі університету. Чітка постановка вимог дозволяє уникнути хаотичної розробки, забезпечує логічну структуру майбутнього програмного продукту та створює основу для подальшого технічного проєктування.

Функціональні вимоги описують ті конкретні задачі, які система повинна виконувати для досягнення своєї основної мети — автоматизації збору, обробки та аналізу студентського зворотного зв’язку. Центральним функціональним модулем виступає система створення та управління анкетами, оскільки саме через неї формується первинна інформація для подальшої аналітики.

Адміністратор повинен мати можливість створювати гнучкі опитувальники, які складаються з логічно структурованих блоків питань. Питання можуть бути згруповані за різними категоріями оцінювання, такими як якість викладання, актуальність змісту дисципліни, рівень практичної підготовки, доступність навчальних матеріалів, комунікація з викладачем, складність курсу, організація самостійної роботи та загальна задоволеність навчальним процесом. Такий підхід дозволяє отримувати не лише загальну оцінку курсу, а й деталізовану інформацію щодо окремих складових освітнього процесу.

Система повинна підтримувати різні типи запитань, що значно розширює її практичну цінність. До них належать шкальні питання за моделлю Лікерта, де студент оцінює твердження за визначеною шкалою, відкриті текстові поля для коментарів, варіанти з одиничним або множинним вибором, демографічні блоки, а також допоміжні питання для уточнення контексту навчання. Можливість налаштування послідовності питань, умовного відображення окремих блоків та

повторного використання шаблонів анкет значно підвищує гнучкість системи та спрощує роботу адміністратора.

Після створення анкети важливим завданням є організація самого процесу опитування. Проходження анкетування повинно базуватися на принципах добровільності, анонімності та простоти доступу. Студент отримує можливість пройти оцінювання через веб-інтерфейс за персональним посиланням або через внутрішню університетську систему. При цьому програмне забезпечення повинно контролювати унікальність участі — кожен студент може надати відповідь лише один раз для конкретної дисципліни в межах визначеного семестру. Такий механізм запобігає дублюванню результатів та підвищує достовірність аналітики.

Особливо важливим аспектом є забезпечення анонімності респондентів. Система не повинна зберігати персональні ідентифікаційні дані, такі як ПІБ, електронні адреси або IP-адреси, якщо це не є критично необхідним для технічного контролю доступу. Усі відповіді повинні прив’язуватися лише до курсу, семестру, освітньої програми та інших службових параметрів, але не до конкретної особи. Саме така модель дозволяє отримати чесний, відкритий і неупереджений зворотний зв’язок від студентів.

Після завершення етапу збору відповідей ключову роль починає відігравати аналітичний модуль системи. Його основне завдання полягає в автоматичній обробці великих масивів даних та перетворенні їх у зрозумілу управлінську інформацію. Система повинна розраховувати середні оцінки за кожним критерієм, визначати розподіли відповідей, формувати порівняльні таблиці між дисциплінами, викладачами, кафедрами та освітніми програмами. Це дозволяє швидко виявляти слабкі місця в організації навчального процесу та приймати обґрунтовані рішення щодо його вдосконалення.

Додатково можуть використовуватися складніші статистичні інструменти, зокрема коефіцієнти варіації, кореляційний аналіз, побудова трендів за кілька семестрів та прогнозування змін показників якості освіти. Наприклад, система може виявляти залежність між рівнем складності дисципліни та загальною задоволеністю студентів або між якістю практичної підготовки та підсумковою

успішністю. У перспективі можливе підключення методів машинного навчання для виявлення прихованих закономірностей та автоматичного формування рекомендацій для адміністрації.

Окремий функціональний блок становить модуль управління довідковою інформацією. Він включає ведення переліку дисциплін, викладачів, академічних груп, факультетів, кафедр, навчальних курсів, семестрів та інших структурних елементів освітнього середовища. Актуальність цих даних є критично важливою, оскільки саме на їх основі здійснюється правильна прив’язка результатів анкетування до конкретних навчальних об’єктів.

Не менш важливим є модуль управління доступом та ролями користувачів. Система повинна підтримувати чітке розмежування прав залежно від функціональної ролі користувача. Студенти мають доступ лише до проходження опитування, викладачі — до перегляду узагальнених результатів власних дисциплін без можливості ідентифікації окремих відповідей, а адміністративний персонал — до розширеної аналітики, формування офіційної звітності та управління структурою системи. Такий підхід забезпечує безпеку, конфіденційність і прозорість роботи платформи.

З технічної точки зору важливою функцією є можливість інтеграції з іншими університетськими сервісами. Система повинна підтримувати імпорт даних із LMS-платформ, електронних деканатів або внутрішніх реєстрів, а також експорт аналітичних звітів у формати CSV, Excel, PDF та інші поширені стандарти. Це спрощує подальшу роботу з даними та дозволяє інтегрувати платформу в загальну цифрову екосистему університету.

Окрім функціональних можливостей, велике значення мають нефункціональні вимоги, які визначають якість роботи системи. Однією з найважливіших є продуктивність. Платформа повинна стабільно працювати навіть при значному навантаженні, забезпечуючи одночасну роботу декількох тисяч користувачів без суттєвого зниження швидкодії. Формування аналітичних звітів має відбуватися протягом кількох секунд, а реакція інтерфейсу на дії користувача повинна бути максимально швидкою та природною.

Інтерфейс користувача повинен бути простим, інтуїтивно зрозумілим і адаптивним. Студент не повинен витрачати багато часу на проходження опитування — оптимальною вважається тривалість у межах 5–10 хвилин. Саме зручність користування безпосередньо впливає на відсоток участі студентів в анкетуванні та, відповідно, на якість отриманих результатів.

Надійність системи забезпечується механізмами транзакційного збереження даних, регулярного резервного копіювання, автоматичного відновлення після збоїв та безперервного моніторингу працездатності сервера. Для цього можуть використовуватися сучасні засоби контейнеризації та адміністрування, зокрема Docker, systemd, CI/CD-процеси та інші інструменти DevOps-підходу.

Окремо варто виділити вимогу модульності архітектури. Система повинна будуватися за принципами незалежності функціональних блоків, дотриманням SOLID-принципів та використанням сучасних шаблонів проєктування. Це спрощує подальшу підтримку програмного забезпечення, дозволяє масштабувати окремі компоненти незалежно один від одного та значно полегшує внесення змін у майбутньому.

Важливою складовою також є повноцінна документація системи. Вона включає опис структури бази даних, документацію REST API, інструкції для адміністратора та кінцевого користувача, технічний опис модулів, а також якісне коментування програмного коду. Документованість безпосередньо впливає на довговічність програмного продукту та забезпечує можливість його подальшого розвитку іншими розробниками.

Таким чином, сукупність функціональних і нефункціональних вимог формує цілісне бачення майбутньої аналітичної платформи як сучасного, масштабованого, безпечного та практично корисного програмного забезпечення. Саме системний підхід до формування вимог забезпечує створення рішення, яке не лише автоматизує процес збору зворотного зв’язку, а й стає реальним інструментом стратегічного управління якістю вищої освіти.

2.3 Структура програмної забезпечення

У межах реалізації програмного забезпечення для моніторингу та аналітики навчальних курсів було обрано двошарову архітектурну модель, яка складається з клієнтської частини (front-end) та серверної частини (back-end). Такий підхід є одним із найбільш поширених у сучасній веб-розробці, оскільки дозволяє чітко розмежувати логіку обробки даних і користувацький інтерфейс, забезпечуючи при цьому зручність підтримки, масштабованість та можливість подальшого розширення функціоналу системи.

На початковому етапі проєктування було визначено основні функціональні модулі серверного застосунку, які формують логічну основу всієї системи. Центральне місце займає модуль опитувань, який відповідає за створення, редагування, публікацію та архівацію анкет для адміністратора, а також за отримання студентом актуальних форм опитування та збереження наданих відповідей. Саме цей модуль забезпечує основну бізнес-логіку системи, оскільки через нього формується первинна інформація для подальшої аналітики.

Другим важливим компонентом є модуль аутентифікації та безпеки, який реалізує контроль доступу до системи, перевірку користувачів, керування ролями та захист API-запитів. У повноцінній реалізації він передбачає використання токенів авторизації та розмежування прав доступу між студентами, викладачами й адміністраторами. У межах прототипу допускається спрощений механізм входу без складної системи реєстрації, однак навіть у такому форматі зберігається принцип рольового доступу до функціоналу платформи.

Окрему роль виконує аналітичний модуль, який здійснює математичну обробку зібраних результатів. Його завданням є автоматичний розрахунок середніх оцінок, побудова статистичних розподілів, визначення тенденцій, аналіз кореляцій між окремими критеріями оцінювання та формування узагальнених звітів для адміністрації університету. Саме цей модуль перетворює звичайні відповіді студентів на практично корисну управлінську інформацію.

Ще одним важливим елементом системи є модуль керування довідниками, який забезпечує API для роботи зі списками навчальних курсів, дисциплін, курсів, категорій питань, семестрів та інших структурних елементів освітнього процесу. Завдяки цьому система зберігає актуальну організаційну структуру університету та дозволяє коректно прив’язувати результати опитувань до відповідних навчальних об’єктів.

Після визначення основних модулів була реалізована двошарова архітектура, яка дозволяє незалежно розвивати кожен із рівнів застосунку. Клієнтська частина відповідає за взаємодію з користувачем, відображення інформації та формування запитів до сервера, тоді як серверна частина забезпечує бізнес-логіку, обробку даних, автентифікацію, збереження інформації та формування аналітичних результатів. Такий принцип розподілу суттєво спрощує адміністрування системи та підвищує її стійкість до змін у майбутньому.

Серверна частина програмного продукту була реалізована із використанням середовища Node.js та фреймворку Express.js, які широко застосовуються для побудови швидких, масштабованих та продуктивних веб-застосунків. Node.js дозволяє виконувати JavaScript-код на стороні сервера, що значно спрощує розробку завдяки використанню однієї мови програмування як для front-end, так і для back-end частини. Це зменшує складність підтримки проєкту та пришвидшує процес розробки.

Фреймворк Express.js [8] використовується для організації серверної логіки, створення API-маршрутів, обробки HTTP-запитів, роботи з middleware та реалізації механізмів авторизації. Його головною перевагою є простота побудови REST API, що є особливо важливим для систем, де клієнтська частина активно взаємодіє із сервером через асинхронні запити.

Для збереження інформації на етапі прототипування було використано легку файлову JSON-базу даних, яка фізично розміщується у структурі проєкту у вигляді окремого файлу. Такий підхід є доцільним для навчальних систем, дипломних проєктів та MVP-рішень, де не передбачається надвелике навантаження або одночасна робота тисяч користувачів. JSON-сховище дозволяє швидко реалізувати

базові CRUD-операції без складного налаштування повноцінної системи керування базами даних.

Попри простоту такої реалізації, структура програмного забезпечення була побудована таким чином, щоб у майбутньому забезпечити легкий перехід до використання повноцінної СКБД, наприклад PostgreSQL, MySQL або MongoDB. Завдяки ізольованому рівню роботи з даними зміна механізму збереження інформації не потребуватиме суттєвої перебудови всієї системи, що позитивно впливає на довговічність програмного рішення.

Взаємодія між сервером і сховищем даних реалізується через набір API-ендпоїнтів, які забезпечують виконання основних функціональних операцій системи. Через API здійснюється автентифікація користувачів, отримання списку дисциплін, створення нових курсів, редагування опитувальників, надсилання студентських відповідей, формування аналітичних звітів та перегляд статистики. Усі запити та відповіді передаються у форматі JSON, що забезпечує універсальність, зручність інтеграції та відповідність сучасним стандартам веб-розробки.

Клієнтська частина системи реалізована з використанням бібліотеки React.js, яка дозволяє створювати сучасні інтерактивні односторінкові застосунки (Single Page Application), [6]. React забезпечує компонентний підхід до розробки, що значно підвищує повторне використання коду, спрощує підтримку проєкту та дозволяє логічно структурувати інтерфейс користувача.

Інтерфейс системи був розділений на окремі функціональні компоненти відповідно до ролей та задач користувачів. До основних компонентів належать сторінка авторизації, домашня сторінка, модуль перегляду курсів, сторінка проходження анкетування, адміністративна панель, модуль аналітики, система керування курсами та сторінка формування звітності. Такий підхід забезпечує чітку організацію структури застосунку та полегшує подальше розширення функціоналу.

Маршрутизація реалізована за допомогою React Router, що забезпечує логічний перехід між сторінками залежно від ролі користувача та його прав доступу. Наприклад, студент після входу отримує доступ лише до сторінок

проходження опитувань і перегляду власних доступних дисциплін, тоді як адміністратор має розширений функціонал: створення курсів, редагування анкет, керування довідниками та перегляд повної аналітики.

Механізм авторизації реалізовано через токенну модель доступу. Після успішного входу в систему користувач отримує токен авторизації, який зберігається на локальному пристрої та автоматично додається до всіх подальших API-запитів у заголовках HTTP-запитів. Це дозволяє серверу перевіряти автентичність кожного звернення та обмежувати доступ до захищених ресурсів.

Формування анкет та надсилання студентських оцінок реалізується через динамічні форми, де питання генеруються автоматично залежно від обраного курсу або типу опитування. Варіанти відповідей можуть бути представлені у вигляді шкал оцінювання, чекбоксів, відкритих текстових полів або комбінованих блоків. Після завершення анкети відповіді надсилаються на сервер у вигляді структурованого масиву об’єктів, що дозволяє легко виконувати подальшу аналітичну обробку.

Рольовий доступ реалізовано як на клієнтському, так і на серверному рівні. На front-end це забезпечує приховування недоступного функціоналу та побудову різних інтерфейсів для різних категорій користувачів. На back-end сервер додатково перевіряє права доступу до конкретних API-ендпоїнтів [8], що унеможливлює несанкціонований доступ навіть при спробі зовнішнього втручання.

Створена архітектурна модель дозволяє забезпечити високу гнучкість при побудові анкет, оскільки питання можуть формуватися індивідуально для кожного курсу, семестру або освітньої програми. Це дозволяє адаптувати систему під специфіку різних факультетів та забезпечує значно вищу практичну цінність порівняно зі статичними універсальними опитувальниками.

Архітектура програмної системи побудована таким чином, щоб забезпечувати прозору логіку взаємодії, модульність структури, незалежність компонентів та можливість подальшого масштабування. Обраний стек технологій був сформований з урахуванням актуальності сучасних підходів до веб-розробки, підтримки відкритих стандартів, простоти розгортання та зручності супроводу. Це

дозволяє не лише реалізувати ефективний навчальний прототип, але й створює надійну основу для подальшого переходу до повноцінної промислової інформаційної системи університетського рівня.

Під час реалізації серверної частини програмного забезпечення було сформовано базову структурну модель з окремими сутностями, які відображають основні об’єкти предметної області. До таких сутностей належать користувачі, курси, анкети, відповіді студентів, навчальні програми, категорії запитань та службові довідники. Кожна з них має чітко визначений набір атрибутів, що дозволяє забезпечити логічну цілісність даних, коректну взаємодію між компонентами системи та подальшу можливість масштабування архітектури.

На етапі створення прототипу для зберігання інформації було використано файлове сховище у форматі JSON, яке виконує функції спрощеної бази даних. Усі дані зберігаються у вигляді структурованих масивів в окремому файлі всередині проєкту. Такий підхід дозволяє значно спростити розробку, уникнути складного налаштування повноцінної СКБД та зосередитися безпосередньо на реалізації бізнес-логіки системи. Для навчальних проєктів, прототипів та MVP-рішень це є цілком виправданим рішенням, оскільки забезпечує швидке тестування, локальне розгортання та зручність демонстрації функціоналу.

Попри спрощену модель збереження даних, структура застосунку була спроєктована таким чином, щоб у майбутньому можна було без суттєвих змін перейти до використання повноцінних систем керування базами даних, таких як PostgreSQL, MySQL або MongoDB. Це досягається завдяки ізоляції логіки доступу до даних від основної бізнес-логіки сервера, що відповідає принципам модульного проєктування сучасних вебсистем.

Обробка клієнтських запитів реалізована за допомогою фреймворку Express.js, який використовується для побудови REST API. У серверному застосунку створено набір REST-контролерів, кожен із яких відповідає за конкретну функціональну область системи: автентифікацію, роботу з курсами, управління анкетами, збереження відповідей, аналітичну обробку результатів та

формування звітності. Такий підхід дозволяє чітко структурувати логіку застосунку та забезпечує зручність подальшого супроводу коду.

Кожен маршрут API обробляє HTTP-запити відповідного типу — GET, POST, PUT або DELETE — залежно від виконуваної операції. Наприклад, GET-запити використовуються для отримання списків курсів, результатів опитувань або аналітичних даних, POST — для створення нових записів та надсилання відповідей студентів, PUT — для редагування існуючих анкет, а DELETE — для видалення або архівації застарілої інформації.

Особливу увагу було приділено механізму збереження відповідей студентів на анкети. Для цього реалізовано окремий маршрут типу POST, який приймає структурований масив відповідей від клієнтської частини. Перед записом нових даних система виконує перевірку: чи користувач уже проходив оцінювання конкретного курсу в межах відповідного семестру. Якщо попередній запис існує, він автоматично видаляється, а нові відповіді зберігаються замість нього. Такий підхід дозволяє уникнути дублювання інформації, забезпечує актуальність даних та надає студенту можливість повторного проходження анкети без спотворення загальної статистики.

Маршрут типу GET забезпечує отримання всіх збережених відповідей, які використовуються для побудови адміністративної аналітики. Саме на основі цих даних формуються середні оцінки, рейтинги курсів, порівняльні таблиці, статистичні розподіли та інші управлінські звіти для адміністрації університету. У подальшому така модель може бути розширена автоматичною генерацією PDF-звітів, експортом у Excel або інтеграцією з BI-системами.

Клієнтська частина системи реалізована у вигляді React-застосунку зі структурованою компонентною архітектурою. Такий підхід дозволяє розділити інтерфейс на незалежні функціональні блоки, кожен із яких відповідає за конкретну частину взаємодії з користувачем. Наприклад, окремо реалізовані компоненти авторизації, домашньої сторінки, списку дисциплін, форми проходження анкетування, адміністративної панелі, сторінки перегляду аналітики та керування курсами.

Компонентний підхід значно покращує підтримуваність проєкту, спрощує повторне використання елементів інтерфейсу та дозволяє легко розширювати функціонал без порушення вже реалізованої логіки. Кожен компонент має власну локальну відповідальність, що відповідає принципам чистої архітектури та сучасним стандартам front-end розробки.

На поточному етапі реалізації питання анкети представлені у вигляді жорстко закодованого масиву даних, що дозволяє швидко перевірити основну логіку роботи системи без додаткового динамічного конструктора анкет. Для кожного запитання користувачу пропонується вибір відповіді через HTML-елементи типу <select>, що забезпечує простий та зрозумілий інтерфейс проходження опитування.

Після завершення заповнення анкети всі обрані значення автоматично збираються у структурований масив об’єктів, де кожен об’єкт містить ідентифікатор питання, значення відповіді та службову інформацію про курс або семестр. Далі ці дані передаються на сервер за допомогою HTTP POST-запиту, де проходять перевірку та збереження. Такий формат передачі є зручним для подальшої аналітичної обробки та відповідає принципам REST-взаємодії між клієнтом і сервером.

Візуальне оформлення інтерфейсу реалізовано з використанням сучасних стилів, що дозволило швидко побудувати адаптивний, структурований і зручний користувацький інтерфейс без необхідності створення великої кількості власних CSS-рішень. У проєкті активно використовується grid-система для побудови сторінок, картки для відображення курсів, кнопки для керування функціоналом, індикатори статусу, модальні вікна, форми введення та інші стандартні UI-компоненти.

Використання стилів забезпечує коректне відображення системи на різних типах пристроїв — від настільних комп’ютерів до планшетів і смартфонів, що особливо важливо для студентської аудиторії, яка часто проходить анкетування саме з мобільних пристроїв. Адаптивність інтерфейсу безпосередньо впливає на зручність користування та рівень залучення користувачів до системи.

Проєкт реалізовано з дотриманням принципів чистої архітектури. Логіка застосунку поділена на окремі контексти, бізнес-правила ізольовані від візуального представлення, а компоненти мають єдину відповідальність. Назви змінних, функцій та структур даних визначені відповідно до принципів читабельності, зрозумілості та підтримуваності коду. Це значно спрощує подальшу розробку, зменшує кількість потенційних помилок та полегшує командну роботу над проєктом.

Особливу роль відіграє використання токенної моделі автентифікації, яка дозволяє чітко відокремити клієнтську та серверну частини застосунку. Після авторизації користувач отримує токен доступу, який використовується для підтвердження прав при виконанні API-запитів. Такий підхід підвищує безпеку системи та наближає реалізацію до реальних стандартів розробки вебзастосунків.

2.4 Висновки до розділу 2

Отже, у другому розділі було сформовано практичну основу для реалізації програмного забезпечення моніторингу та аналітики навчальних курсів, починаючи від архітектурного проєктування і завершуючи безпосередньою побудовою функціонального прототипу системи. Проведений аналіз підтвердив, що використання клієнт-серверної архітектури є найбільш доцільним рішенням для подібного типу інформаційних систем, оскільки саме така модель забезпечує централізоване зберігання даних, безпечну обробку інформації, підтримку багаторівневого доступу та можливість подальшого масштабування системи відповідно до потреб університету. Клієнтська частина забезпечує зручну взаємодію користувача з платформою, тоді як серверна сторона виконує бізнес-логіку, аналітичні розрахунки та управління всіма процесами обробки освітніх даних.

У межах інженерії вимог було визначено повний комплекс функціональних і нефункціональних характеристик, яким має відповідати система для ефективного

впровадження в освітньому середовищі. Було встановлено необхідність реалізації гнучкого конструктора анкет, підтримки анонімного проходження опитувань, багаторівневої системи доступу, аналітичного модуля для статистичної обробки результатів, а також інтеграції з іншими університетськими сервісами. Окрема увага приділялася продуктивності, адаптивності інтерфейсу, безпеці передачі даних, надійності збереження інформації та модульності архітектури. Саме системний підхід до формування вимог дозволив побудувати чітку логіку майбутнього програмного продукту та закласти основу для його стабільної роботи в реальних умовах експлуатації.

У підпункті, присвяченому структурі програмного забезпечення, було реалізовано двошарову модель застосунку на основі сучасного вебстеку Node.js, Express.js та React.js. Серверна частина забезпечує роботу REST API, автентифікацію користувачів, збереження відповідей, аналітичну обробку результатів та управління довідковими даними. Клієнтська частина побудована за компонентним принципом із використанням React, Context API, React Router, що дозволило створити адаптивний, зручний та функціонально завершений інтерфейс для студентів, викладачів і адміністрації. Використання JSON-сховища на етапі прототипування спростило реалізацію системи та водночас забезпечило можливість подальшого переходу до повноцінної СКБД без суттєвих змін у структурі застосунку.

Таким чином, другий розділ підтвердив технічну здійсненність розробленої системи та продемонстрував, що обрані архітектурні рішення, інженерні підходи та програмні технології повністю відповідають поставленим завданням дипломного проєкту. Реалізований функціональний прототип доводить можливість створення сучасної освітньої аналітичної платформи, яка не лише автоматизує процес збору студентського зворотного зв’язку, а й виступає ефективним інструментом управління якістю освіти, підтримки прийняття управлінських рішень та цифрової трансформації університетського середовища загалом.

РОЗДІЛ 3
ПРИКЛАДНЕ ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ТА АНАЛІТИКИ НАВЧАЛЬНИХ КУРСІВ

У третьому розділі продемонстровано роботу розробленої системи на практиці. Було показано створення анкет, проходження опитування, обробку результатів, перегляд аналітики та формування звітів. Також проведено тестування, яке підтвердило працездатність основних функцій системи.

3.1 Демонстрація можливостей програмного забезпечення

У даному підрозділі продемонстровано практичні можливості розробленого програмного забезпечення для моніторингу та аналітики навчальних курсів. Основною метою системи є автоматизація процесу збору, обробки, аналізу та візуалізації зворотного зв’язку від студентів щодо якості викладання навчальних дисциплін, а також забезпечення адміністративного контролю за освітнім процесом.

Для демонстрації роботи системи використовується тестове середовище, яке моделює реальні умови функціонування університетської освітньої програми. Прикладом виступає освітня програма «Комп’ютерна інженерія», у межах якої студенти проходять оцінювання дисциплін весняного семестру. До переліку дисциплін входять такі курси, як «Інженерія програмного забезпечення», «Бази даних», «Програмна інженерія » та інші.

Система реалізована у вигляді веб-застосунку з клієнтською частиною на основі React та серверною частиною на Node.js з використанням Express.js. Для збереження даних використовується структурований JSON-файл, що дозволяє забезпечити простоту розгортання, швидкість тестування та наочність роботи системи в межах кваліфікаційної роботи.

У системі реалізовано розмежування прав доступу між двома основними ролями: студентом та адміністратором. Кожен користувач проходить процедуру авторизації за електронною поштою та паролем. Дані користувачів зберігаються у файлі users.json із використанням хешування паролів для підвищення безпеки доступу до системи.

Після відкриття веб-застосунку користувач потрапляє на стартову сторінку, де йому пропонується виконати вхід до системи або пройти процедуру реєстрації нового облікового запису.

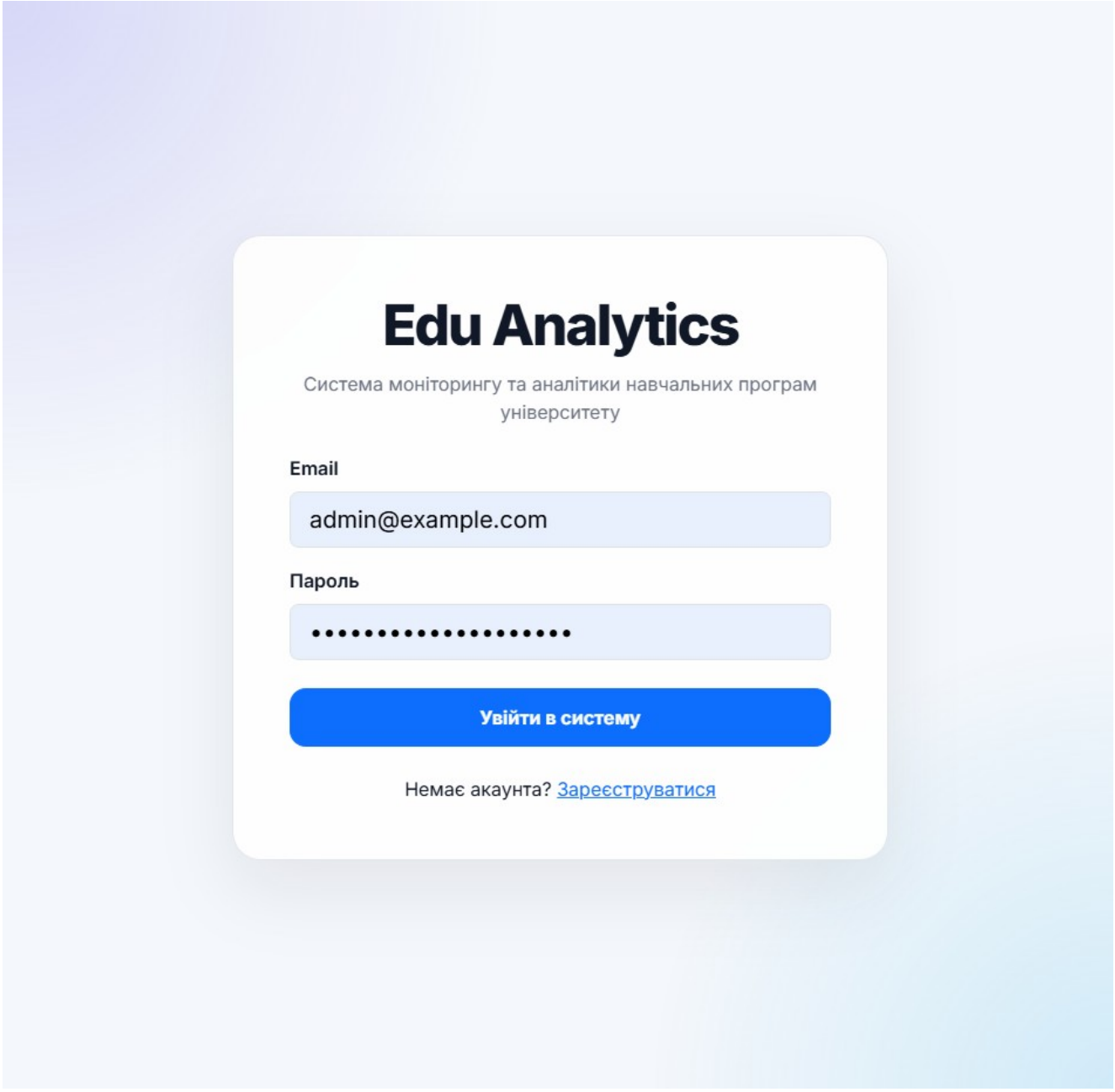


Рисунок 3.1 – Авторизація для наявного акаунту

Форма авторизації містить стандартні поля для введення електронної пошти та пароля.

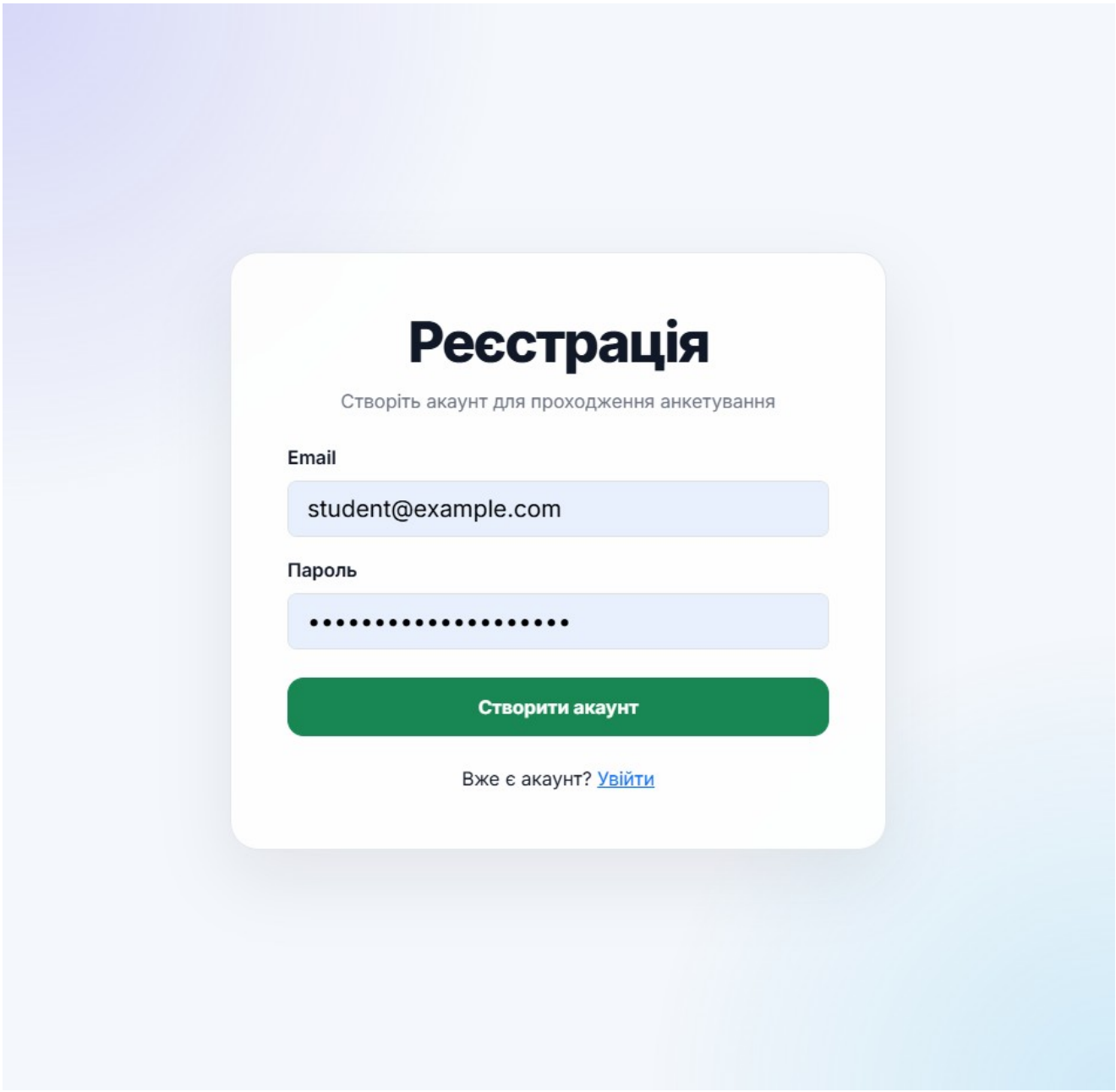


Рисунок 3.2 – Реєстрація нового користувача

У випадку некоректного введення даних система виконує валідацію та повідомляє користувача про помилку входу.

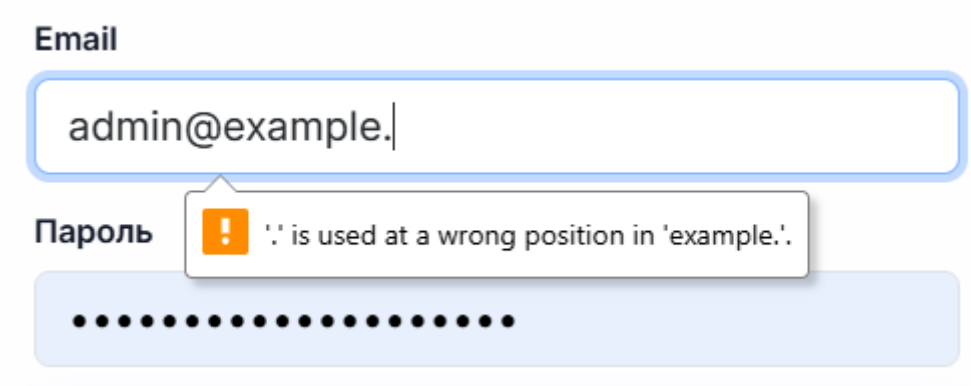


Рисунок 3.3 – Некоректне введення даних

Після реєстрації система повідомляє про успішність цієї дії, якщо всі дані були заповнені коректно, і далі користувач має можливість авторизуватися. Після успішної авторизації студент переходить до сторінки доступних дисциплін. Тут відображається перелік курсів, які доступні для оцінювання.

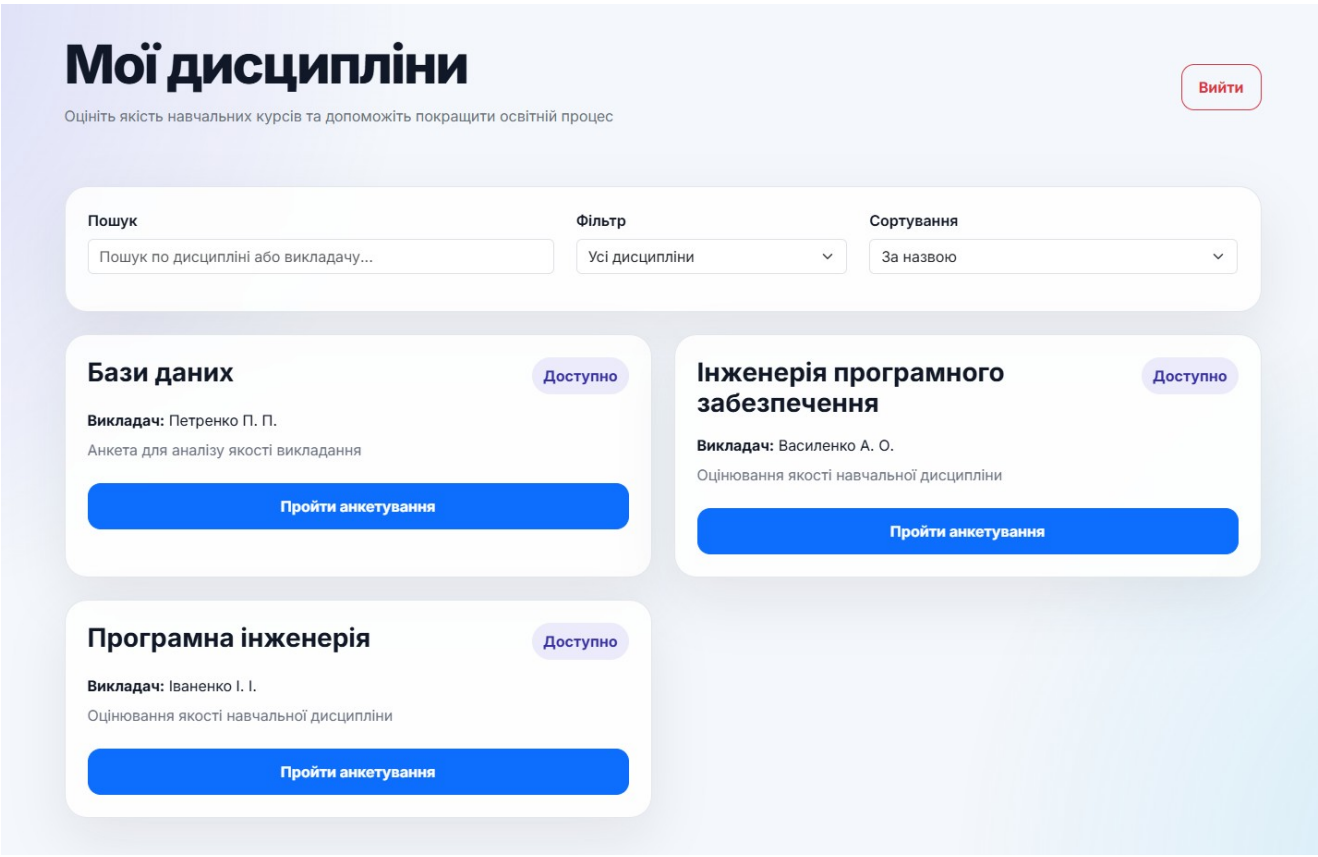


Рисунок 3.4 – Перелік курсів, доступних для проходження студентом

Для підвищення зручності використання реалізовано сучасну систему пошуку, фільтрації та сортування дисциплін. Студент може здійснювати пошук за назвою дисципліни або прізвищем викладача, фільтрувати дисципліни за статусом проходження («доступні», «завершені») та сортувати список за різними параметрами. Такий підхід значно покращує користувацький досвід, особливо у випадку великої кількості навчальних дисциплін.

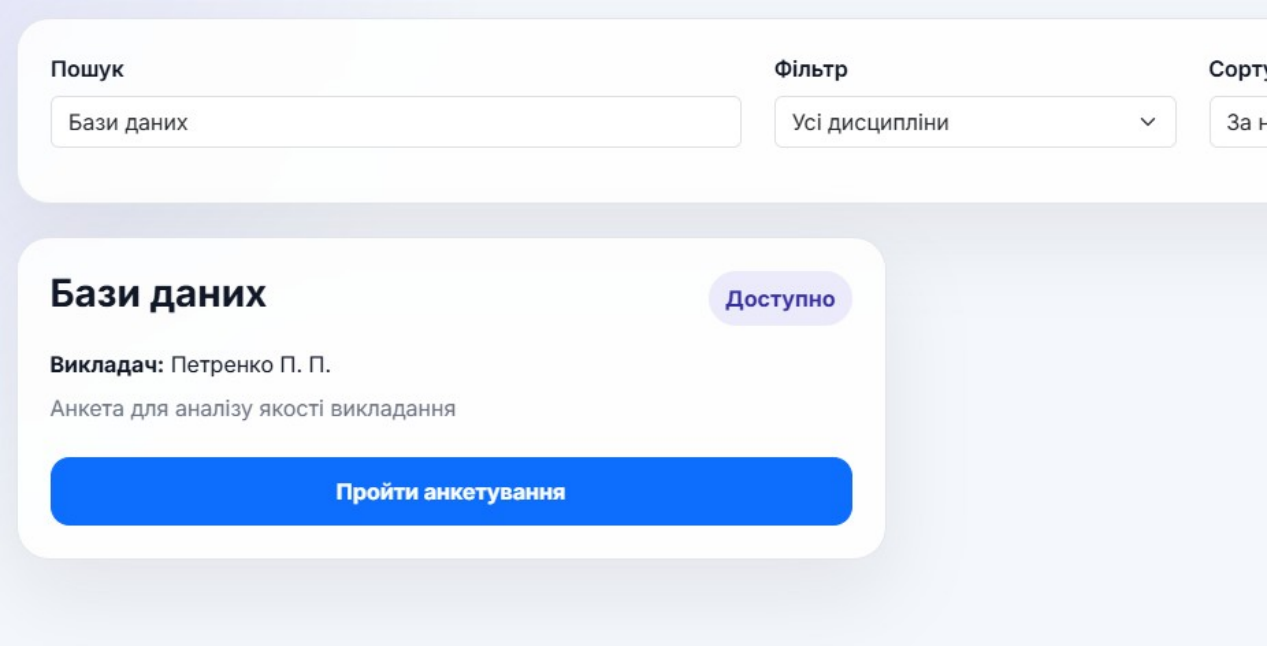


Рисунок 3.5 – Фільтрація за назвою дисципліни

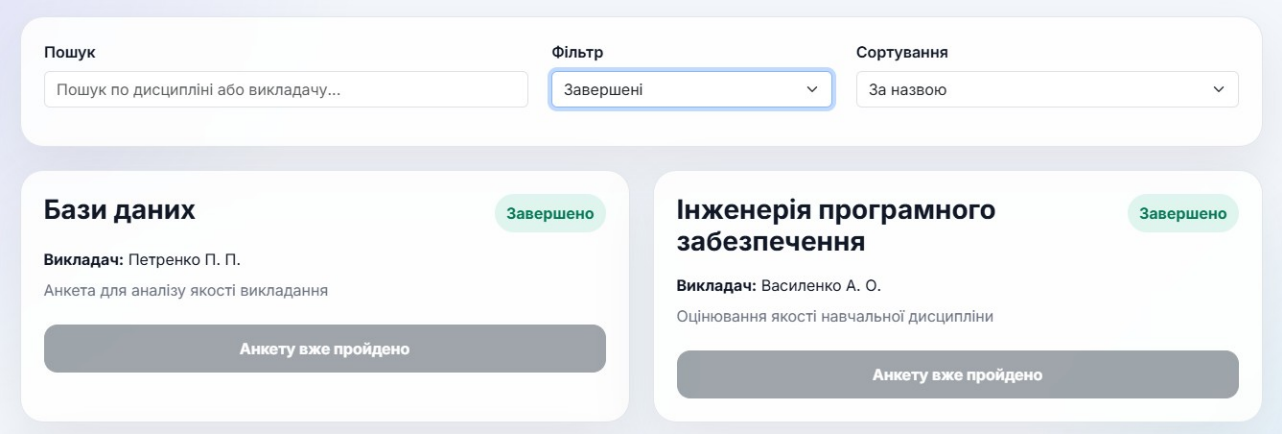


Рисунок 3.6 – Фільтрація за статусом дисципліни

Кожна дисципліна відображається у вигляді окремої інформаційної картки, що містить назву курсу, ім’я викладача, короткий опис дисципліни та кнопку переходу до анкетування.

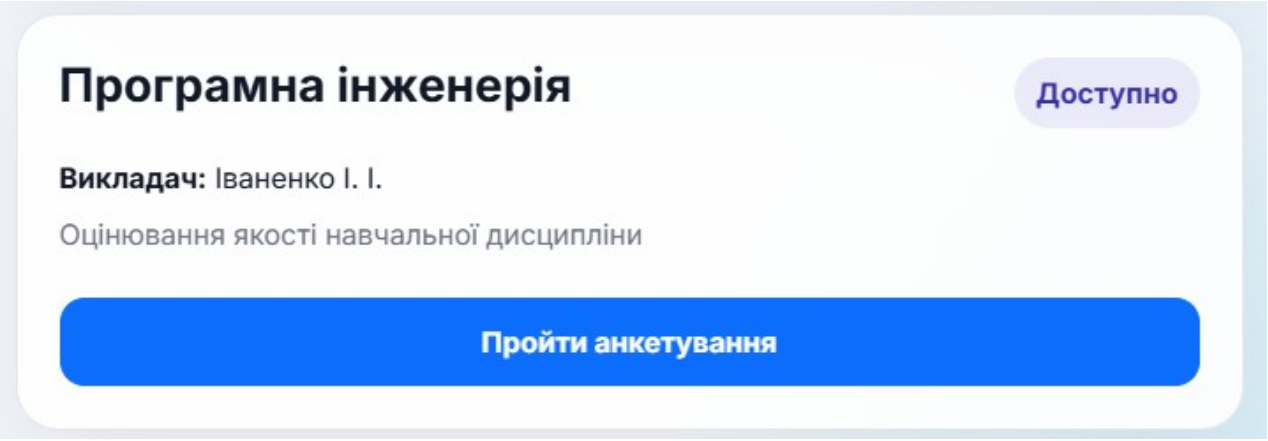


Рисунок 3.7 – Картка дисципліни

Якщо студент уже проходив анкетування для конкретного курсу, система автоматично блокує повторне проходження та замість кнопки «Пройти анкетування» відображає статус «Анкету вже пройдено». Така перевірка реалізована на серверному рівні та забезпечує об’єктивність статистичних результатів.

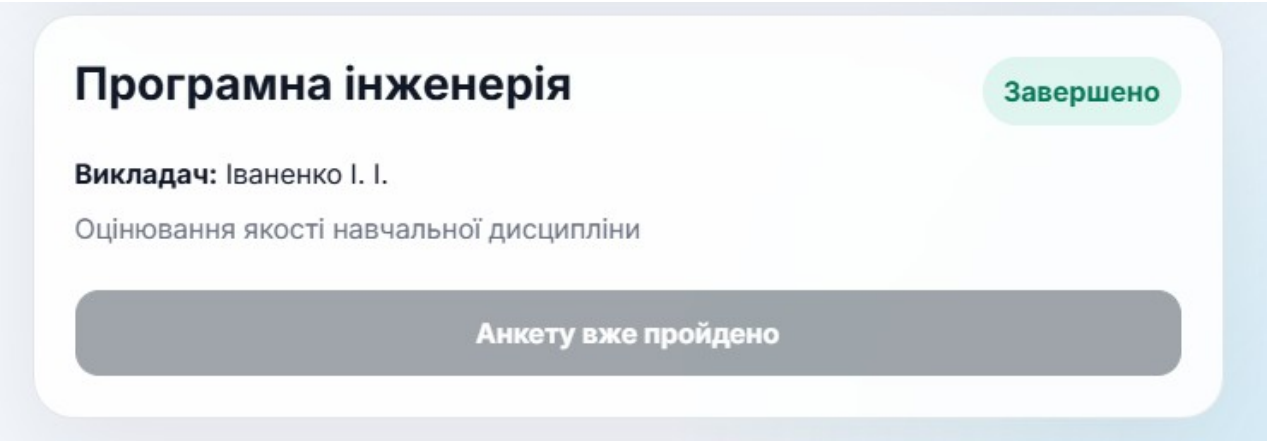


Рисунок 3.8 – Картка дисципліни після проходження

При переході до анкети оцінювання студент отримує форму опитування, що складається з десяти питань різних типів. Частина питань реалізована у форматі рейтингового оцінювання за шкалою від 1 до 5, де студент оцінює зрозумілість викладання, актуальність змісту дисципліни, логічність структури курсу, корисність практичних завдань, об’єктивність оцінювання, якість зворотного зв’язку, відповідність дисципліни майбутній професійній діяльності та загальне враження від курсу.

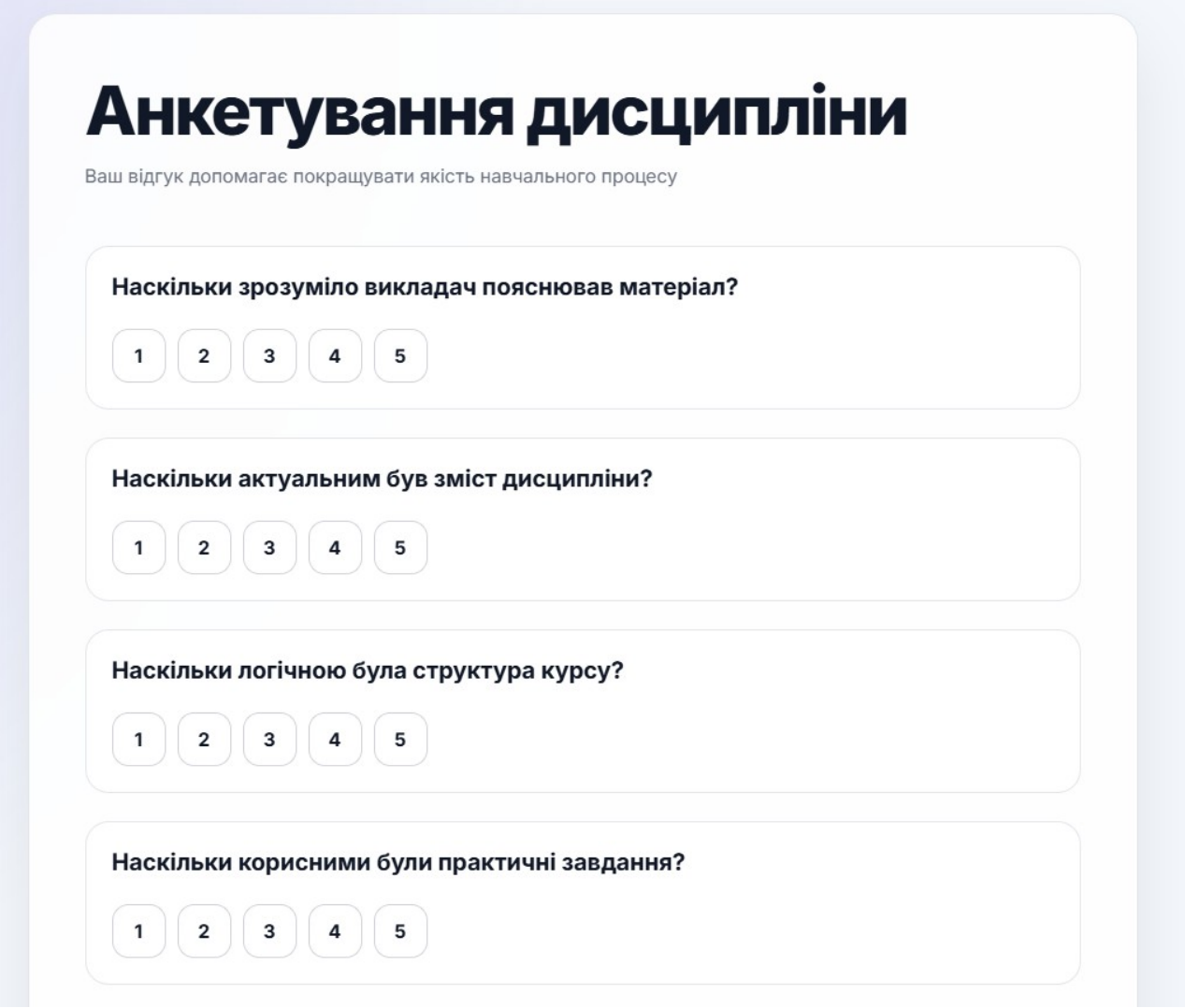


Рисунок 3.9 – Приклад запитань з числовою оцінкою

Окрім числового оцінювання, система містить відкриті текстові питання, що дозволяють студенту сформулювати власну думку щодо сильних сторін

дисципліни та запропонувати можливі покращення. Саме ці відповіді формують якісну складову аналітики та дозволяють адміністрації університету отримувати не лише числові показники, а й реальні коментарі студентів.

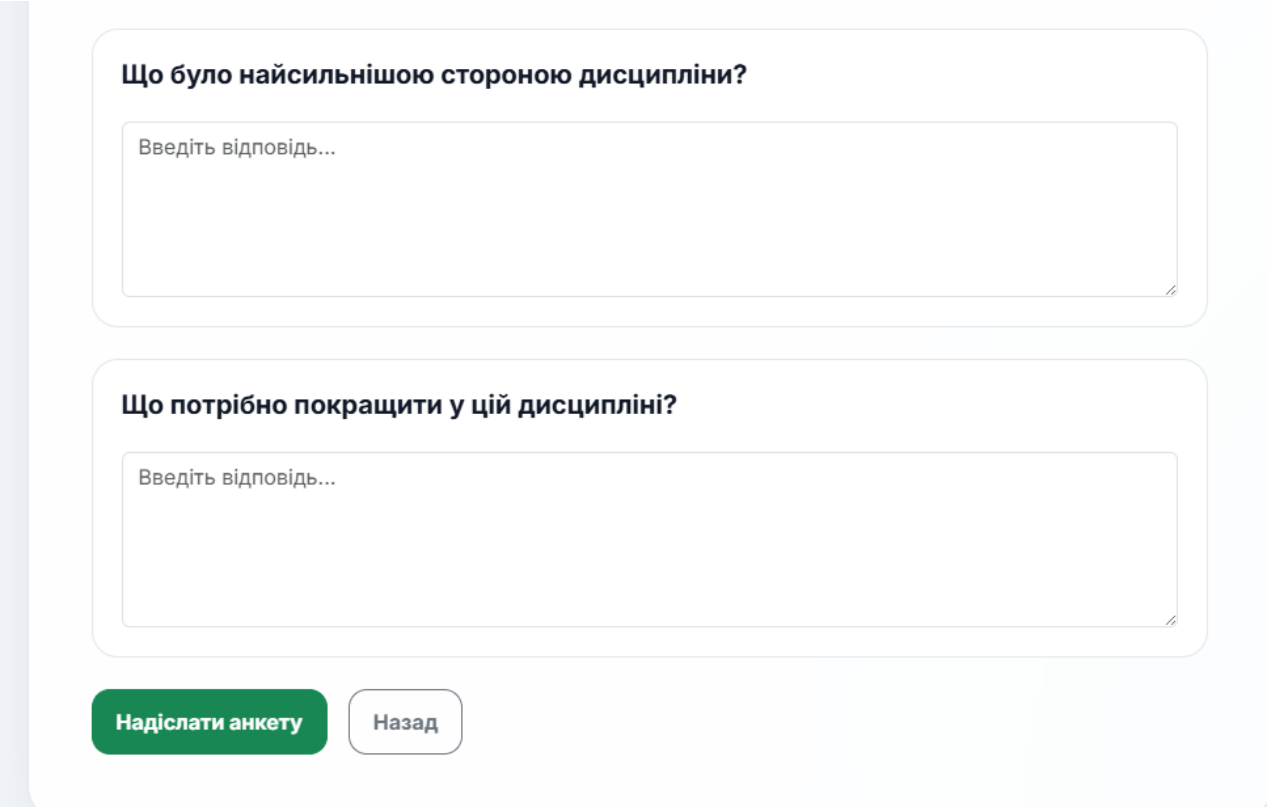


Рисунок 3.10 – Приклад запитань з відкритою відповіддю

Перед надсиланням анкети система виконує повну валідацію всіх обов’язкових полів. У випадку пропуску хоча б одного питання користувач отримує повідомлення про необхідність завершення заповнення форми. Після успішного проходження анкети відповіді зберігаються у базовому JSON-сховищі, а студент автоматично повертається до списку дисциплін, де статус анкети змінюється на завершений.

Окрему функціональну роль у системі виконує адміністратор. Тип запитань та їх склад задається адміністратором, є динамічним та може змінюватися в залежності від потреб університету або конкретного адміністратора.

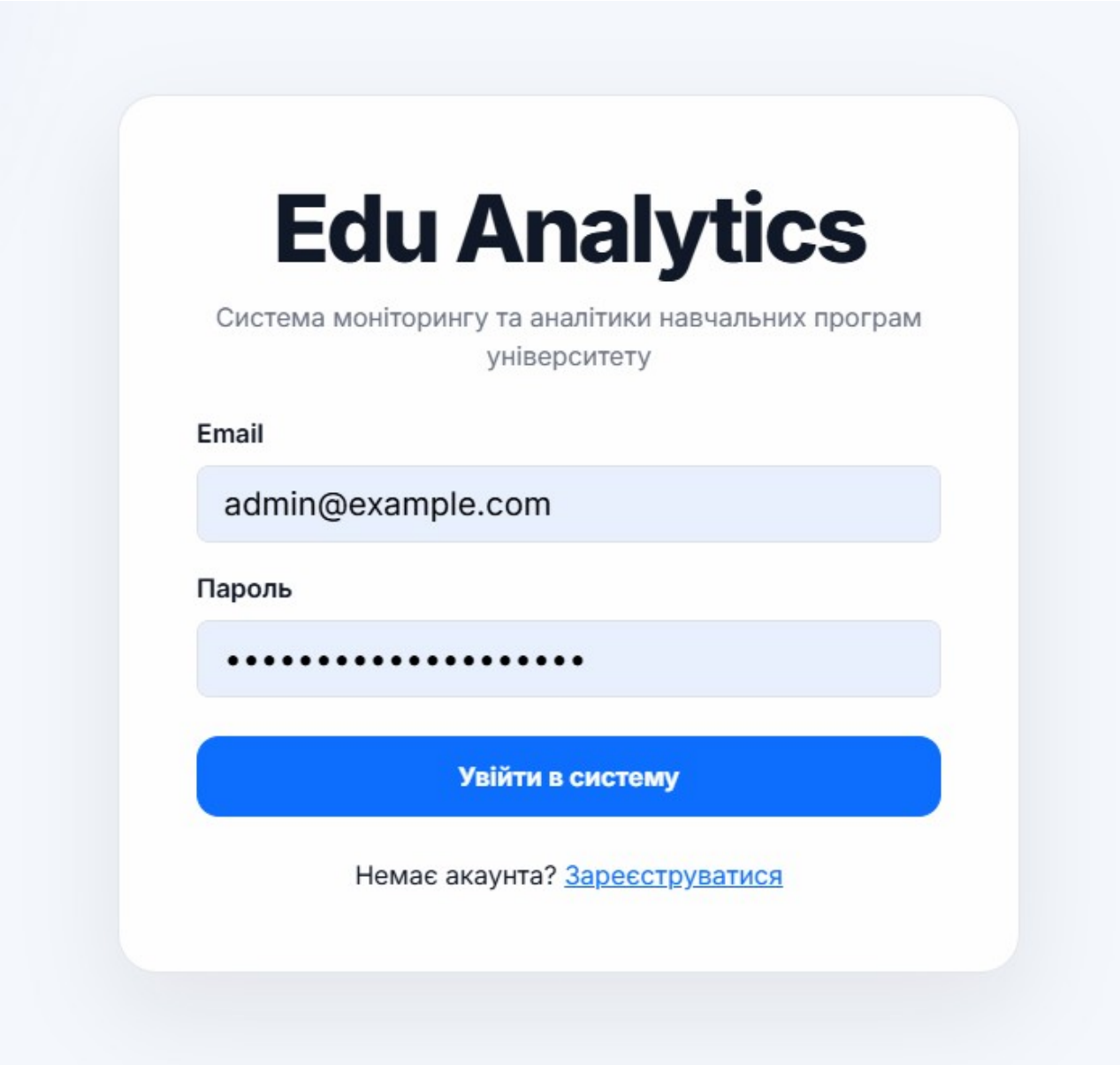


Рисунок 3.12 – Авторизація у якості адміністратора

Після авторизації з адміністративними правами користувач отримує доступ до розширеної панелі управління, яка включає декілька функціональних модулів: аналітика, управління дисциплінами, управління анкетами, рекомендації системи, історія змін та генерація PDF-звітів.

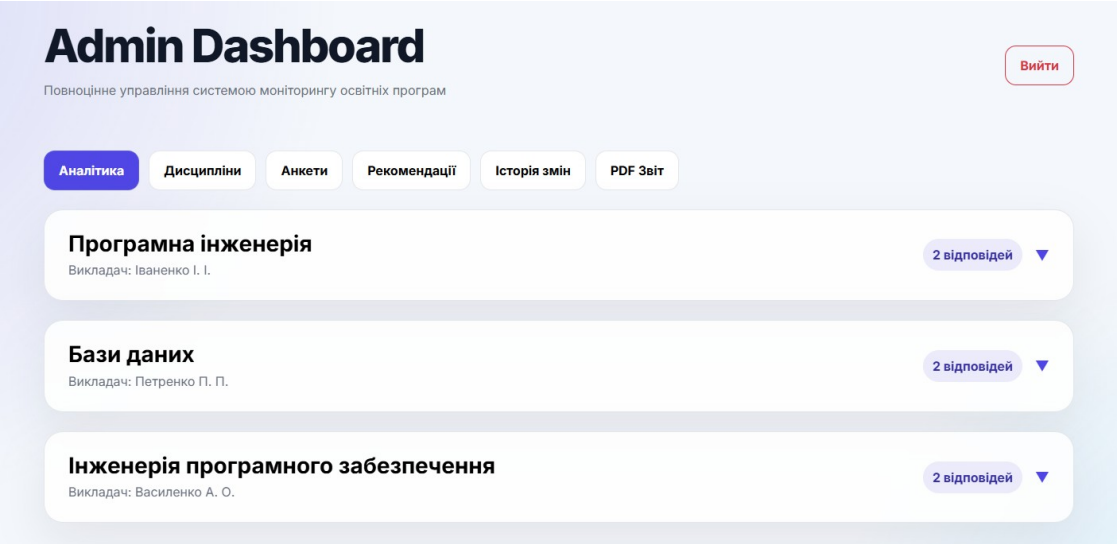


Рисунок 3.13 – Панель адміністратора

У вкладці аналітики адміністратор отримує доступ до статистичних показників для кожної дисципліни. Для зручності роботи з великою кількістю курсів реалізовано механізм згортання та розгортання блоків дисциплін. У згорнутому вигляді відображається лише назва курсу, викладач та кількість отриманих відповідей, а при розгортанні відкривається детальна статистика.

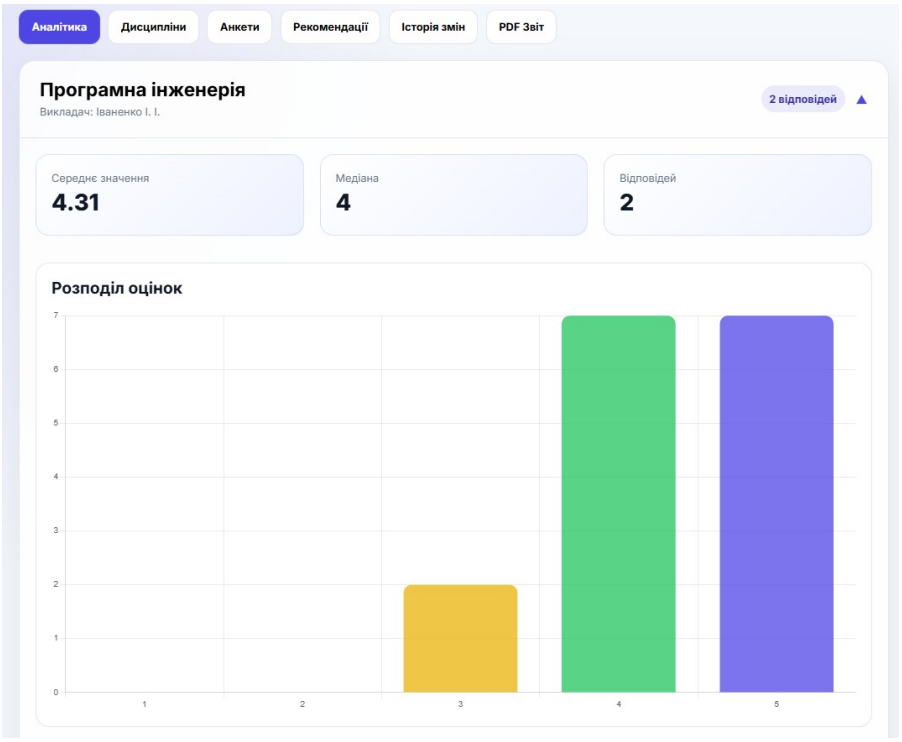
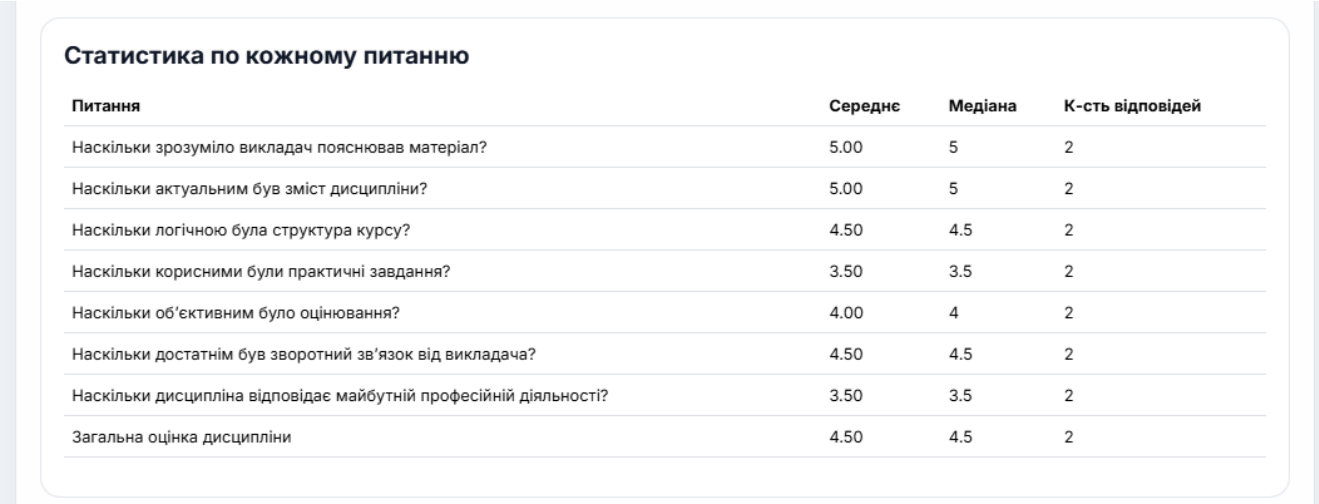


Рисунок 3.14 – Розгорнута інформація щодо дисципліни

Аналітичний блок містить середнє значення оцінювання, медіанне значення, кількість відповідей, розподіл оцінок за шкалою від 1 до 5 у вигляді графіків, а також деталізовану статистику по кожному окремому питанню анкети.

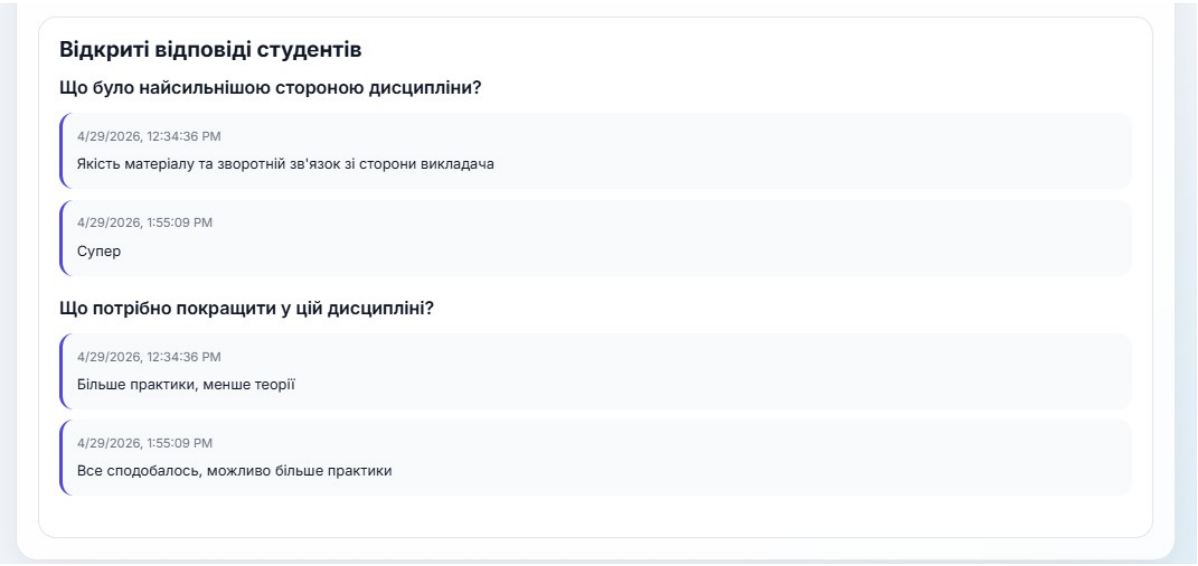


Статистика по кожному питанню

Питання	Середнє	Медіана	К-сть відповідей
Наскільки зрозуміло викладач пояснював матеріал?	5.00	5	2
Наскільки актуальним був зміст дисципліни?	5.00	5	2
Наскільки логічною була структура курсу?	4.50	4.5	2
Наскільки корисними були практичні завдання?	3.50	3.5	2
Наскільки об'єктивним було оцінювання?	4.00	4	2
Наскільки достатнім був зворотний зв'язок від викладача?	4.50	4.5	2
Наскільки дисципліна відповідає майбутній професійній діяльності?	3.50	3.5	2
Загальна оцінка дисципліни	4.50	4.5	2

Рисунок 3.15 – Аналітичний блок щодо кожного питання

Додатково відображаються всі відкриті текстові відповіді студентів із часовими позначками надсилання, що дозволяє адміністрації отримати повну картину якості викладання дисципліни.



Відкриті відповіді студентів

Що було найсильнішою стороною дисципліни?

4/29/2026, 12:34:36 PM
Якість матеріалу та зворотній зв'язок зі сторони викладача

4/29/2026, 1:55:09 PM
Супер

Що потрібно покращити у цій дисципліні?

4/29/2026, 12:34:36 PM
Більше практики, менше теорії

4/29/2026, 1:55:09 PM
Все сподобалось, можливо більше практики

Рисунок 3.16 – Відкриті відповіді студентів

Функціонал управління дисциплінами реалізує повноцінний CRUD-механізм. Адміністратор може створювати нові дисципліни, редагувати вже існуючі курси, змінювати викладача, опис дисципліни та видаляти непотрібні записи. При створенні нового курсу система автоматично формує базову анкету з типовими питаннями оцінювання, що значно прискорює роботу адміністратора.

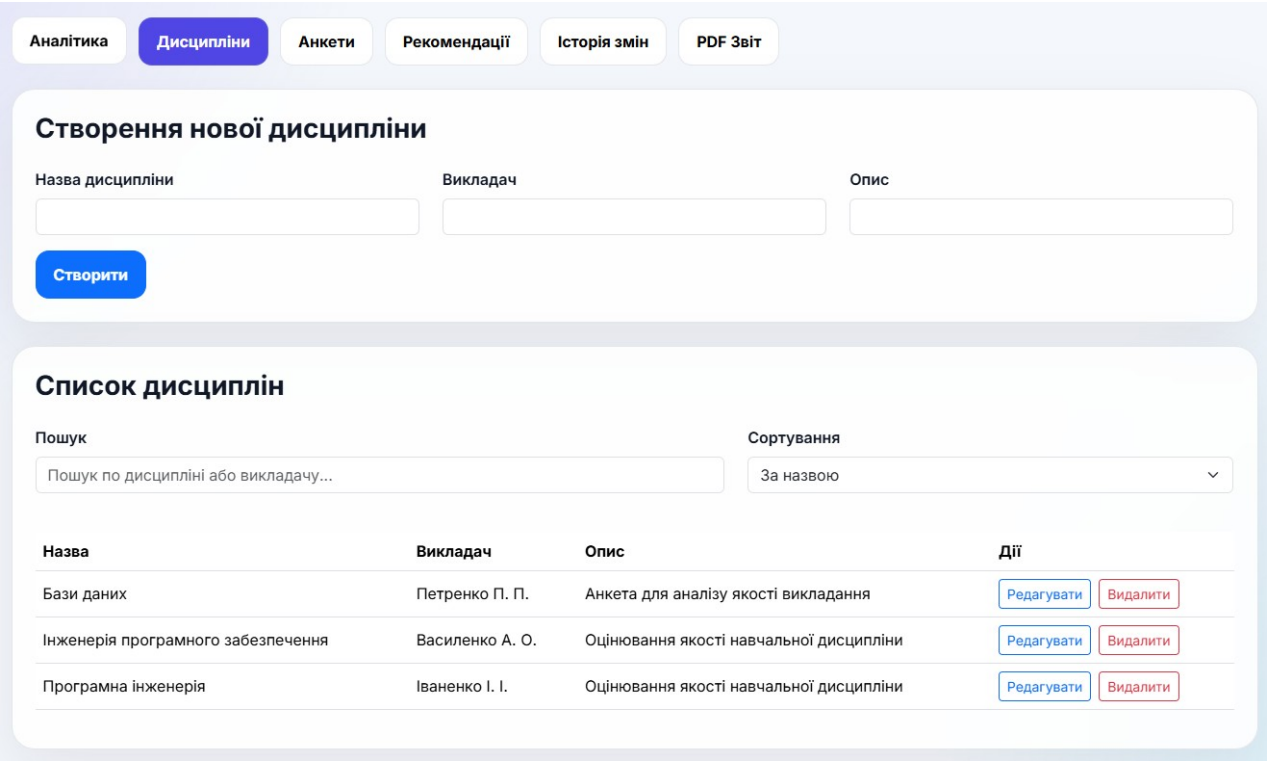


Рисунок 3.17 – CRUD механізм роботи з дисциплінами

Окремо реалізовано модуль конструктора анкет (Survey Builder), який дозволяє адміністратору формувати структуру опитування без необхідності внесення змін у програмний код. Адміністратор може додавати нові питання, змінювати їхній тип (числове оцінювання або відкрита текстова відповідь), визначати обов’язковість заповнення та видаляти непотрібні елементи. Такий підхід значно підвищує гнучкість системи та дозволяє адаптувати її до різних освітніх програм.

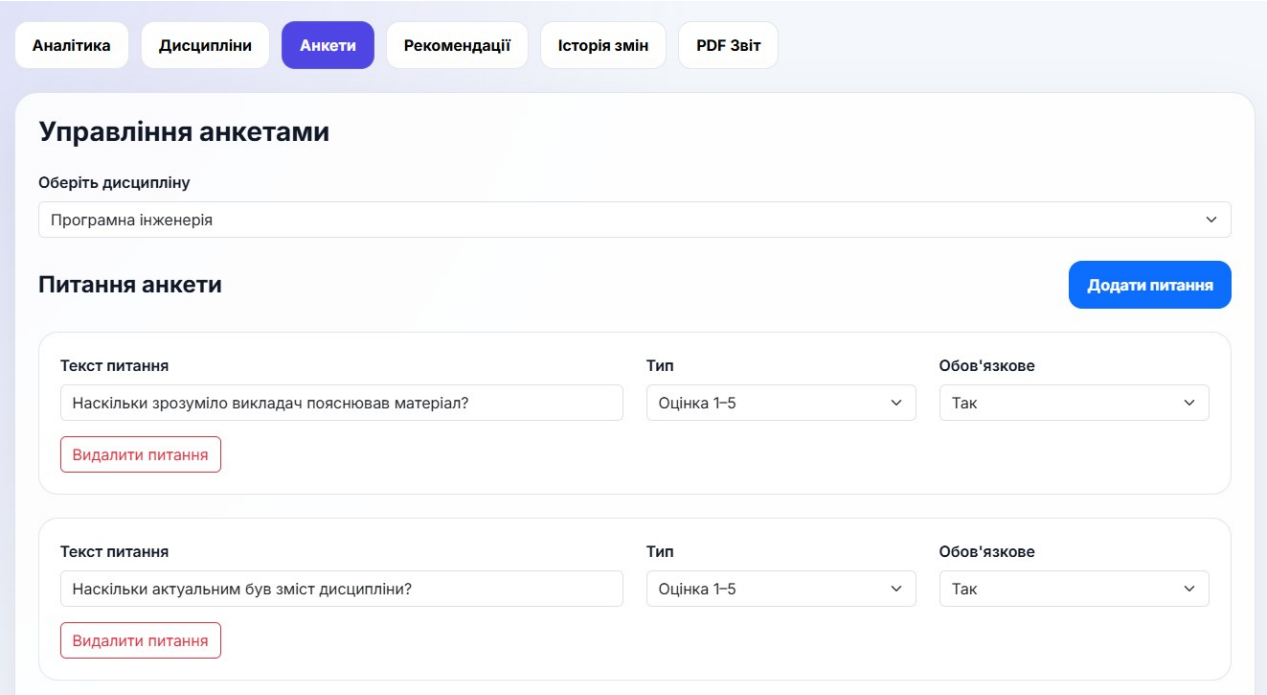


Рисунок 3.18 – Управління анкетами

Вкладка рекомендацій реалізує елементи інтелектуального аналізу даних. Система автоматично аналізує середні показники оцінювання дисциплін та визначає курси, які потребують додаткової уваги. Наприклад, якщо середня оцінка дисципліни є низькою, система формує рекомендацію щодо перегляду змісту курсу, методики викладання або структури практичних завдань. Це дозволяє адміністрації швидко виявляти проблемні зони без ручного аналізу всіх результатів.

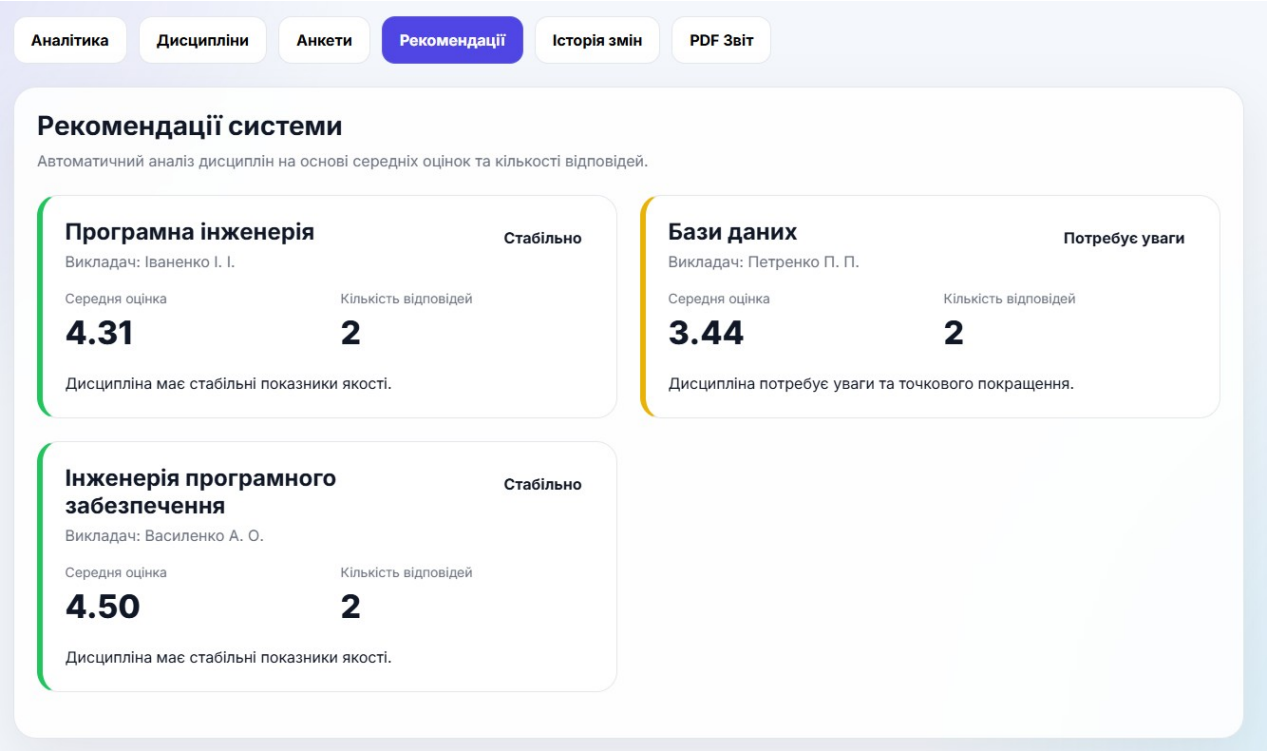


Рисунок 3.19 – Рекомендації системи

Модуль історії змін (Audit Log) забезпечує повний контроль адміністративних дій у системі. Тут фіксуються всі операції зі створення, редагування та видалення дисциплін, а також зміни в анкетах. Для кожної дії зберігається тип операції, опис змін, електронна пошта адміністратора та точний час виконання. Це дозволяє забезпечити прозорість роботи системи та контроль відповідальності за внесені зміни.

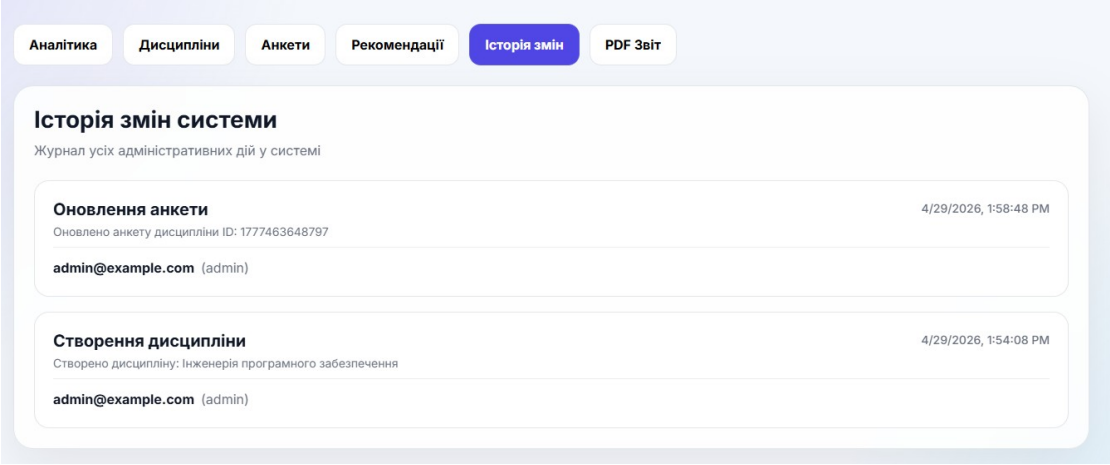


Рисунок 3.20 – Модуль історії змін

Останнім важливим модулем є автоматичне формування PDF-звітів. Адміністратор має можливість одним натисканням сформувати структурований звіт по всіх дисциплінах із ключовими показниками якості викладання. Такий документ може використовуватися для внутрішньої університетської звітності, роботи кафедр, акредитаційних процедур або подальшого аналізу ефективності освітніх програм.

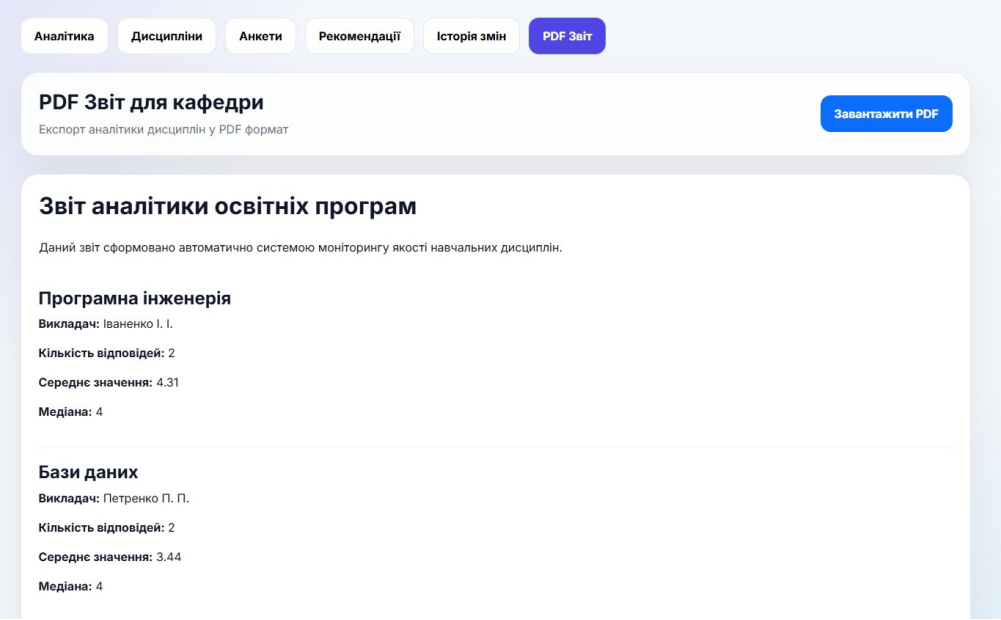


Рисунок 3.21 – Формування звіту для кафедри

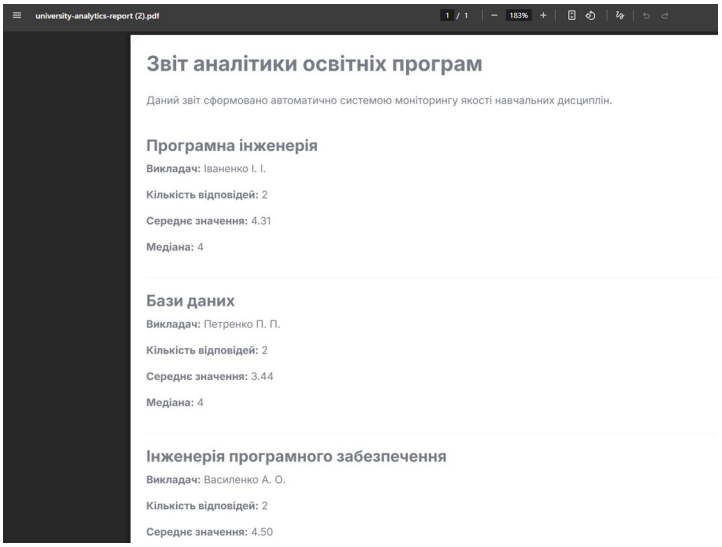


Рисунок 3.22 – Вигляд та структура PDF звіту

Таким чином, розроблена система забезпечує не лише збір студентських відгуків, а й повноцінну аналітичну підтримку процесу управління якістю освіти. Вона поєднує зручний користувацький інтерфейс, автоматизовану обробку даних, розширені можливості адміністрування та сучасні інструменти аналітики, що дозволяє значно підвищити ефективність внутрішнього моніторингу освітніх програм університету.

3.2 Тестування розробленої системи

Важливою складовою реалізації програмного забезпечення для моніторингу та аналітики навчальних курсів є перевірка коректності, стабільності та надійності роботи всієї системи. У процесі тестування основна увага приділялася правильному функціонуванню як клієнтської, так і серверної частини застосунку, перевірці логіки взаємодії між модулями, а також забезпеченню коректної роботи для всіх типів користувачів — студентів та адміністраторів.

Під час перевірки особлива увага приділялася процесам авторизації, реєстрації, проходження анкетування, обмеження повторного проходження опитування, побудови аналітики, роботі адміністративної панелі та формуванню PDF-звітів. Окремо тестувалася система перевірки ролей користувачів, яка забезпечує чітке розмежування прав доступу між студентами та адміністраторами.

У випадку введення некоректних даних під час авторизації система автоматично виконує перевірку наявності користувача у файлі users.json, правильності введеного пароля та відповідності хешованого значення. Якщо користувач вводить неправильну електронну пошту або пароль, система виводить повідомлення про помилку авторизації та не дозволяє виконати вхід до застосунку. Після виправлення помилкових даних користувач успішно отримує доступ до відповідного функціоналу згідно зі своєю роллю.

Аналогічно реалізована перевірка при проходженні анкетування. Якщо студент залишає незаповненими обов’язкові поля, система виконує валідацію

введених данх і не дозволяє завершити надсилання анкети. Особливо це стосується рейтингових питань за шкалою від 1 до 5 та відкритих текстових відповідей, які мають обов’язковий характер. У разі виявлення пропущених полів користувач отримує повідомлення про необхідність завершення заповнення анкети. Лише після коректного введення всіх необхідних даних система дозволяє зберегти результати опитування.

Важливим елементом тестування стала перевірка механізму одноразового проходження анкетування. Для кожної дисципліни студент має лише одну спробу проходження анкети, що реалізовано на серверному рівні через перевірку вже збережених відповідей у файлі db.json. У випадку повторної спроби система блокує можливість повторного надсилання відповідей, а на сторінці дисциплін відображається статус «Анкету вже пройдено». Такий підхід дозволяє забезпечити об’єктивність статистичних результатів та виключити можливість штучного впливу на аналітику.

Окремо проводилося тестування адміністративної панелі. Було перевірено коректність CRUD-операцій для дисциплін: створення нових курсів, редагування існуючих, видалення дисциплін, автоматичне створення базових анкет, а також робота конструктора анкет. Адміністратор має можливість безпосередньо через інтерфейс створювати нові питання, змінювати типи питань, визначати їхню обов’язковість та зберігати оновлену структуру опитування без внесення змін у програмний код.

Особливу увагу приділено перевірці аналітичного модуля. Було протестовано правильність розрахунку середніх значень, медіан, кількості відповідей, побудови графіків розподілу оцінок та відображення відкритих текстових відповідей студентів. Додатково перевірено роботу модуля рекомендацій, який автоматично визначає дисципліни з низькими показниками задоволеності та формує рекомендації щодо покращення якості викладання.

Також виконувалося тестування журналу змін (Audit Log), де фіксуються всі адміністративні дії в системі: створення, редагування та видалення дисциплін,

зміни в анкетах, а також формування PDF-звітів. Це дозволяє забезпечити прозорість роботи системи та контроль відповідальності за внесені зміни.

Окремо було проведено безпекове тестування механізму авторизації та захисту API-запитів. Для цього використовувалися JWT-токени, що забезпечують захищену передачу даних між клієнтською та серверною частинами застосунку. Перевірено неможливість доступу до адміністративних маршрутів без відповідних прав доступу, а також коректність роботи middleware для перевірки ролей користувачів.

У процесі перевірки працездатності системи було проведено декілька основних типів тестування, які дозволили підтвердити коректність реалізації всіх функціональних можливостей програмного забезпечення.

Першим етапом стало юніт-тестування, під час якого перевірялися окремі функції системи, що відповідають за базову логіку роботи застосунку. Зокрема, тестувалася робота функцій авторизації користувачів, обробки форм реєстрації, перевірки правильності введених даних, а також функції handleSubmit, що відповідає за відправлення анкет та збереження відповідей студентів. Це дозволило переконатися у правильності роботи окремих незалежних модулів.

Наступним етапом було інтеграційне тестування, яке перевіряло взаємодію між клієнтською та серверною частинами системи. Основна увага приділялася процесам авторизації, передачі JWT-токенів, коректному перенаправленню користувачів після входу в систему, а також перевірці role-based access control — правильному розмежуванню доступу між студентами та адміністраторами. Було підтверджено, що всі API-запити коректно обробляються сервером, а фронтенд правильно відображає отримані дані.

Системне тестування дозволило перевірити роботу всієї платформи як єдиного цілісного програмного продукту. У межах цього етапу аналізувалася відповідність застосунку поставленим функціональним та нефункціональним вимогам, зокрема стабільність роботи при переходах між сторінками, коректність роботи анкетування, побудови аналітики, генерації PDF-звітів та збереження інформації у файловій структурі бази даних.

Окремо було проведено тестування інтерфейсу користувача (UI), під час якого перевірялася зручність взаємодії з платформою, зрозумілість форм, коректність відображення повідомлень про помилки, адаптивність сторінок, робота пошуку, фільтрації та сортування дисциплін. Особлива увага приділялася сучасному вигляду інтерфейсу, логічному розташуванню елементів та простоті навігації для користувачів різного рівня підготовки.

Тестування обробки помилок охоплювало перевірку ситуацій із введенням некоректних даних. Наприклад, перевірялися випадки неправильного логіна або пароля, незаповнених обов'язкових полів в анкетах, повторної спроби проходження опитування, а також спроби доступу до недоступних маршрутів. У всіх випадках система коректно виявляла помилки, повідомляла користувача про проблему та не дозволяла виконати некоректну операцію.

Важливим етапом стало тестування ролей користувачів, яке підтвердило правильне розмежування прав доступу між студентами та адміністраторами. Студенти мають доступ лише до перегляду дисциплін та проходження анкетування, тоді як адміністратори можуть працювати з аналітикою, створювати нові дисципліни, редагувати анкети, переглядати історію змін, формувати рекомендації та експортувати звіти. Це забезпечує безпечну та логічну структуру використання системи.

Безпекове тестування було спрямоване на перевірку механізму JWT-авторизації та захисту API-маршрутів. Було підтверджено, що доступ до адміністративних ендпоїнтів неможливий без валідного токена та відповідної ролі користувача. Middleware-система коректно перевіряє права доступу, а хешування паролів забезпечує додатковий рівень захисту персональних даних користувачів.

Окремо перевірявся модуль PDF Export, який відповідає за автоматичне формування звітів для кафедри або адміністрації університету. Було протестовано правильність формування звітів, коректне відображення статистичних показників, структурування інформації по дисциплінах та стабільність роботи генератора PDF-файлів при великій кількості аналітичних даних.

Результати всіх проведених тестів підтвердили, що система працює стабільно, відповідає поставленим вимогам та може використовуватися як повноцінний інструмент для моніторингу якості освітніх програм університету. Реалізовані механізми захисту, аналітики, адміністрування та збору зворотного зв’язку забезпечують не лише технічну надійність, а й практичну цінність розробленого програмного забезпечення. розроблена система демонструє високу стабільність роботи, коректність бізнес-логіки, безпечне розмежування доступу та практичну придатність для впровадження в освітньому середовищі університету.

3.3 Програмна документація користувача

Завершальною складовою реалізації програмного забезпечення для моніторингу та аналітики навчальних курсів є підготовка супровідної документації як для кінцевих користувачів системи, так і для розробників, які в майбутньому можуть здійснювати підтримку, модифікацію або подальший розвиток платформи. Наявність якісної документації є важливою ознакою завершеного програмного продукту, оскільки забезпечує простоту впровадження, експлуатації та масштабування системи. Основними користувачами розробленого застосунку є студенти та адміністратори, тому для кожної ролі було сформовано відповідні інструкції використання.

Для студента робота із системою починається з отримання доступу до опитування. Після завершення навчального модуля або наприкінці семестру студент отримує інформацію про необхідність проходження анкетування через навчальну платформу університету або через адміністративне повідомлення. Після переходу до веб-застосунку користувач бачить сторінку авторизації, де необхідно ввести електронну пошту та пароль. У системі реалізовано окремі ролі користувачів, тому доступ до функціоналу надається відповідно до прав студента або адміністратора.

Після успішного входу студент переходить до сторінки з переліком доступних дисциплін для оцінювання. Для зручності роботи реалізовано сучасний інтерфейс із пошуком, фільтрацією та сортуванням курсів. Студент може швидко знайти необхідну дисципліну за назвою або викладачем, відфільтрувати вже завершені або доступні для проходження анкети, а також впорядкувати список за різними параметрами. Такий підхід особливо важливий при великій кількості дисциплін протягом кількох років навчання.

Для кожного курсу відображається окрема інформаційна картка, що містить назву дисципліни, викладача, короткий опис та кнопку переходу до анкетування. Якщо студент уже проходив оцінювання конкретного курсу, система автоматично змінює статус на «Анкету вже пройдено» та блокує можливість повторного надсилання відповідей. Це забезпечує чесність оцінювання та запобігає викривленню статистичних результатів.

Під час заповнення анкети студент послідовно відповідає на всі поставлені запитання. Система містить як рейтингові питання за шкалою від 1 до 5, так і відкриті текстові питання, де студент може залишити власні коментарі щодо сильних сторін дисципліни або запропонувати шляхи її покращення. Усі відповіді рекомендується надавати об’єктивно та чесно, оскільки саме вони формують основу подальшого аналітичного звіту.

Після натискання кнопки надсилання система виконує повну перевірку заповнення всіх обов’язкових полів. Якщо деякі питання залишилися без відповіді, користувач отримує повідомлення про помилку, а незаповнені поля додатково підсвічуються. Після виправлення помилок система повторно перевіряє введені дані та зберігає результати у випадку коректного заповнення. Після успішного завершення студент отримує підтвердження надсилання анкети та повертається до списку дисциплін.

Важливим аспектом функціонування системи є забезпечення конфіденційності. Хоча авторизація студентів здійснюється через особистий обліковий запис, самі відповіді не відображаються викладачам у прив’язці до конкретного користувача. Аналітика формується в узагальненому вигляді, що

дозволяє зберегти анонімність та забезпечити відкритість студентів при формуванні зворотного зв’язку.

Для адміністратора система передбачає значно ширший функціонал, зосереджений у спеціальній адміністративній панелі. Після авторизації адміністратор отримує доступ до вкладок аналітики, управління дисциплінами, управління анкетами, рекомендацій системи, журналу змін та модуля PDF Export.

Перед початком нового семестру або нового циклу опитувань адміністратор може створювати нові дисципліни через вкладку «Управління дисциплінами». Для цього необхідно вказати назву курсу, викладача та короткий опис дисципліни. Після створення система автоматично формує базову анкету з типовими питаннями оцінювання, що значно пришвидшує процес адміністрування. Також доступні функції редагування існуючих дисциплін та видалення застарілих курсів.

Окремо реалізовано конструктор анкет (Survey Builder), який дозволяє адміністратору повністю керувати структурою опитування без внесення змін у програмний код. Через графічний інтерфейс можна додавати нові питання, змінювати їхній тип (оцінювання за шкалою або текстова відповідь), визначати обов’язковість заповнення, редагувати формулювання та видаляти непотрібні елементи. Такий підхід забезпечує гнучкість системи та можливість адаптації під різні освітні програми.

У процесі проведення опитування адміністратор може відстежувати активність студентів через аналітичний модуль. Система відображає кількість отриманих відповідей, середні значення, медіану, розподіл оцінок та відкриті текстові відповіді студентів. Для зручності реалізовано механізм згортання та розгортання аналітики по кожній дисципліні окремо, що дозволяє ефективно працювати навіть із великою кількістю курсів.

Додатково реалізовано модуль рекомендацій, у якому система автоматично аналізує результати анкетування та визначає дисципліни, що потребують додаткової уваги. Якщо середній показник задоволеності є низьким, адміністратор отримує автоматичну рекомендацію щодо перегляду методики викладання, змісту

курсу або практичної складової дисципліни. Це дозволяє значно прискорити процес прийняття управлінських рішень.

Модуль історії змін (Audit Log) забезпечує прозорість роботи адміністративної частини системи. У журналі фіксуються всі дії адміністратора: створення, редагування та видалення дисциплін, зміни в анкетах, формування звітів. Для кожної операції зберігається тип дії, короткий опис, електронна пошта адміністратора та точний час виконання. Це дозволяє контролювати відповідальність за внесені зміни та спрощує аудит системи.

Після завершення опитування адміністратор має можливість сформувати PDF-звіт, який використовується для передачі результатів на кафедру, деканат або під час внутрішніх процедур контролю якості освіти. Звіт автоматично містить основні статистичні показники по всіх дисциплінах, що значно спрощує підготовку офіційної документації.

Технічне обслуговування системи полягає у підтримці актуальності програмного забезпечення, резервному копіюванні файлів даних та контролі коректної роботи серверної частини застосунку. Оскільки система побудована на Node.js, Express.js та React, адміністратор або технічний спеціаліст може легко розгорнути її на іншому сервері, змінити конфігурацію портів, оновити залежності або виконати міграцію до повноцінної СУБД у майбутньому.

Для розробників було підготовлено технічну документацію, яка містить опис архітектури системи, структуру REST API, логіку взаємодії frontend та backend, опис файлової структури проєкту, інструкції з розгортання, перелік необхідних залежностей та можливі напрямки подальшого розвитку системи. Серед перспектив вдосконалення можна виділити інтеграцію з PostgreSQL, додавання email-сповіщень, Docker-контейнеризацію, розширення ролей користувачів та інтеграцію з університетськими LMS-платформами.

Таким чином, наявність детально підготовленої користувацької та технічної документації забезпечує повноцінну готовність розробленого програмного забезпечення до впровадження, подальшої підтримки та масштабування в реальному освітньому середовищі університету.

3.4 Програмна документація розробника

Для розгортання та подальшої підтримки розробленого програмного забезпечення розробнику необхідно мати встановлене середовище Node.js, менеджер пакетів npm, систему контролю версій Git та середовище розробки, наприклад WebStorm або Visual Studio Code. Рекомендовано використовувати Node.js версії 16 або новішої, оскільки саме ця версія коректно підтримує всі використані залежності frontend та backend частин застосунку.

Перед початком роботи необхідно завантажити проєкт. Після цього розробник отримує доступ до повної структури застосунку, яка складається з двох основних директорій: backend та frontend.

Серверна частина застосунку знаходиться у папці backend. Для її запуску необхідно перейти до цієї директорії командою cd backend, після чого встановити всі необхідні залежності командою npm install. Після завершення встановлення сервер запускається командою npm run dev. У разі успішного запуску backend буде доступний за адресою http://localhost:5000. Саме ця частина відповідає за авторизацію користувачів, роботу з ролями, обробку анкет, збереження відповідей, формування статистики, рекомендацій та журналу змін.

Клієнтська частина застосунку знаходиться у папці frontend. Для її запуску потрібно відкрити новий термінал, перейти до директорії frontend командою cd frontend, встановити залежності командою npm install, а потім запустити React-застосунок командою npm start. Після успішного запуску frontend буде доступний у браузері за адресою http://localhost:3000. Саме ця частина відповідає за інтерфейс користувача, сторінки авторизації, реєстрації, проходження анкет, адміністративну панель, графіки, PDF-експорт та взаємодію з API.

Якщо запуск frontend виконується з кореневої папки проєкту, потрібно спочатку перейти до відповідної директорії командою cd frontend. Якщо запуск backend виконується з кореневої папки, потрібно перейти до серверної частини командою cd backend. У випадку помилки Could not read package.json це означає, що

команда npm install, npm start або npm run dev була виконана не в тій папці, де знаходиться відповідний файл package.json.

Основним файлом запуску backend є server.js. У ньому підключаються основні маршрути системи: authRoutes.js, surveyRoutes.js та adminRoutes.js. Файл authRoutes.js відповідає за реєстрацію та авторизацію користувачів, surveyRoutes.js — за отримання дисциплін, проходження анкет та збереження відповідей студентів, а adminRoutes.js — за адміністративний функціонал, зокрема аналітику, CRUD-операції з дисциплінами, керування анкетами, рекомендації та історію змін.

Дані системи зберігаються у папці backend/data. Файл users.json містить облікові записи користувачів, їх електронні пошти, хешовані паролі та ролі. Файл db.json містить список дисциплін, анкети, відповіді студентів і журнал адміністративних змін. При роботі з цими файлами потрібно бути обережним, оскільки некоректна структура JSON може призвести до помилок запуску або роботи API. Перед ручним редагуванням цих файлів бажано створювати резервну копію.

Для створення адміністратора вручну необхідно додати користувача у файл users.json. Пароль не можна записувати у відкритому вигляді, оскільки система використовує bcrypt-хешування. Щоб згенерувати хеш для пароля, потрібно перейти у папку backend командою cd backend і виконати команду node -e "const bcrypt=require('bcryptjs'); bcrypt.hash('парольАдміністратора',10).then(console.log)". Отримане значення потрібно вставити у поле password для користувача з роллю admin. Після цього адміністратор зможе входити в систему з поштою, вказаною у users.json, та паролем admin123.

Для frontend частини основним файлом є App.js, де налаштовано маршрутизацію між сторінками. Сторінки користувача знаходяться у папці frontend/src/pages, а компоненти адміністративної панелі — у папці frontend/src/components/admin. Файл api.js у папці frontend/src/services відповідає за налаштування Axios та автоматичне додавання JWT-токена до захищених API-запитів. Стилі застосунку знаходяться у файлі App.css.

Під час розробки важливо контролювати правильність підключення frontend до backend. У файлі frontend/src/services/api.js базова адреса API повинна мати вигляд http://localhost:5000/api. Якщо backend запускається на іншому порту, цю адресу потрібно змінити відповідно до нового значення. Якщо виникають помилки CORS, необхідно перевірити, чи у файлі server.js backend частини підключено cors через app.use(cors()).

Для встановлення додаткових frontend-бібліотек використовується команда npm install назва_бібліотеки, яку потрібно виконувати у папці frontend. Наприклад, для встановлення бібліотек генерації PDF використовується команда npm install jspdf html2canvas. Для встановлення бібліотек графіків використовується команда npm install chart.js react-chartjs-2. Для backend-залежностей команди потрібно виконувати у папці backend, наприклад npm install express cors bcryptjs jsonwebtoken dotenv.

Після внесення змін у backend сервер бажано перезапустити. Для цього потрібно зупинити процес у терміналі комбінацією Ctrl + C, після чого знову виконати команду npm run dev. Frontend зазвичай автоматично оновлюється після збереження файлів, однак у випадку помилок залежностей або стилів його також можна перезапустити командою npm start.

Резервне копіювання даних у поточній версії системи здійснюється шляхом копіювання файлів users.json та db.json. Саме ці файли містять усю основну інформацію про користувачів, дисципліни, анкети, відповіді студентів та журнал змін. Перед оновленням структури даних або масовим редагуванням дисциплін рекомендується створити копії цих файлів, щоб у разі помилки швидко відновити працездатність системи.

У майбутньому розробник може розширити систему шляхом переходу з JSON-файлів на повноцінну базу даних, наприклад PostgreSQL або MongoDB. Також можливе додавання Docker-контейнеризації, Swagger-документації для API, email-сповіщень, ролі викладача, інтеграції з Moodle або іншою LMS-платформою, автоматичних резервних копій, Excel-експорту та складнішої аналітики на основі методів машинного навчання.

Таким чином, для підтримки системи розробнику необхідно розуміти її клієнт-серверну структуру, порядок запуску frontend та backend, призначення основних файлів, принцип роботи JWT-авторизації, структуру JSON-сховища та базову логіку REST API. Дотримання цих інструкцій дозволяє швидко розгорнути застосунок, вносити зміни, тестувати новий функціонал і забезпечувати подальший розвиток програмного забезпечення.

3.5 Висновки до розділу 3

У третьому розділі було розглянуто прикладне використання розробленого програмного забезпечення для моніторингу та аналітики навчальних курсів, продемонстровано основні функціональні можливості системи, проведено комплексне тестування її працездатності, а також сформовано користувацьку та технічну документацію для подальшого впровадження і супроводу.

У підрозділі демонстрації можливостей програмного забезпечення було детально показано практичну реалізацію всіх основних функцій системи. Реалізований веб-застосунок забезпечує повноцінну роботу як для студентів, так і для адміністраторів. Для студентів доступні функції авторизації, реєстрації, перегляду дисциплін, проходження анкетування, оцінювання дисциплін за рейтинговою шкалою та залишення відкритих текстових відповідей. Для адміністраторів реалізовано розширену панель управління з модулями аналітики, CRUD-управління дисциплінами, конструктора анкет, рекомендацій системи, журналу змін та автоматичного формування PDF-звітів. Особливу цінність становить аналітичний модуль, який дозволяє не лише збирати статистичні показники, а й формувати практичні рекомендації щодо покращення якості освітнього процесу.

У процесі тестування було підтверджено коректність реалізації бізнес-логіки, стабільність роботи клієнтської та серверної частин, а також надійність системи захисту доступу. Перевірено функціонування авторизації, реєстрації, валідації

форм, одноразового проходження анкетування, побудови аналітики, роботи адміністративної панелі та генерації PDF-звітів. Проведені юніт-тести, інтеграційне, системне, UI, безпекове тестування та перевірка ролей користувачів підтвердили відповідність системи поставленим функціональним і нефункціональним вимогам. Реалізоване використання JWT-токенів, bcrypt-хешування паролів та middleware-перевірки ролей забезпечує безпечне розмежування прав доступу та захист персональних даних користувачів.

Окрему увагу було приділено підготовці програмної документації користувача, яка дозволяє студентам та адміністраторам швидко освоїти функціонал системи без необхідності додаткового технічного навчання. Інструкції охоплюють порядок авторизації, проходження анкетування, управління дисциплінами, використання аналітичних модулів, формування звітів та роботу з рекомендаціями системи. Це значно спрощує процес впровадження програмного продукту в реальне освітнє середовище університету.

Також було сформовано технічну документацію для розробника, яка містить опис архітектури системи, структури проєкту, REST API, файлового сховища, порядку запуску frontend та backend частин, принципів роботи JWT-авторизації та можливостей подальшого масштабування. Наявність такої документації забезпечує простоту передачі проєкту іншому розробнику, подальшу підтримку системи та її модернізацію без необхідності глибокого аналізу всього програмного коду.

Отримані результати підтверджують, що розроблене програмне забезпечення є повноцінним інструментом для внутрішнього моніторингу якості освітніх програм університету. Система поєднує сучасний інтерфейс, автоматизований збір даних, аналітичну обробку результатів, безпечне адміністрування та практичну придатність до реального впровадження. Це дозволяє значно підвищити ефективність управління освітнім процесом, оперативно виявляти проблемні аспекти навчальних дисциплін та приймати обґрунтовані управлінські рішення на основі реальних даних.

РОЗДІЛ 4
БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У четвертому розділі розглянуто питання безпеки життєдіяльності та основ охорони праці, пов’язані з використанням програмного забезпечення для моніторингу й аналітики навчальних курсів. Основну увагу приділено працездатності користувача та вимогам до організації робочого місця під час роботи з персональним комп’ютером [11, 12].

4.1 Працездатність користувача програмного забезпечення в умовах впливу зовнішніх факторів

Працездатність людини є одним із важливих показників її здатності виконувати певну роботу з необхідною якістю та продуктивністю протягом визначеного часу. У сфері інформаційних технологій працездатність користувача або оператора значною мірою залежить не лише від його професійних знань, а й від умов праці, рівня втоми, психологічного стану, організації робочого процесу та впливу зовнішнього середовища [11, 14].

Під час використання програмного забезпечення для моніторингу та аналітики навчальних курсів користувач виконує переважно інтелектуальну роботу. Вона пов’язана з опрацюванням інформації, введенням даних, переглядом результатів анкетування, аналізом статистичних показників, порівнянням навчальних курсів і формуванням висновків щодо якості освітнього процесу. Такий вид діяльності вимагає уважності, точності, зосередженості та здатності швидко оцінювати інформацію, подану у вигляді таблиць, графіків і діаграм.

Користувача розробленої системи можна розглядати як оператора у системі «людина — машина — середовище». У цій системі людина приймає рішення, вводить і аналізує інформацію, комп’ютер або інший пристрій виконує технічну обробку даних, а середовище створює умови, які можуть як підвищувати, так і

знижувати ефективність роботи. До таких умов належать освітлення, шум, температура повітря, якість робочого місця, стабільність електроживлення, стан мережевого з'єднання та психоемоційний фон користувача [11, 14].

Працездатність людини під час роботи з комп'ютером не є сталою величиною. Вона змінюється протягом робочого дня. На початку роботи спостерігається період входження в робочий ритм, коли користувач поступово адаптується до завдання. Далі настає період відносно сталої працездатності, коли робота виконується найбільш ефективно. Після тривалого безперервного навантаження з'являються ознаки втоми: знижується концентрація уваги, уповільнюється реакція, збільшується кількість помилок, погіршується здатність до аналізу інформації [11, 14].

Для користувачів системи моніторингу та аналітики навчальних курсів втома може проявлятися по-різному. Студенти можуть неуважно відповідати на питання анкети або пропускати окремі пункти. Викладачі можуть помилково інтерпретувати результати оцінювання курсу. Адміністративний персонал може неправильно порівняти статистичні показники або зробити неточні висновки на основі графіків. Тому підтримання працездатності користувача є важливою умовою не лише з погляду охорони праці, а й з погляду якості даних, які опрацьовує система.

Одним із основних факторів, що впливають на працездатність користувача, є тривалість безперервної роботи з екраном. Тривале зосередження на моніторі призводить до зорової втоми, сухості очей, головного болю, напруження м'язів шиї та плечового поясу. Крім того, робота з великою кількістю числової інформації створює додаткове нервово-психічне навантаження. Особливо це актуально під час перегляду аналітичних панелей, де одночасно можуть відображатися таблиці, діаграми, відсоткові показники, середні значення та порівняльні результати.

Важливим чинником є також монотонність роботи. Якщо користувач тривалий час виконує однотипні дії, наприклад заповнює форми, переглядає подібні таблиці або повторно аналізує результати опитувань, знижується рівень уваги. Монотонність призводить до того, що людина починає автоматично

виконувати дії, не завжди контролюючи їх правильність. Для зменшення цього впливу доцільно чергувати різні види роботи: введення даних, перегляд результатів, аналіз графіків, формування висновків та короткі перерви.

На працездатність користувача впливають і зовнішні фактори, які не завжди залежать від нього безпосередньо. У сучасних умовах освітній процес часто відбувається не лише в аудиторіях університету, а й дистанційно. Користувач може працювати з дому, гуртожитку, бібліотеки або іншого приміщення. Такі умови не завжди відповідають ергономічним і санітарно-гігієнічним вимогам. Наприклад, робота може виконуватися при недостатньому освітленні, на незручному столі, у приміщенні з підвищеним шумом або за нестабільного інтернет-з’єднання.

Окремо слід враховувати зовнішні небезпечні фактори, характерні для сучасного освітнього середовища в Україні. До них належать повітряні тривоги, необхідність тимчасового переходу в укриття, перебої електропостачання, нестабільність мережевого з’єднання та підвищене психоемоційне навантаження. Ці фактори не є частиною самого програмного забезпечення, однак вони впливають на умови його використання. У таких ситуаціях користувач повинен мати можливість швидко припинити роботу, зберегти важливі дані та продовжити роботу після нормалізації умов [14, 15].

З огляду на це, програмне забезпечення має бути спроєктоване так, щоб не створювати додаткового ризику втрати інформації у випадку раптового переривання роботи. Доцільним є використання механізмів автоматичного або періодичного збереження даних, підтвердження важливих дій, коректної обробки помилок і можливості повторного входу до системи. Якщо користувач заповнює анкету або працює з налаштуваннями курсу, система повинна мінімізувати ризик втрати вже введеної інформації.

Перебої електропостачання також можуть негативно впливати на працездатність та безпеку користувача. Раптове вимкнення комп’ютера спричиняє переривання роботи, втрату концентрації, можливу втрату незбережених даних і додаткове психологічне напруження. Для зниження такого ризику доцільно використовувати ноутбуки із зарядженою батареєю або джерела безперебійного

живлення для стаціонарних робочих місць. На рівні організації освітнього процесу важливо передбачати можливість продовження роботи у зручний для користувача час.

Нестабільне інтернет-з’єднання є ще одним фактором, що впливає на роботу веб-орієнтованих систем. Якщо з’єднання переривається під час надсилання анкети або формування звіту, користувач може не розуміти, чи була дія виконана успішно. Це створює додаткове напруження і може призвести до повторного введення даних або помилок. Тому інтерфейс системи повинен надавати зрозумілі повідомлення про стан операції: успішне збереження, помилку з’єднання, необхідність повторити дію або відновити сторінку.

Психоемоційне навантаження є одним із найважливіших факторів, що впливають на працездатність людини. Постійне напруження, тривожність, перевантаження інформацією та необхідність швидко реагувати на зовнішні події можуть знижувати здатність користувача до уважної роботи. У таких умовах особливо важливим є простий і зрозумілий інтерфейс програмного забезпечення. Користувач повинен легко орієнтуватися в системі, розуміти призначення кнопок, бачити результат виконаних дій і не витрачати зайвий час на пошук потрібної функції.

Для підтримання працездатності користувача доцільно застосовувати організаційні та програмні заходи. До організаційних заходів належать раціональне планування роботи, уникнення тривалих безперервних сеансів, перерви для відпочинку, чергування видів діяльності та робота в безпечних умовах. До програмних заходів належать зрозуміла структура інтерфейсу, логічна навігація, помірне використання кольорів, достатній розмір шрифтів, наочне подання інформації та запобігання перевантаженню екрана зайвими елементами.

Під час розробки системи моніторингу та аналітики навчальних курсів важливо враховувати, що користувачі можуть мати різний рівень цифрової підготовки. Для одних робота з аналітичними панелями є звичною, для інших — складною. Тому інтерфейс має бути інтуїтивно зрозумілим, а основні дії —

простими й послідовними. Це знижує нервово-психічне навантаження і сприяє збереженню працездатності.

Особливе значення має візуальна організація інформації. Аналітичні дані повинні подаватися так, щоб користувач міг швидко зрозуміти їх зміст. Для цього доцільно використовувати логічне групування показників, підписи до графіків, зрозумілі одиниці вимірювання, однаковий стиль оформлення таблиць і діаграм. Якщо інформація подана хаотично, користувач витрачає більше часу на її сприйняття, швидше втомлюється і частіше допускає помилки.

Таким чином, працездатність користувача програмного забезпечення залежить від поєднання фізичних, психологічних, організаційних та інформаційних факторів. Для розробленої системи моніторингу та аналітики навчальних курсів важливо забезпечити такі умови використання, за яких користувач може ефективно працювати з даними, не зазнаючи надмірного зорового, м’язового чи нервово-психічного навантаження. Урахування зовнішніх факторів, можливість переривання та відновлення роботи, зрозумілий інтерфейс, стабільне збереження даних і раціональна організація робочого процесу сприяють підвищенню безпеки, надійності та ефективності використання програмного продукту.

4.2 Ергономічні та гігієнічні вимоги до організації робочого місця користувача персонального комп’ютера

Ергономічна та гігієнічна організація робочого місця користувача персонального комп’ютера є важливою умовою безпечної та продуктивної роботи з програмним забезпеченням. Оскільки розроблена система моніторингу та аналітики навчальних курсів передбачає роботу з електронними формами, таблицями, графіками, текстовими блоками та звітами, користувач тривалий час взаємодіє з екранним пристроєм, клавіатурою та мишею. Неправильна організація робочого місця може призвести до швидкої втоми, зниження уваги, погіршення зору, болю у спині, шиї, плечах і кистях рук [12, 13].

Робоче місце користувача персонального комп’ютера повинно забезпечувати зручну робочу позу, достатній простір для рухів, правильне розміщення обладнання та належні параметри виробничого середовища [12, 14]. До основних елементів такого робочого місця належать робочий стіл, крісло, монітор або екран ноутбука, клавіатура, миша, джерела освітлення та допоміжні засоби. Усі ці елементи повинні бути розташовані так, щоб користувач міг працювати без надмірного напруження м’язів і органів зору.

Важливим елементом робочого місця є робочий стіл. Його висота повинна відповідати антропометричним параметрам користувача і забезпечувати природне положення рук під час роботи [12, 13]. Для більшості користувачів під час роботи сидячи зручною є висота столу приблизно 720–750 мм. Однак оптимальна висота залежить від зросту людини, висоти крісла, положення клавіатури та миші. Під час роботи передпліччя мають розташовуватися приблизно горизонтально, лікті — бути зігнутими близько під прямим кутом, а плечі — залишатися розслабленими.

Якщо стіл занадто високий, користувач змушений піднімати плечі та напружувати м’язи шиї й плечового поясу. Якщо стіл занадто низький, користувач нахиляється вперед, що збільшує навантаження на хребет. Тому висота столу або висота крісла повинні регулюватися так, щоб користувач міг зберігати природну робочу позу. Для людей високого або низького зросту особливо важливо індивідуально налаштовувати робоче місце, а за потреби використовувати підставку для ніг або регульований стіл.

Робоча поверхня столу повинна мати достатню площу для розміщення монітора, клавіатури, миші, документів і додаткового обладнання. На столі не повинно бути зайвих предметів, які заважають рухам або змушують користувача працювати в незручному положенні. Під столом необхідно залишати вільний простір для ніг, щоб користувач міг змінювати положення тіла протягом роботи. Обмеження простору під столом призводить до статичного напруження і погіршення кровообігу.

Крісло користувача повинно бути стійким, зручним і бажано регульованим за висотою. Сидіння має забезпечувати рівномірний розподіл навантаження, а спинка

— підтримувати поперековий відділ хребта. Оптимальне положення тіла передбачає, що стопи повністю спираються на підлогу або підставку, коліна зігнуті приблизно під прямим кутом, спина підтримується спинкою крісла, а плечі залишаються розслабленими. Неправильне положення тіла під час тривалої роботи може спричинити біль у спині, порушення постави та загальну втому.

Якщо користувач працює з ноутбуком, виникає додаткова ергономічна проблема: екран і клавіатура з’єднані в одному пристрої. Якщо ноутбук стоїть низько на столі, користувач змушений нахиляти голову вперед. Якщо ж ноутбук підняти на рівень очей, стає незручно користуватися вбудованою клавіатурою. Тому для тривалої роботи доцільно використовувати підставку для ноутбука, окрему клавіатуру та мишу. Це дозволяє розмістити екран на зручній висоті й одночасно зберегти правильне положення рук.

Монітор або екран ноутбука повинен розташовуватися прямо перед користувачем. Рекомендована відстань від очей до екрана становить приблизно 50–70 см. Верхня межа екрана має бути на рівні очей або трохи нижче. Таке розміщення дозволяє уникнути постійного нахилу голови та зменшує навантаження на шийний відділ хребта. Екран не повинен бути розташований під кутом, який змушує користувача постійно повертати голову або тулуб.

Під час роботи з аналітичними панелями, таблицями та графіками велике значення має якість зображення. Інформація повинна бути чіткою, контрастною та достатньо великою для комфортного читання. Дрібний шрифт, низький контраст або надмірна кількість елементів на екрані збільшують зорове навантаження. У розробленій системі доцільно використовувати достатній розмір шрифтів, чіткі підписи до графіків, логічне групування блоків інформації та помірну кількість кольорів.

Клавіатура повинна розташовуватися перед користувачем на такій відстані, щоб руки могли вільно лежати на робочій поверхні без напруження. Під час введення тексту кисті не повинні бути надмірно зігнуті вгору або вниз. Миша повинна розміщуватися поруч із клавіатурою на одному рівні з нею. Якщо миша розташована занадто далеко, користувач змушений тягнутися до неї, що створює

навантаження на плече та передпліччя. Для тривалої роботи доцільно використовувати зручну мишу, яка відповідає розміру кисті користувача.

Освітлення робочого місця є одним із ключових санітарно-гігієнічних факторів. Воно повинно бути достатнім, рівномірним і не створювати засліплювальної дії [13, 14]. Найкращим є поєднання природного та штучного освітлення. Робоче місце бажано розташовувати боком до вікна, щоб пряме сонячне світло не потрапляло в очі користувача і не створювало відблисків на екрані. Якщо вікно розташоване перед користувачем, яскраве світло може засліплювати очі. Якщо вікно позаду користувача, на екрані можуть з’являтися відблиски.

У вечірній час або при недостатньому природному освітленні потрібно використовувати штучне освітлення. Настільна лампа має бути розташована так, щоб світло падало збоку і не відбивалося від екрана. Недостатнє освітлення змушує очі напружуватися, а надмірно яскраве світло викликає дискомфорт і швидку втому. Яскравість екрана повинна відповідати освітленості приміщення: у темному приміщенні екран не повинен бути надто яскравим, а при яскравому зовнішньому світлі — надто тьмяним.

Важливим фактором є відсутність відблисків на екрані. Відблиски виникають через неправильне розташування монітора відносно вікон, ламп або інших джерел світла. Вони погіршують видимість інформації та змушують користувача напружувати зір або змінювати положення тіла. Для усунення відблисків потрібно змінити положення монітора, використовувати жалюзі або штори, правильно розміщувати джерела штучного освітлення.

Мікроклімат приміщення також впливає на працездатність користувача. Температура повітря має бути комфортною, без різких перепадів. Приміщення потрібно регулярно провітрювати. Недостатня вентиляція призводить до сонливості, зниження концентрації уваги та швидкої втоми. Надмірна сухість повітря може спричиняти подразнення очей, що особливо помітно під час тривалої роботи з екраном. У робочому приміщенні потрібно підтримувати чистоту, уникати надмірного шуму й забезпечувати достатній простір для переміщення.

Режим праці та відпочинку є необхідною умовою профілактики перевтоми під час тривалої роботи з персональним комп'ютером [12, 14]. Слід робити регулярні короткі перерви. Доцільно після кожних 45–60 хвилин роботи робити перерву тривалістю 5–10 хвилин. Під час перерви варто змінити положення тіла, пройтися, виконати прості вправи для очей, шиї, плечей і кистей рук. Для профілактики зорової втоми корисно періодично переводити погляд з екрана на віддалені предмети.

Для користувачів розробленої системи особливо важливо дотримуватися режиму праці під час роботи з аналітичними даними. Аналіз результатів опитувань, порівняння показників і формування висновків потребують високої концентрації уваги. Якщо користувач працює без перерв, зростає ризик помилкової інтерпретації даних. Тому складні аналітичні дії доцільно виконувати у періоди найвищої працездатності, а після тривалої роботи з графіками й таблицями робити короткі перерви.

Правильна організація робочого місця також позитивно впливає на якість роботи з програмним забезпеченням. Якщо користувач сидить зручно, добре бачить екран, не відчуває болю в спині чи напруження очей, він швидше й точніше виконує робочі дії. Це особливо важливо для адміністративного персоналу, який працює зі звітами та статистикою, а також для викладачів, які аналізують результати оцінювання навчальних курсів.

Таким чином, ергономічні та гігієнічні вимоги до організації робочого місця користувача персонального комп'ютера мають безпосередній вплив на безпеку, здоров'я та продуктивність праці. Для ефективного використання програмного забезпечення для моніторингу та аналітики навчальних курсів необхідно забезпечити правильну висоту столу, зручне крісло, раціональне розміщення монітора, клавіатури та миші, достатнє освітлення, належний мікроклімат і регулярні перерви. Дотримання цих вимог дозволяє зменшити навантаження на зір, опорно-руховий апарат і нервову систему, а також забезпечити комфортну та безпечну роботу користувача.

4.4 Висновки до розділу 4

У четвертому розділі було розглянуто питання безпеки життєдіяльності та основ охорони праці, пов’язані з використанням програмного забезпечення для моніторингу та аналітики навчальних курсів. Визначено, що робота з такою системою належить переважно до розумової праці користувача персонального комп’ютера та потребує належної організації робочого процесу.

У підрозділі 4.1 проаналізовано працездатність користувача програмного забезпечення в умовах впливу зовнішніх факторів. Встановлено, що на ефективність роботи впливають тривалість безперервної взаємодії з екраном, монотонність, інформаційне навантаження, психоемоційний стан, стабільність електропостачання та інтернет-з’єднання. Для підтримання працездатності доцільно передбачити зручний інтерфейс, можливість збереження даних, коректну обробку помилок, перерви в роботі та раціональне планування складних аналітичних дій.

У підрозділі 4.2 розглянуто ергономічні та гігієнічні вимоги до організації робочого місця користувача персонального комп’ютера. Встановлено, що висота столу, положення крісла, розміщення монітора, якість освітлення, мікроклімат, положення клавіатури й миші та режим праці й відпочинку безпосередньо впливають на здоров’я, комфорт і продуктивність користувача. Дотримання цих вимог дозволяє зменшити зорове, м’язове та нервово-психічне навантаження під час роботи з розробленою системою.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто актуальну проблему підвищення якості освітнього процесу курсів шляхом автоматизації моніторингу та аналітики навчальних курсів. В умовах сучасної цифровізації освітнього середовища особливо важливим є своєчасне отримання об’єктивного зворотного зв’язку від студентів щодо якості викладання дисциплін, структури курсів, практичної складової навчання та загального рівня задоволеності освітнім процесом. Саме тому розробка програмного забезпечення, яке дозволяє систематизувати цей процес, є важливим практичним завданням для сучасного університету.

У першому розділі було проведено аналіз предметної області, досліджено сучасні підходи до оцінювання якості освітніх програм, методи збору та обробки студентського зворотного зв’язку, а також існуючі програмні рішення у сфері освітньої аналітики. Було визначено основні недоліки традиційних підходів до анкетування, зокрема фрагментарність збору інформації, складність ручної обробки результатів, відсутність оперативної аналітики та низький рівень гнучкості існуючих систем. На основі проведеного аналізу було обґрунтовано доцільність створення власного програмного рішення, орієнтованого на комплексний моніторинг якості навчальних курсів.

У другому розділі було виконано проєктування програмного забезпечення, сформульовано функціональні та нефункціональні вимоги до системи, визначено основні ролі користувачів та логіку їхньої взаємодії із застосунком. Було розроблено архітектуру клієнт-серверної системи, обрано технологічний стек на основі React для frontend частини та Node.js з Express.js для backend частини. Окрему увагу приділено моделюванню структури даних, механізмам авторизації, розмежуванню прав доступу, побудові аналітичного модуля та організації файлового JSON-сховища. Також було спроєктовано логіку роботи адміністративної панелі, конструктора анкет, системи рекомендацій та журналу змін.

У третьому розділі було реалізовано прикладне використання розробленого програмного забезпечення та продемонстровано практичну роботу всіх основних модулів системи. Було створено повноцінний веб-застосунок, який забезпечує авторизацію та реєстрацію користувачів, проходження анкетування студентами, автоматичне блокування повторного проходження опитувань, аналітичну обробку результатів, CRUD-управління дисциплінами, динамічне управління анкетами, автоматичне формування рекомендацій, журнал адміністративних змін та генерацію PDF-звітів. Проведене тестування підтвердило стабільність роботи системи, коректність реалізації бізнес-логіки та безпечне розмежування доступу між студентами та адміністраторами. Додатково було сформовано користувацьку та технічну документацію, що забезпечує готовність програмного продукту до впровадження у реальному освітньому середовищі.

У четвертому розділі було сформовано основи БЖД та охорони праці щодо коректного підходу до використання системи, правильності підготовки робочого місця до роботи.

У результаті виконання кваліфікаційної роботи поставлена мета була досягнута — розроблено та впроваджено програмне забезпечення для моніторингу та аналітики навчальних курсів, яке дозволяє автоматизувати процес збору, обробки та аналізу зворотного зв’язку від студентів. Реалізована система забезпечує підвищення ефективності внутрішнього контролю якості освіти, скорочення часу на обробку результатів анкетування, покращення прозорості адміністративних процесів та підтримку прийняття управлінських рішень на основі реальних аналітичних показників.

Практична цінність розробленого програмного забезпечення полягає у можливості його безпосереднього використання для організації внутрішнього моніторингу освітніх програм, підготовки звітності для кафедр, деканатів та акредитаційних процедур. Гнучка архітектура системи дозволяє легко адаптувати її до потреб різних освітніх програм, факультетів та закладів вищої освіти.

Перспективами подальшого розвитку системи є інтеграція з повноцінними системами управління базами даних, зокрема PostgreSQL, впровадження Docker-

контейнеризації, розширення ролей користувачів, інтеграція з LMS-платформами типу Moodle, реалізація email-сповіщень, автоматичних резервних копій та використання методів машинного навчання для побудови прогнозної аналітики якості освітнього процесу.

Таким чином, розроблене програмне забезпечення є сучасним, функціональним та практично значущим інструментом цифровізації внутрішньої системи забезпечення якості освіти, що відповідає сучасним вимогам розвитку університетського середовища та може бути основою для подальшого масштабування й впровадження на рівні закладу вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Siemens G. Learning Analytics: The Emergence of a Discipline // American Behavioral Scientist. 2013. Vol. 57, No. 10. P. 1380–1400.
2. Lang C., Siemens G., Wise A. F., Gašević D., Merceron A. Handbook of Learning Analytics. 2nd ed. Vancouver : Society for Learning Analytics Research, 2022.
3. Clow D. The Learning Analytics Cycle: Closing the Loop Effectively // Proceedings of the 2nd International Conference on Learning Analytics and Knowledge. 2012. P. 134–138.
4. Romero C., Ventura S. Educational Data Mining and Learning Analytics: An Updated Survey. 2024.
5. Blei D. M., Ng A. Y., Jordan M. I. Latent Dirichlet Allocation // Journal of Machine Learning Research. 2003. Vol. 3. P. 993–1022.
6. React Documentation. Quick Start. React official documentation.
7. Node.js Documentation. Node.js official API documentation.
8. Express.js. Node.js web application framework. Official documentation.
9. Chart.js Documentation. Getting Started. Official documentation.
10. Qualtrics XM for Education. Student & Staff Experience Management for Education.
11. Атаманчук П. С. Безпека життєдіяльності : навч. посіб. Київ : Центр учбової літератури, 2020. 276 с.
12. Жидецький В. Ц. Охорона праці користувачів комп’ютерів : підручник. Львів : Афіша, 2020. 176 с.
13. Андрейчук Н. І. Охорона праці : навч. посіб. / Н. І. Андрейчук, Ю. В. Кіт, С. В. Шибанов, О. В. Шерстньова. Львів : Видавництво Львівська політехніка, 2021. 276 с.
14. Безпека життєдіяльності та охорона праці : підруч. / В. В. Сокуренко, О. М. Бандурка та ін. Харків : ХНУВС, 2021. 308 с.

15. Желібо Є. П. Безпека життєдіяльності : підручник / В. В. Зацарний. Київ : Каравела, 2023. 344 с.