Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна система контролю навчального процесу з дисциплін комп'ютерної інженерії»

08-54.БКР.001.00.000 ПЗ

Виконала: студентка 4 курсу, групи 1КІ-21б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма – «Комп'ютерна інженерія»
Багрій Д. В.

Керівник к.т.н., доц. каф. ОТ
Крупельницький Л.В.

Рецензент к.т.н., доц. каф. ПЗ
Ракитянська Г.Б.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«17» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень бакалвр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Багрій Дарії Володимирівні

1 Тема роботи: «Інформаційна система контролю навчального процесу з дисциплін комп'ютерної інженерії», керівник роботи Крупельницький Леонід Віталійович, к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «20» березня 2025 року №97.

2 Строк подання студентом роботи 13.05.2025.

3 Вихідні дані до роботи: інформаційна система контролю навчального процесу з дисциплін комп'ютерної інженерії, технології розробки: Next.js для клієнтської частини, NestJS для серверної логіки, MongoDB як система керування базами даних, бібліотека Mantine для створення адаптивного вебінтерфейсу, формат взаємодії — REST API, цільова платформа — веббраузер, середовище розробки — Visual Studio Code, структура даних представлена у вигляді моделей, функціональні модулі: реєстрація та авторизація викладача, перегляд предметів і груп, таблиці студентів, оцінювання за видами робіт, відображення загального балу, сторінка розкладу.

4 Зміст розрахунково-пояснювальної записки: вступ, огляд та аналіз інформаційних систем контролю навчального процесу, аналіз існуючих підходів до моніторингу прогресу у навчанні, формування вимог до функціоналу

проєктування архітектури та моделі даних системи, реалізація клієнтської частини вебзастосунку, реалізація серверної частини та взаємодія з базою даних, інструкція користувача для роботи з системою, тестування функціональності, висновки.

5 Перелік графічного матеріалу: функціональна схема інтерфейсу вебзастосунку.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	к.т.н., доц. каф. ОТ Крупельницький Л.В.	21.03.2025	13.06.2025
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.2025	30.05.2025

7. Дата видачі завдання 21.03.2025 р.

8. Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Під
1	Постановка задачі роботи	21.03.25	
2	Пошук матеріалів та інформаційних джерел	21.03.25—30.03.25	
3	Аналіз інформаційних систем контролю навчального процесу та визначення вимог до розробки	21.03.25—30.03.25	
4	Обгрунтування архітектури, структури інтерфейсу та моделі даних системи	31.03.25—13.04.25	
5	Розробка клієнтської частини вебзастосунку	07.04.25 – 20.04.25	
6	Розробка клієнтської частини вебзастосунку	13 .04.25 – 27.04.25	
7	Тестування функціональності системи і аналіз результатів тестування	28.04.25—05.05.25	
8	Оформлення пояснювальної записки та ілюстративного матеріалу. Перевірка якості виконання	06.05.25 – 13.05.25	

Студент Дарія Багрій

Керівник роботи к.т.н., доц. Леонід КРУПЕЛЬНИЦЬКИЙ

АНОТАЦІЯ

УДК 004.4:378.147

Багрій Д. В. Інформаційна система контролю навчального процесу з дисциплін комп’ютерної інженерії. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна інженерія, освітня програма — Комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 50 ст.

На укр. мові. Бібліогр.: 15 назв; рис.: 10; табл.: 2.

У даній бакалаврській роботі основною задачею є розробка вебзастосунку для викладачів з метою автоматизації контролю навчального процесу. Система дозволяє реєструватися, обирати дисципліни, переглядати групи, список студентів та їх оцінки, а також відстежувати розклад.

Актуальність роботи обумовлена потребою викладачів у швидкому та зручному інструменті взаємодії з навчальними групами, дисциплінами та успішністю студентів. У роботі розглянуто існуючі рішення, обґрунтовано вибір технологій, описано структуру системи, основні модулі, інтерфейс, а також проведено тестування розробленого програмного забезпечення.

ABSTRACT

UDC 004.4:378.147

Bagriy D. V. Information system for controlling the educational process in computer engineering disciplines. Bachelor's qualification work in specialty 123 - Computer Engineering, educational program - Computer Engineering. Vinnytsia: VNTU, 2025. 50 p.

In Ukrainian. Bibliography: 15 titles; fig.: 10; tab.: 2.

In this bachelor's thesis, the main task is to develop a web application for teachers to automate the control of the educational process. The system allows you to register, select disciplines, view groups, a list of students and their grades, and track the schedule.

The relevance of the work is due to the need of teachers for a quick and convenient tool for interacting with study groups, disciplines and student performance. The work considers existing solutions, justifies the choice of technologies, describes the system structure, main modules, interface, and tests the developed software.

ЗМІСТ

ВСТУП

У сучасному світі швидкого розвитку цифрових технологій навчальні заклади дедалі активніше впроваджують інформаційні системи для автоматизації управління освітнім процесом, що дозволяє оптимізувати рутинні завдання та підвищувати якість освіти. Однією з ключових складових таких систем є забезпечення ефективної комунікації між викладачем і студентами, збереження, обробка та швидкий доступ до інформації про навчальні дисципліни, оцінювання, розклад занять, а також облік успішності. Такі системи сприяють зменшенню адміністративного навантаження на викладачів, дозволяючи їм зосередитися на педагогічній діяльності. Розробка спеціалізованих вебзастосунків для викладачів є важливим кроком у напрямку цифровізації освіти, оскільки вони забезпечують зручність, гнучкість і адаптивність до реальних потреб навчального процесу. Впровадження подібних рішень також сприяє підвищенню прозорості та доступності інформації для всіх учасників освітнього процесу.

Актуальність теми зумовлена зростаючою потребою у зручних і функціональних інструментах для автоматизації роботи викладачів. Незважаючи на існування систем, таких як JetIQ, Google Classroom чи Moodle, багато викладачів стикаються з проблемами, пов’язаними з незручним інтерфейсом, обмеженим функціоналом, складністю налаштування або недостатньою адаптацією до специфічних вимог викладацької діяльності в українських закладах вищої освіти. Наприклад, деякі системи не дозволяють гнучко керувати розкладом чи швидко оновлювати оцінки, що ускладнює оперативне реагування на потреби студентів. Крім того, більшість популярних платформ не мають повної української локалізації або потребують додаткових ресурсів для інтеграції в локальні освітні процеси. Вебзастосунок, розроблений спеціально для викладачів, забезпечує інтуїтивно зрозумілий інтерфейс, швидкий доступ до необхідних даних і можливість адаптації до індивідуальних вимог, що сприяє підвищенню ефективності освітнього процесу, зменшенню рутинних операцій і покращенню якості ведення обліку успішності студентів.

Метою дослідження є розробка сучасного вебзастосунку для автоматизації контролю навчального процесу викладачами, який би забезпечував зручне керування навчальними дисциплінами, оцінками, групами студентів і розкладом занять, а також підтримував українську локалізацію для спрощення використання в українських вишах. Такий застосунок має на меті спростити щоденні завдання викладачів, забезпечити швидкий доступ до актуальної інформації та підвищити ефективність взаємодії зі студентами. Важливим аспектом є створення системи, яка буде простою у використанні навіть для користувачів із мінімальним досвідом роботи з інформаційними технологіями, але водночас матиме достатній функціонал для виконання всіх необхідних завдань. Розробка такого рішення також сприятиме цифровізації освіти, що є одним із пріоритетів сучасної освітньої політики.

Об’єктом дослідження є інформаційна система управління навчальним процесом зі сторони викладача у закладі вищої освіти, яка охоплює процеси організації, моніторингу та обліку навчальної діяльності. Ця система включає збереження даних про дисципліни, студентів, оцінки, розклад, а також комунікацію між викладачем і навчальними групами. Вона є основою для автоматизації рутинних операцій, таких як виставлення оцінок чи складання розкладу, і спрямована на підвищення ефективності роботи викладачів. Дослідження зосереджене на створенні програмного забезпечення, яке інтегрує всі ці аспекти в єдину зручну платформу, адаптовану до потреб вищої освіти.

Предметом дослідження є сучасні фреймворки, технології та методи розробки вебзастосунків, які використовуються для створення інформаційних систем управління навчальним процесом. Зокрема, досліджуються можливості таких інструментів, як React, TypeScript, Node.js, а також бази даних MongoDB для забезпечення швидкої обробки даних, безпеки та адаптивності інтерфейсу. Аналізується також застосування сучасних підходів до веброзробки, таких як модульна архітектура, API-інтеграція та DevOps-інструменти для розгортання, що дозволяють створювати надійні та масштабовані системи.

Задачі дослідження спрямовані на комплексне вирішення проблеми автоматизації контролю навчального процесу. Основні завдання включають: проведення аналізу існуючих систем управління навчальним процесом для виявлення їхніх недоліків і обмежень; визначення функціональних і нефункціональних вимог до нової системи, які б відповідали реальним потребам викладачів; обґрунтування вибору оптимальних інструментів і технологій для реалізації вебзастосунку, враховуючи їхню продуктивність, простоту інтеграції та підтримку локалізації; розробку архітектури системи, включаючи клієнтську та серверну частини, дизайн адаптивного інтерфейсу та структуру бази даних для ефективного зберігання інформації; реалізацію вебзастосунку з функціоналом реєстрації, авторизації, управління дисциплінами, перегляду груп, оцінок студентів і розкладу занять; тестування системи для перевірки її функціональності, стабільності, адаптивності та зручності використання, а також оцінку її ефективності в умовах реального навчального процесу.

Розробка такого вебзастосунку має значний потенціал для вдосконалення освітнього процесу, оскільки дозволяє викладачам економити час на адміністративних завданнях, таких як облік оцінок чи складання звітів, і зосередитися на викладанні. Система забезпечує централізований доступ до всіх необхідних даних, що підвищує прозорість і контроль за навчальним процесом. Крім того, адаптивний інтерфейс і українська локалізація роблять застосунок доступним для широкого кола користувачів, включаючи викладачів із різним рівнем цифрових навичок. У перспективі система може бути розширена додатковими функціями, такими як інтеграція з електронними журналами, аналітика успішності студентів чи автоматизоване сповіщення про розклад, що зробить її ще більш універсальною для використання в закладах вищої освіти.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сучасний стан питання

Сучасні тенденції розвитку інформаційного суспільства суттєво впливають на всі сфери людської діяльності, зокрема на освіту, яка зазнає активної цифрової трансформації. Навчальні заклади по всьому світу впроваджують електронні платформи та сервіси, що оптимізують управління навчальним процесом, підвищують його ефективність і забезпечують доступність освітніх ресурсів. Одним із ключових інструментів цієї трансформації є інформаційні системи контролю та підтримки освітньої діяльності, які стали невід'ємною частиною сучасної вищої освіти.

Такі системи широко використовуються в Україні та за кордоном, надаючи викладачам можливість вести електронні журнали, виставляти оцінки, формувати звіти, контролювати відвідуваність, переглядати розклади, додавати навчальні матеріали та виконувати інші функції. З початком пандемії COVID-19, коли дистанційне та змішане навчання стали нормою, потреба в таких рішеннях зросла в рази. Вони не лише полегшують адміністрування навчального процесу, а й забезпечують гнучкість у взаємодії між викладачами, студентами та адміністрацією. Однак зростання попиту також висвітлило низку проблем, пов'язаних із зручністю та ефективністю використання цих систем.

Аналіз практичного досвіду використання популярних освітніх платформ показує, що вони не завжди відповідають потребам кінцевих користувачів, зокрема викладачів. Часто такі системи мають надмірно широкий функціонал, орієнтований на різні категорії користувачів: адміністраторів, студентів, кураторів, деканати тощо. Це призводить до ускладнення інтерфейсу, коли викладачам доводиться витрачати значний час на навігацію, пошук потрібної інформації чи виконання рутинних операцій. Наприклад, для виставлення оцінок може знадобитися кілька переходів між вкладками, що знижує продуктивність і викликає незадоволення.

Ще однією проблемою є недостатня адаптація систем до сучасних технологій, зокрема до мобільних пристроїв. Багато платформ не мають адаптивного дизайну, через що їхній вебінтерфейс погано масштабується на смартфонах чи планшетах. Елементи управління можуть бути недоступними, текст — важко читабельним, а функціонал — обмеженим без повноцінного екрана комп'ютера. Такий стан речей суттєво знижує мобільність і оперативність роботи викладачів, які все частіше потребують доступу до системи в будь-який час і з будь-якого пристрою. Сучасний користувач очікує швидкого відгуку інтерфейсу, мінімалістичного дизайну, інтуїтивної логіки роботи та доступності 24/7, але багато платформ не відповідають цим стандартам.

Особливу увагу варто приділити зручності відображення прогресу студентів. У більшості класичних систем дані про успішність представлені у вигляді таблиць без аналітичного групування чи візуалізації. Викладачу складно швидко оцінити динаміку навчання, порівняти результати студентів чи проаналізувати засвоєння конкретної теми. Наприклад, відсутність графіків прогресу чи порівняльних звітів змушує викладачів вести паралельний облік у сторонніх інструментах, таких як Excel або Google-таблиці. Це суперечить ідеї централізованої цифрової платформи, яка має об'єднувати всі функції в одному середовищі.

Іншою проблемою є недостатня персоналізація функціоналу для конкретних ролей. Більшість систем намагаються охопити всіх учасників освітнього процесу, що робить їх громіздкими для викладачів, чиї потреби зосереджені на контролі знань, оцінюванні та звітності. Наприклад, викладачу рідко потрібні функції управління фінансами чи розподілу аудиторій, але вони часто присутні в інтерфейсі, ускладнюючи роботу. Це вказує на потребу в профільованих рішеннях, які б фокусувалися на завданнях окремої категорії користувачів.

У контексті цих викликів постає потреба в розробці цільових інформаційних систем, адаптованих до потреб викладачів, які працюють зі студентами в межах навчального курсу. Такі системи повинні мінімізувати зайвий функціонал,

спрощувати навігацію та забезпечувати інтуїтивну взаємодію. Наприклад, вебзастосунок, орієнтований на викладача, може включати лише інструменти для управління оцінками, перегляду прогресу, додавання матеріалів і формування звітів, виключаючи функції, що стосуються інших ролей. Це дозволить знизити когнітивне навантаження, підвищити продуктивність і зробити цифрову взаємодію більш ефективною.

Крім того, сучасні системи мають враховувати вимоги до кросплатформної доступності. Адаптивний дизайн, оптимізований для мобільних пристроїв, забезпечує викладачам гнучкість у роботі, дозволяючи перевіряти оцінки чи завантажувати матеріали навіть поза офісом. Інтеграція з хмарними технологіями може додатково підвищити доступність, забезпечуючи синхронізацію даних у реальному часі. Наприклад, викладач може почати роботу на комп’ютері в аудиторії, а продовжити з телефону дорогою додому.

Таким чином, сучасний стан освітніх інформаційних систем вказує на необхідність створення профільованих рішень, які враховують специфіку роботи викладачів. Подальше дослідження зосереджується на аналізі існуючих платформ, виявленні їхніх недоліків і розробці вебзастосунку, що відповідає потребам викладача. Такий застосунок має бути зручним, інтуїтивним, адаптивним і орієнтованим на ключові завдання: контроль знань, оцінювання та моніторинг прогресу студентів.

1.2 Аналіз існуючих систем

На сучасному етапі цифрової трансформації вищої освіти інформаційні системи управління навчальним процесом стали ключовими інструментами в роботі викладача. Їхнє використання забезпечує доступ до навчальної інформації, автоматизацію обліку успішності, формування звітів, доступ до розкладу занять і комунікації з іншими учасниками освітнього процесу. [9] Проте досвід використання

таких систем показує, що їхня ефективність часто обмежена у контексті щоденної роботи викладача.

Прикладом такої системи є JetIQ — інформаційна платформа, яка розроблена у Вінницькому національному технічному університеті (ВНТУ) та впроваджена в освітній процес для забезпечення зручного доступу до даних про навчальну діяльність. У низці інших вищих навчальних закладів також використовуються подібні за структурою локальні системи, створені на внутрішньому рівні університетів або адаптовані під їх потреби. JetIQ у цьому контексті є характерним прикладом такого підходу до автоматизації освітнього процесу.

Система JetIQ виконує функцію централізованої бази даних, у якій зберігається інформація про студентів, викладачів, дисципліни, академічні групи, оцінювання, розклад, відвідуваність тощо. Вона дійсно охоплює широке коло завдань і позиціонується як комплексне рішення для адміністрування освітнього процесу.

Проте, попри свою масштабність, JetIQ часто не відповідає критеріям зручності й швидкодії, необхідних у повсякденній практиці викладача. Користувачі відзначають перевантажений інтерфейс, де інформація подається у складній, фрагментованій формі, а доступ до конкретних функцій потребує багаторівневої навігації. Це ускладнює виконання базових дій, таких як виставлення оцінок, перегляд складу групи, фіксація відвідуваності чи робота з розкладом. [11] Часто для отримання потрібної інформації потрібно витратити значну кількість часу — у тому числі на пошук у різних вкладках і розділах системи.

Ще однією проблемою JetIQ є обмежена мобільність. Система оптимізована переважно під роботу на персональному комп’ютері, тоді як сучасні користувачі, зокрема викладачі, все частіше використовують планшети, ноутбуки, мобільні пристрої. Через відсутність адаптивного інтерфейсу можливість оперативного доступу до даних значно знижується. Це обмежує гнучкість роботи викладача і знижує ефективність сприйняття оперативних рішень у процесі викладання.

Ще одним прикладом цифрової освітньої платформи, що використовується здебільшого в загальноосвітньому секторі та для дистанційного навчання, є Google Classroom. На відміну від JetIQ, цей сервіс фокусується на організації навчального процесу у форматі курсів. Google Classroom має простий, зручний інтерфейс, дозволяє створювати завдання, вести базовий облік, прикріплювати файли, коментувати роботи учнів та студентів, організовувати відеозустрічі.

Проте, функціонал Google Classroom є обмеженим у контексті потреб вищої технічної освіти. Система не дозволяє реалізувати складну структуру обліку оцінювання, як-от 100-бальна шкала, ECTS або 12-бальна шкала з коефіцієнтами. Крім того, відсутні можливості для збереження навчального плану, комплексного управління групами, гнучкого формування звітів, ведення журналу за модульною системою. Таким чином, попри простоту та загальну доступність, Google Classroom є лише частковим рішенням і не покриває повний спектр задач, які стоять перед викладачем технічного закладі вищої освіти.

Крім розглянутих прикладів, у деяких університетах впроваджуються власні локальні інформаційні системи, побудовані на внутрішніх розробках або відкритих фреймворках. [15] Але незалежно від технічної реалізації, такі платформи часто мають спільні недоліки: надмірну складність у користуванні, відсутність адаптивного дизайну, нестачу налаштувань саме під потреби викладача. Частина таких рішень є орієнтованими на адміністративний або навчально-методичний персонал і не враховує специфіку повсякденного викладацького навантаження.

У результаті можна зробити висновок, що жодна з наявних систем не дозволяє викладачу швидко і зручно реалізувати стандартні задачі: переглянути список групи, внести оцінки, перевірити відвідуваність, побачити розклад занять або динаміку виконання завдань студентами. Саме в таких ситуаціях викладачі часто звертаються до ручних засобів — табличних редакторів, Google-форм, особистих записників — що знижує ефективність і нівелює суть цифрового середовища. Нижче продемонстровано порівня аналогів в таблиці 1.1.

Таблиця 1.1 — Порівняння аналогів

Характеристика	**JetIQ**	**Google Classroom**	**Розроблений вебзастосунок**
Зручність інтерфейсу	Перевантажений, складна навігація	Простий, інтуїтивний	Інтуїтивний, мінімалістичний
Адаптивність до мобільних пристроїв	Обмежена	Повна	Повна
Підтримка складних шкал оцінювання	Частково	Відсутня	Повна
Гнучкість у звітності	Обмежена	Обмежена	Є можливість формування зведень
Підтримка навчального плану	Так	Ні	Так
Модулі, орієнтовані на викладача	Частково	Частково	Повністю
Аудиторія	Студенти, адміністрація, викладачі	Студенти, викладачі	Лише викладачі
Наявність непотрібного функціоналу	Так	Частково	Немає

Тому виникає нагальна потреба у розробці веборієнтованого інтерфейсу, який буде адаптовано до потреб лише однієї ролі — викладача. Такий інтерфейс повинен мати мінімалістичний і зрозумілий дизайн, містити тільки найнеобхідніші модулі: дисципліни, групи, оцінки, розклад, та бути зручним для використання як на ПК, так і на мобільних пристроях. Основна цінність таких рішень — оперативність, зручність

і простота, що у підсумку підвищує ефективність викладацької діяльності та якість взаємодії з навчальним процесом.

1.3 Моніторинг прогресу навчання

Системи моніторингу прогресу навчання відіграють важливу роль у сучасному освітньому процесі, оскільки дають змогу здійснювати об’єктивний та оперативний контроль знань і навичок студентів. Вони дозволяють не лише фіксувати поточні оцінки, а й аналізувати динаміку засвоєння матеріалу, виявляти слабкі сторони у знаннях, адаптувати навчальні підходи та своєчасно приймати рішення щодо корекції викладання.

На практиці такі системи зазвичай реалізуються у вигляді модулів, вбудованих у загальні навчальні платформи, або як окремі сервіси. Незалежно від реалізації, головною метою залишається можливість збирати, обробляти та інтерпретувати інформацію про результати навчання в реальному часі. [12]

Інформаційна система, яку було розроблено в межах даної дипломної роботи, передбачає саме такий функціонал: викладач має змогу бачити загальну картину прогресу студентів у межах дисципліни, а також переходити до деталізованого перегляду результатів конкретної особи. Передбачена можливість аналізу даних за періодами, заняттями, видами оцінювання (практичні роботи, модулі, колоквіуми тощо).

Також важливою складовою є візуалізація даних. У межах реалізованого інтерфейсу викладач має доступ до зручної таблиці з оцінками, згрупованими за предметами та групами. У майбутньому така таблиця може бути доповнена діаграмами прогресу — графіками, що ілюструють динаміку засвоєння матеріалу кожним студентом. [8] Це дозволяє швидше сприймати інформацію та приймати рішення щодо повторного вивчення певних тем або застосування індивідуального підходу.

Окрему цінність становить функція звітності. Навіть у мінімальному вигляді, система має дозволяти формувати зведені таблиці для аналізу успішності групи загалом, виводити середні бали, мінімальні/максимальні значення, частку відвідуваності або пропущених занять. Такі зведення можуть бути використані як для внутрішнього контролю, так і для звітності на кафедрі, у деканаті або при подачі на академічні рейтинги.

Крім того, важливо враховувати ще одну функцію — самомоніторинг студентом. Хоча вебзастосунок орієнтовано на викладача, потенційне підключення студентського кабінету дозволяє кожному студенту бачити свій навчальний прогрес, орієнтуватися в рівні засвоєння матеріалу та своєчасно вживати заходів для його покращення. Це особливо актуально в умовах гнучкого навчання, коли накопичення балів протягом семестру має вирішальне значення.

Використання сучасної аналітики в системах моніторингу дозволяє автоматично генерувати оцінки, визначати середні показники, помічати тенденції. Такі системи можуть не лише зберігати оцінки, а й інтерпретувати їх. Наприклад, студент, який постійно отримує оцінки нижчі за середні, може автоматично потрапити в зону ризику, а система може вивести викладачеві сповіщення про потребу у втручанні.

Також аналітичні модулі можуть визначати так звані “проблемні теми” — ті, які найчастіше викликають складнощі у студентів. У майбутньому це дасть змогу системі рекомендувати перегляд окремих тем, а викладачу — адаптувати навчальну програму. Навіть базова статистика, зібрана з поточних оцінок, дозволяє більш об’єктивно оцінити реальний рівень знань.

У системах тестування прогрес може оцінюватися за допомогою кількісного аналізу — частоти правильних відповідей, часу на виконання, повторного звернення до тем. Також часто застосовується поведінковий аналіз — фіксується, скільки разів користувач відкривав ту чи іншу тему, які завдання переглядав, з якими складнощами стикався. [12] Ці принципи можуть бути реалізовані і в системах навчального

контролю, орієнтованих на викладача — наприклад, через вивід тенденцій щодо окремої групи чи загального успіху по дисципліні.

Розроблений у межах дипломної проєкт вебзастосунок дозволяє реалізувати базовий рівень моніторингу з потенціалом до подальшого розширення. У майбутньому можливе доповнення системи елементами адаптивного навчання — наприклад, матеріалами курсу, підказками щодо тем, на які варто звернути увагу студенту, або автоматичним коригуванням структури курсу відповідно до реальних досягнень. Такий підхід поєднує функції викладача, аналітика й адміністратора, що в сукупності створює ефективне, сучасне цифрове середовище навчання.

Таким чином, моніторинг прогресу навчання — це не лише облік оцінок. Це комплексна система збору, обробки та представлення даних, яка дає змогу викладачу бачити реальний стан знань, вчасно реагувати на труднощі, будувати індивідуальні траєкторії для студентів і підвищувати загальну якість навчального процесу.

1.4 Визначення вимог до моніторингу прогресу вивчення матеріалу

Моніторинг прогресу вивчення матеріалу є важливою функціональною складовою інформаційної системи, призначеної для контролю навчального процесу. Саме через моніторинг викладач може оцінити рівень засвоєння знань студентами, виявити слабкі місця у навчанні, своєчасно відреагувати на зниження результатів, а також об’єктивно сформувати підсумкові оцінки.

Інформаційна система, орієнтована на потреби викладача, повинна не лише зберігати та відображати оцінки, а й реалізовувати ефективну аналітику результатів. Це передбачає врахування певних вимог до модуля моніторингу, який забезпечить повноцінну та зручну роботу з навчальними даними. [13]

Перш за все, моніторинг повинен бути об'єктивним. Інформація про прогрес студентів має ґрунтуватися виключно на фактичних результатах навчальної діяльності. Це означає, що усі дії — оцінювання, формування підсумків — повинні

мати цифрове підтвердження і не залежати від суб’єктивних оцінок або візуального аналізу.

Важливою є постійність моніторингу. Система повинна забезпечувати безперервне відстеження змін у результатах студентів протягом усього навчального періоду. Наприклад, кожне заняття має фіксуватись із можливістю додавання оцінки, коментаря або відмітки про відсутність. Такий підхід дозволяє формувати повну картину навчальної активності студента.

Не менш важливою характеристикою є точність. Вебсистема повинна надавати викладачу точні числові значення результатів, з можливістю подальшої фільтрації, сортування та аналітики. Наприклад, викладач повинен мати змогу дізнатися середній бал студента за дисципліною, кількість незданих завдань або середній рівень успішності всієї групи.

Аналітика також є обов’язковим компонентом модуля моніторингу. Система повинна вміти аналізувати введені дані та формувати висновки, які викладач зможе використовувати для адаптації викладання. До прикладу, якщо велика частина студентів не впоралася з певною темою, це може бути сигналом про необхідність її повторного пояснення або зміни підходу до оцінювання.

У контексті роботи з великими групами важливим є і автоматизований характер моніторингу. Викладач не повинен витрачати значний час на ручний аналіз оцінок або формування звітів. Система повинна автоматично генерувати зведення, підсумкові таблиці, графіки успішності або попередження про ризики — наприклад, якщо студент не має жодної оцінки понад два тижні.

Ще однією ключовою вимогою є надійність. Система повинна надійно зберігати усі дані про навчальні результати, унеможливлюючи їх втрату або зміну без авторизації. Особливо важливо передбачити резервне копіювання інформації, захист бази даних та обмеження прав доступу. [15] Оцінки мають бути зафіксовані так, щоб не виникало сумнівів у їхній достовірності.

Крім технічних вимог, варто враховувати також мотиваційний аспект. У майбутньому можливо додати у систему, яка надає викладачу повну інформацію про навчальний прогрес, має сприяти не лише контролю, а й формуванню позитивного зворотного зв’язку зі студентами. Наприклад, за підсумками роботи система може автоматично генерувати короткі рекомендації: хто потребує додаткової підтримки, хто працює стабільно, а хто демонструє високі результати. Це дозволяє викладачу краще мотивувати студентів, надавати індивідуальні зауваження або підбадьорити тих, хто має позитивну динаміку.

Таким чином, система моніторингу прогресу вивчення матеріалу в межах інформаційної системи для викладача повинна відповідати таким узагальненим вимогам: бути об'єктивною, постійною, точною, аналітичною, автоматизованою, надійною і мотиваційною. Лише поєднання цих характеристик дозволить реалізувати повноцінний модуль моніторингу, який не лише зберігатиме та виводитиме дані, а й стане дієвим інструментом у процесі прийняття педагогічних рішень.

1.5 Психологічні аспекти моніторингу навчання

Моніторинг прогресу навчання є не лише технічним процесом, але й має значний психологічний вплив на студентів, викладачів і освітній процес загалом. Правильно організований моніторинг може підвищувати мотивацію, сприяти саморефлексії та допомагати студентам досягати кращих результатів. Однак некоректний підхід до оцінювання може викликати стрес, знижувати впевненість і навіть провокувати уникнення навчання. У цьому підрозділі розглядаються ключові психологічні аспекти моніторингу, включаючи вплив оцінювання на мотивацію, емоційний стан студентів і стратегії адаптивного оцінювання.

Вплив оцінювання на мотивацію студентів, мотивація є одним із центральних факторів успішного навчання. Згідно з теорією самодетермінації, мотивація може бути внутрішньою (пов’язаною з інтересом до предмета) або зовнішньою (зумовленою оцінками, нагородами чи покараннями). Моніторинг навчання, що

базується на регулярному оцінюванні, часто виступає зовнішнім мотиватором. Наприклад, чітке відображення прогресу у вебсистемі, де студенти бачать свої оцінки та сумарний бал, може стимулювати їх до активнішої роботи. Однак надмірний акцент на оцінках може пригнічувати внутрішню мотивацію, особливо якщо студенти сприймають оцінювання як контроль, а не як інструмент розвитку.

Дослідження показують, що формативне оцінювання (оцінювання для навчання) ефективніше сприяє мотивації, ніж сумативне (оцінювання результатів). У вебсистемі, яка дозволяє викладачам надавати зворотний зв’язок разом із оцінками, студенти отримують не лише числову оцінку, а й розуміння своїх сильних і слабких сторін. Наприклад, коментарі до виконаних завдань у системі можуть допомогти студентам зосередитися на покращенні, а не лише на отриманні високого балу. Це сприяє розвитку відчуття компетентності, що є ключовим елементом внутрішньої мотивації.

Емоційний вплив моніторингу навчання може викликати широкий спектр емоцій у студентів, від задоволення до тривоги. Регулярне відстеження прогресу, особливо у прозорій системі, де оцінки доступні в реальному часі, може створювати відчуття контролю над навчальним процесом. Наприклад, можливість бачити, як окремі оцінки впливають на загальний бал, дозволяє студентам планувати свою роботу та відчувати прогрес. Це позитивно впливає на їхню впевненість і знижує невизначеність.

Однак для деяких студентів постійний моніторинг може викликати стрес, особливо якщо вони отримують низькі оцінки або порівнюють себе з іншими. Теорія соціального порівняння (Festinger, 1954) пояснює, чому студенти можуть відчувати тиск, якщо система відкрито відображає рейтинги чи результати одногрупників. Щоб мінімізувати негативний вплив, вебсистема повинна уникати публічного порівняння результатів і надавати персоналізований зворотний зв’язок. Наприклад, замість рейтингової таблиці доцільно показувати лише індивідуальний прогрес студента у вигляді графіку чи діаграми.

Стрес також може бути пов’язаний із частотою оцінювання. Надто часті перевірки знань створюють відчуття постійного контролю, що може призводити до вигорання. У системі варто передбачити гнучкий графік оцінювання, де викладач може налаштовувати частоту завдань залежно від потреб групи. Крім того, важливо забезпечити можливість перегляду оцінок у зручний для студента час, щоб уникнути тиску через несподівані повідомлення про результати.

Стратегією адаптивного оцінювання є ефективним інструментом для врахування психологічних особливостей студентів. Воно передбачає налаштування завдань і критеріїв оцінювання залежно від рівня підготовки, стилю навчання та індивідуальних потреб. У контексті вебсистеми адаптивність може бути реалізована через кілька підходів. По-перше, система може пропонувати завдання різного рівня складності, дозволяючи студентам обирати ті, що відповідають їхнім можливостям. Наприклад, після виконання базового завдання студент може отримати доступ до складнішого, що мотивує до розвитку.

По-друге, адаптивне оцінювання включає персоналізований зворотний зв’язок. У вебсистемі викладач може налаштувати автоматичні повідомлення, які не лише вказують на помилки, а й пропонують ресурси для їх виправлення (наприклад, посилання на навчальні матеріали). Це допомагає студентам сприймати оцінки як частину навчального процесу, а не як остаточний вирок. Дослідження показують, що зворотний зв’язок, орієнтований на завдання та процес, значно підвищує ефективність навчання.

По-третє, адаптивність передбачає врахування психологічного стану студентів. Наприклад, система може надавати опцію “відкласти оцінювання” для студентів, які повідомляють про тимчасові труднощі (стрес, хвороба). Такий підхід демонструє турботу про добробут студентів і сприяє створенню підтримуючого навчального середовища.

Роль викладача у психологічному аспекті моніторингу, психологічний вплив моніторингу залежить не лише від технічних можливостей системи, а й від підходу

викладача. Викладач виступає медіатором між системою та студентами, інтерпретуючи оцінки та надаючи зворотний зв’язок. У вебсистемі доцільно передбачити інструменти для спрощення цієї ролі, наприклад, шаблони коментарів до оцінок або автоматичне формування звітів про прогрес студента. Це дозволяє викладачу зосередитися на індивідуальній підтримці, а не на рутинних операціях.

Викладачі також повинні бути обізнані з психологічними аспектами оцінювання. Наприклад, уникнення надмірної критики та акцент на досягненнях можуть підвищувати мотивацію студентів. Система може включати рекомендації для викладачів, такі як використання позитивної мови у зворотному зв’язку або періодичне проведення бесід про прогрес.

1.6 Аналіз ризиків і обмежень при розробці системи

Розробка інформаційної системи для моніторингу прогресу навчання викладачами пов’язана з низкою ризиків і обмежень, які можуть вплинути на якість, терміни реалізації та ефективність кінцевого продукту. Ці ризики охоплюють технічні, організаційні, користувацькі та зовнішні аспекти, а обмеження стосуються доступних ресурсів, технологічного стеку та нормативних вимог. Своєчасне виявлення та аналіз таких факторів дозволяють розробити стратегії їхньої мінімізації, що забезпечує стабільність і надійність системи. У цьому підрозділі розглянуто основні ризики та обмеження, а також заходи, спрямовані на їх подолання, щоб гарантувати успішне впровадження системи.

Одним із ключових технічних ризиків є можливі збої в роботі серверної інфраструктури або бази даних, що може призвести до втрати даних або тимчасової недоступності системи. Наприклад, MongoDB, обрана як основна база даних, чутлива до неправильної конфігурації кластера, що може спричинити зниження продуктивності при високому навантаженні. Для зменшення цього ризику передбачено регулярне резервне копіювання за допомогою інструменту mongodump із зберіганням копій у віддаленому сховищі протягом 30 днів. Крім того,

використовується моніторинг продуктивності через Prometheus і Grafana, що дозволяє виявляти аномалії, такі як перевантаження сервера, і оперативно реагувати. Ще одним технічним ризиком є помилки в коді, які можуть спричинити некоректну обробку запитів або вразливості. Для їх мінімізації застосовується автоматизоване тестування через Jest для серверної частини (NestJS) і Cypress для клієнтської (Next.js), а також код-рев'ю перед розгортанням через CI/CD-пайплайн.

Користувацькі ризики пов'язані з низьким рівнем цифрових навичок у деяких викладачів, що може ускладнити адаптацію до системи. Наприклад, старші викладачі можуть відчувати труднощі з навігацією в вебінтерфейсі або введенням оцінок через адаптивний дизайн, який потребує базового розуміння роботи зі смартфонами чи планшетами. Для вирішення цієї проблеми розроблено інтуїтивно зрозумілий інтерфейс із мінімалістичним дизайном, використовуючи бібліотеку Mantine для уніфікованих компонентів. Додатково передбачено створення детальної інструкції користувача з покроковим описом основних функцій, таких як авторизація, перегляд розкладу та виставлення оцінок. Планується також проведення тренінгів для викладачів перед впровадженням системи, щоб підвищити їхню впевненість у роботі з платформою. Іншим користувацьким ризиком є можливе небажання викладачів переходити з традиційних методів (наприклад, паперових журналів) на цифрову систему. Для подолання цього бар'єру система демонструє чіткі переваги, такі як економія часу на підрахунок балів і автоматичне формування звітів, що підкреслюється в інструкціях і презентаціях.

Безпекові ризики є особливо критичними, оскільки система обробляє персональні дані, такі як ПІБ, email і оцінки студентів. Несанкціонований доступ або витік даних може порушити довіру до системи та призвести до юридичних наслідків.

Обмеження масштабування є ще одним важливим аспектом, оскільки зростання кількості користувачів (наприклад, при розширенні системи на весь університет) може призвести до уповільнення обробки запитів. MongoDB підтримує горизонтальне масштабування через шардинг, але це вимагає додаткових ресурсів і складної

конфігурації. На початковому етапі система розрахована на 100-200 одночасних користувачів, що відповідає потребам одного факультету. Для забезпечення продуктивності застосовано індексацію в MongoDB і пагінацію даних у клієнтській частині. Проте обмеженням є залежність від обраного технологічного стеку (Next.js, NestJS, MongoDB), який може ускладнити інтеграцію з іншими системами університету, наприклад, із застарілими платформами на PHP. Для вирішення цього передбачено API на основі REST, що забезпечує гнучкість для майбутніх інтеграцій, але потребує додаткового тестування сумісності.

Організаційні ризики включають обмежені терміни розробки та бюджет, що може вплинути на якість реалізації. Наприклад, недостатнє фінансування обмежує можливість залучення додаткових розробників або використання платних сервісів, таких як хмарні бази даних MongoDB Atlas. Для компенсації використано відкриті технології з активною підтримкою спільноти, а розробка ведеться з чітким плануванням через Agile-методологію, що дозволяє розподіляти завдання на спринти. Ще одним організаційним ризиком є можливі зміни вимог з боку замовника (університету) під час розробки, що може затримати реліз. Для цього на етапі аналізу проведено детальні консультації з викладачами та адміністрацією, а вимоги задокументовано в технічному завданні.

Наприклад, обробка персональних даних вимагає згоди користувачів і можливості видалення їхніх даних, що реалізовано через функцію деактивації облікового запису. Проте це ускладнює розробку, оскільки потрібна додаткова логіка для безпечного видалення зв’язаних даних (оцінок, логів). У разі розширення системи за межі України необхідно враховувати GDPR, що вимагатиме додаткових ресурсів для імплементації. Для мінімізації цього ризику політика конфіденційності чітко описує обробку даних, а всі дії з даними логуються для підзвітності.

Зовнішні ризики, такі як відключення електроенергії чи проблеми з інтернет-з’єднанням, можуть ускладнити доступ до системи, особливо в умовах воєнного стану в Україні. Для часткового вирішення передбачено офлайн-кэшування даних у браузері

через Next.js, що дозволяє переглядати збережені розклади чи оцінки без підключення. Однак повноцінна офлайн-робота обмежена через клієнт-серверну архітектуру. У майбутньому планується розробка прогресивного вебдодатку (PWA), що розширить офлайн-функціонал.

Ці ризики та обмеження враховані на етапі проєктування, що дозволяє звести до мінімуму їхній вплив. Застосування сучасних технологій, автоматизованого тестування, інтуїтивного інтерфейсу та чіткого планування забезпечує стабільність розробки та готовність системи до впровадження. Подальший розвиток передбачає моніторинг нових ризиків і вдосконалення стратегій їхньої мінімізації.

2 РОЗРОБКА СТРУКТУРИ ВЕБСИСТЕМИ

2.1. Обґрунтування вибору технологій

Проєктування та розробка інформаційної системи контролю навчального процесу потребують відповідального підходу до вибору інструментів реалізації. Враховуючи вимоги до продуктивності, гнучкості, швидкодії та простоти масштабування, було прийнято рішення використати сучасний технологічний стек, який поєднує в собі ефективні рішення як на стороні клієнта, так і на стороні сервера.

Для реалізації клієнтської частини системи (інтерфейсу для викладача) обрано Next.js — популярний фреймворк для розробки вебдодатків на основі бібліотеки React. Серед основних переваг Next.js слід відзначити:

- підтримку серверного рендерингу (SSR), що покращує швидкість завантаження сторінок та SEO;
- зручну маршрутизацію без необхідності сторонніх бібліотек;
- високий рівень інтеграції з сучасними інструментами для оптимізації продуктивності (наприклад, автоматичне розбиття коду);
- підтримку TypeScript, що забезпечує кращу підтримку масштабованості та контроль типів.

Next.js дозволив реалізувати інтерфейс, який є водночас динамічним, інтуїтивним та зручним для користувача. Адаптивна верстка забезпечує коректне відображення на різних пристроях — від десктопів до мобільних телефонів, що є критичним для сучасних вебрішень.

Серверну частину системи реалізовано за допомогою NestJS — прогресивного фреймворку на базі Node.js, який базується на архітектурі модулів та принципах об’єктно-орієнтованого програмування. NestJS надає зручну структуру проєкту, підтримку ін’єкції залежностей, декоратори, а також сумісність з TypeScript, що забезпечує високий рівень типізації та зручність у роботі з великим кодовим базисом.

На відміну від простих реалізацій на Express.js, фреймворк NestJS дозволяє швидше масштабувати застосунок, чітко розділяти логіку контролерів, сервісів та репозиторіїв, а також інтегрувати зовнішні модулі, такі як бібліотеки аутентифікації або валідації. Ці переваги особливо важливі в контексті розробки системи, яка потенційно може розширюватися в майбутньому: додаватися підтримка нових ролей користувачів (наприклад, студенти, адміністратори), нові модулі (опитування, звітність, графіки тощо), або зміна структури бази даних.

У якості сховища даних було обрано MongoDB — документно-орієнтовану NoSQL базу даних, що дозволяє зберігати інформацію у вигляді гнучких структур JSON-подібних документів. На відміну від традиційних реляційних баз даних, MongoDB не потребує суворого дотримання схем і дозволяє швидко змінювати структуру документів без складної міграції даних. Це особливо актуально в проєктах, що активно змінюються у процесі розробки.

MongoDB добре підходить для зберігання об'єктів, які мають складну або змінну структуру — наприклад, групи, студенти, оцінки і тд.. Крім того, вона забезпечує високу продуктивність, легку інтеграцію з NestJS через Mongoose або інші обгортки, а також має добре розвинену екосистему, що значно спрощує адміністрування.

Для розробки інтерфейсу користувача було обрано бібліотеку компонентів Mantine. Вона дозволяє швидко створювати зручні, адаптивні, а головне — стилістично узгоджені інтерфейси. Mantine забезпечує набір компонентів (таблиці, форми, модальні вікна, повідомлення, кнопки), які легко налаштовуються, мають вбудовану підтримку темної теми, кастомізації та адаптивного дизайну. Завдяки цьому розробка UI була не лише швидкою, а й відповідала сучасним вимогам зручності користування.

У сукупності обраний технологічний стек забезпечив реалізацію вебзастосунку, який є продуктивним, масштабованим, безпечним і зручним для користувача. Такий підхід дозволяє не тільки ефективно реалізувати поставлені завдання, а й закласти основу для подальшого розвитку системи. Усі технології є відкритими, активно

підтримуються спільнотою, мають гарну документацію, що дозволяє легко залучати нових розробників до проєкту в разі необхідності.

Таким чином, вибір Next.js, NestJS, MongoDB та Mantine як основних інструментів для реалізації інформаційної системи контролю навчального процесу повністю обґрунтований як з позиції технічної ефективності, так і з погляду зручності для кінцевого користувача — викладача.

2.2. Архітектура системи

Інформаційна система контролю навчального процесу повинна мати чітку, логічну та модульну архітектуру, що забезпечує надійність, масштабованість, простоту супроводу та зручність для кінцевого користувача — викладача. Архітектура такої системи реалізована за принципом розділення відповідальностей між клієнтською (front-end) і серверною (back-end) частинами, а також базою даних, яка виступає центральним сховищем усієї інформації.

Клієнтська частина (front-end) реалізована за допомогою фреймворку Next.js, який дозволяє будувати динамічні, інтуїтивно зрозумілі сторінки з підтримкою маршрутизації, компонуванням за допомогою React-компонентів та використанням сучасних бібліотек стилізації. Інтерфейс складається з окремих логічних розділів:

— головна панель навігації;

— перелік дисциплін, закріплених за викладачем;

— таблиця студентів конкретної групи;

— модуль виставлення та перегляду оцінок;

— перегляд розкладу.

Фронтенд реалізує логіку взаємодії з користувачем — відправляє запити до серверу, приймає відповіді та відображає інформацію у зручному вигляді. Для забезпечення візуальної цілісності та зручності було використано бібліотеку Mantine, яка надає адаптивні компоненти інтерфейсу та підтримує темізацію застосунку.

Серверна частина (back-end), побудована з використанням NestJS, відповідає за обробку запитів, реалізацію логіки роботи з даними, а також взаємодію з базою даних. Основними компонентами серверної частини є:

— контролери, які приймають HTTP-запити;

— сервіси, що реалізують логіку обробки даних;

— репозиторії для роботи з базою даних;

— механізми авторизації та автентифікації;

— модулі для валідації, помилок та обробки відповіді.

Архітектура NestJS передбачає модульну побудову, завдяки чому кожна частина системи (наприклад, модуль користувачів, модуль дисциплін, модуль оцінок) існує незалежно, зберігаючи при цьому можливість взаємодії з іншими частинами проєкту. Така структура дозволяє швидко вносити зміни, розширювати функціонал або здійснювати рефакторинг без порушення загальної цілісності коду.

База даних реалізована на основі MongoDB. Вона містить основні сутності системи: користувачі (викладачі), навчальні дисципліни, академічні групи, студенти, оцінки, розклад занять.

Усі ці сутності зберігаються у вигляді документів з гнучкою структурою. MongoDB дозволяє зручно зберігати вкладені об’єкти, масиви даних, додаткові параметри — без потреби у складному описі реляцій. Це забезпечує гнучкість при зміні структури даних, що особливо актуально у процесі активної розробки.

Серверна частина взаємодіє з базою даних за допомогою Mongoose — популярної обгортки для MongoDB, що дозволяє визначати схеми, валідувати дані, створювати запити та обробляти результати у зручному вигляді.

Комунікація між клієнтською та серверною частинами відбувається через HTTP-запити з використанням REST-архітектури. Наприклад, при відкритті сторінки з групою система виконує запит до маршруту, а у відповідь отримує список студентів.

Аналогічно обробляються запити на отримання оцінок, зміни даних або додавання нових записів.

Система також реалізує базову автентифікацію — викладач має змогу увійти до свого облікового запису та отримати доступ лише до тих дисциплін і груп, які йому призначені. Механізм аутентифікації базується на JSON Web Token (JWT), що дозволяє підтримувати захищені сесії без збереження паролів у запитах.

Усі компоненти системи об’єднані у логічно цілісну архітектуру, де кожен рівень виконує свою функцію. Такий підхід забезпечує масштабованість, зручність супроводу та можливість розширення функціоналу в майбутньому. Наприклад, у подальших версіях системи можна легко реалізувати додаткові ролі (наприклад, декан, студенти), модулі аналітики, звітності або навіть мобільну версію застосунку.

Отож, архітектура інформаційної системи розроблена з урахуванням сучасних підходів до розподіленої веброзробки, забезпечує ефективну взаємодію між складовими та відповідає реальним потребам викладача у контролі навчального процесу.

2.3. Розробка інтерфейсу та бази даних

Розробка вебінтерфейсу та побудова структури бази даних є ключовими складовими при створенні інформаційної системи контролю навчального процесу. Саме через інтерфейс викладач взаємодіє з системою, виконує необхідні дії, переглядає навчальні дані, тоді як база даних забезпечує збереження інформації, її структуризацію та подальшу обробку.

Інтерфейс користувача. Головною метою при створенні інтерфейсу було забезпечення максимальної зручності для викладача, логічності розміщення елементів і швидкої доступності основних функцій. Розробка реалізована на основі фреймворку Next.js з використанням Mantine — бібліотеки інтерфейсних компонентів, яка дозволяє створювати адаптивні, естетично привабливі та легкі у використанні UI-рішення.

Загальна структура інтерфейсу включає наступні основні екрани:

- Реєстрації — для початку викладач реєструється, email, ім’я(ПІБ), вказує номер телефону, пароль, а також предмет, який викладатиме.
- Вхід в систему — авторизація. Після введення email та пароля, система виконує перевірку облікових даних і перенаправляє до основного кабінету користувача.
- Профіль — після входу в систему викладач бачить свою особисту інформацію.
- Групи — при виборі певної дисципліни викладач має можливість перегляду та вибору групи. На цій сторінці відображається список студентів, їх ПІБ, email та сумарний бал.
- Оцінювання — реалізоване у вигляді таблиці, де кожен рядок відповідає студенту, а кожна колонка — види контролю знань. Інтерфейс підтримує введення оцінок, редагування, збереження результатів.
- Розклад — викладач має змогу переглянути розклад занять.
- Вихід — вихід з системи.

Усі сторінки є адаптивними, тобто коректно відображаються на широкоформатних екранах. Завдяки використанню компонентів Mantine було реалізовано зручні форми, таблиці, сповіщення, модальні вікна та інші елементи, що підвищують загальну зручність користування.

Інтерфейс дотримується мінімалістичного підходу: відсутні зайві блоки, непотрібна інформація прихована або згрупована у вкладки, доступ до головних функцій не перевищує двох кліків. Такий підхід сприяє швидкому навчанню користувача та мінімізує помилки під час взаємодії з системою.

Структура бази даних. Базу даних побудовано на основі MongoDB. Усі дані зберігаються у вигляді JSON-подібних документів у відповідних колекціях. Основні сутності системи:

— Викладачі — зберігаються дані про зареєстрованих користувачів системи. Для кожного викладача вказується email, пароль (у зашифрованому вигляді), ПІБ, список дисциплін, до яких він прикріплений.
— Студенти — кожен студент має власний запис із ПІБ, унікальним ідентифікатором, приналежністю до академічної групи, а також списком оцінок, які відносяться до певних робіт.
— Групи — академічні групи, які об’єднують студентів. Кожна група має свою назву, факультет, список студентів та унікальним ідентифікатором.
— Дисципліни — навчальні предмети, які викладає певний викладач для конкретної групи. Кожна дисципліна містить назву, код, а також зв’язки з викладачем і групою.
— Оцінки — для кожного студента зберігається інформація про отримані бали з певної роботи.
— Розклад — містить дату, час, дисципліну, групу та викладача, якому призначено заняття.

Завдяки гнучкій структурі MongoDB можливо зберігати вкладені структури — наприклад, у документі студента можна одразу розміщувати масив оцінок або зберігати теми дисципліни без потреби у складній схемі зовнішніх ключів. Це суттєво пришвидшує роботу з даними, особливо у вебзастосунках, де важлива швидка відповідь сервера.

Модель взаємозв’язку між сутностями відповідає типовій структурі: один викладач може мати кілька дисциплін; кожна дисципліна прив’язана до однієї групи; у групі — множина студентів; кожен студент має множину оцінок по кожній дисципліні Рисунок 2.1.

Рисунок 2.1 – Схема взаємозв’язків основних сутностей бази даних вебсистеми

Для взаємодії з базою використовується бібліотека Mongoose, яка дозволяє створювати схеми, валідувати поля, виконувати складні запити, а також будувати зв’язки між сутностями без втрати продуктивності.

Таким чином, база даних виступає надійним ядром усієї системи, забезпечуючи зберігання, доступність та цілісність інформації про студентів, оцінки, дисципліни та розклад. Її структура адаптована під потреби викладача і дозволяє зручно працювати з навчальними даними у щоденній практиці.

2.4. Моделі даних та взаємодія компонентів

У процесі розробки інформаційної системи контролю навчального процесу важливим етапом стало моделювання даних і організація взаємодії між основними компонентами системи. Вебзастосунок побудований за класичною клієнт-серверною архітектурою, де клієнтська частина відповідає за інтерфейс і зручність взаємодії,

серверна — за логіку обробки запитів, а база даних — за зберігання структурованої інформації.

Вся інформація в системі зберігається у базі даних MongoDB. Це документно-орієнтована система, яка дозволяє зберігати дані у вигляді гнучких об’єктів з довільною структурою. Робота з базою здійснюється через бібліотеку Mongoose, яка надає можливість описувати структуру документів, встановлювати обов’язкові поля, типи даних, а також реалізовувати прості й складні зв’язки між об’єктами.

Для кожного логічного об’єкта в системі створена окрема модель. Модель викладача включає унікальний ідентифікатор, прізвище, ім’я, по батькові, електронну пошту, номер телефону, зашифрований пароль, а також список предметів, до яких він має доступ. Модель предмета містить назву, унікальний код, посилання на викладача, до якого він належить, і зв’язок із навчальною групою. Студенти мають власні облікові записи, які включають ідентифікатор, ПІБ, email, номер у списку, а також структуру з оцінками. Оцінки представлені як вкладений об’єкт у межах студента і містять лише числові значення по кожному з типів роботи: практична 1, практична 2, колоквіум, практична 3, модуль та сумарний бал. Крім цього, реалізовано модель розкладу, яка включає дані про дату, час, предмет і групу.

Передача даних між клієнтською та серверною частинами здійснюється за допомогою HTTP-запитів у форматі REST API. Фронтенд, реалізований з використанням Next.js, надсилає запити на сервер, побудований на фреймворку NestJS. У відповідь сервер обробляє запит, звертається до бази даних, отримує потрібну інформацію та надсилає її клієнту у форматі JSON. Наприклад, при відкритті сторінки з оцінками, коритувач відправляє запит із параметрами предмета та групи, у відповідь система повертає об’єкти студентів та відповідні бали. При зміні оцінки викладач вводить нове значення, і клієнтська частина формує запит на оновлення даних. На сервері виконується перевірка, після чого нове значення зберігається у базі даних.

Усі дії користувача, які стосуються закритих розділів системи, вимагають проходження авторизації. Для цього реалізовано механізм автентифікації за допомогою токенів. Після входу викладач отримує унікальний токен доступу, який зберігається на стороні клієнта та додається до кожного запиту. На сервері токен перевіряється, що дозволяє обмежити доступ до даних лише тими викладачами, які мають на це відповідні права. Таким чином, система гарантує захист інформації та конфіденційність оцінок.

Структура взаємодії між компонентами є логічно впорядкованою. Кожен рівень виконує власну функцію: інтерфейс — збір і передавання даних, сервер — логіка та обробка, база — надійне зберігання. Усі компоненти працюють узгоджено та ефективно, забезпечуючи надійність і стабільність функціонування вебзастосунку. Такий підхід дозволяє розширювати систему в майбутньому без необхідності кардинальних змін у логіці або структурі даних.

Загалом уся система створювалася з урахуванням потреб користувачів, тому в процесі роботи особлива увага приділялася деталям. Це дозволило досягти цілісності в реалізації та забезпечити зручність на кожному етапі взаємодії з інтерфейсом.

2.5 Оптимізація продуктивності системи

Оптимізація продуктивності вебсистеми для моніторингу навчання є критично важливою для забезпечення швидкої роботи, масштабованості та комфортного користувацького досвіду. Система, яка обробляє дані студентів, викладачів, оцінок і зв’язків між ними, повинна ефективно справлятися з великими обсягами запитів, мінімізувати затримки та оптимально використовувати ресурси. Цей розділ розглядає ключові методи оптимізації продуктивності, включаючи кешування, оптимізацію запитів до бази даних, асинхронну обробку та використання інструментів для підвищення швидкості роботи системи.

Одним із основних методів оптимізації є кешування. Кешування дозволяє зберігати часто запитувані дані в швидкодоступній пам’яті, зменшуючи кількість

звернень до бази даних або серверних обчислень. Наприклад, списки студентів, предметів і груп, які викладач переглядає регулярно, можна кешувати на сервері за допомогою інструментів, таких як Redis. Redis зберігає дані у форматі ключ-значення в оперативній пам’яті, що забезпечує доступ до них за мілісекунди. У вебсистемі кешування можна налаштувати для зберігання результатів запитів, таких як список оцінок студента за семестр. Якщо дані не змінюються, система повертає кешовану версію замість повторного запиту до бази. Це знижує навантаження на сервер і прискорює відгук інтерфейсу. Однак важливо налаштувати час життя кешу (TTL), щоб уникнути відображення застарілих даних. Наприклад, TTL у 5 хвилин для списків студентів забезпечує баланс між актуальністю та швидкістю.

Оптимізація запитів до бази даних є ще одним ключовим аспектом. Некоректно сформовані запити або надмірна кількість звернень до бази можуть значно уповільнити систему. Для підвищення продуктивності використовується індексація таблиць. Наприклад, у таблиці оцінок можна створити індекси для полів student_id і subject_id, що прискорює вибірку оцінок за конкретним студентом чи предметом. Крім того, варто уникати запитів із повним скануванням таблиць, замінюючи їх цільовими вибірками з умовами. Наприклад, замість SELECT * FROM grades використовується SELECT grade, date FROM grades WHERE student_id = ? AND semester = ?. Також ефективним є використання агрегатних функцій у базі даних (наприклад, SUM для підрахунку загального балу), замість обробки даних на сервері. Для складних зв’язків між таблицями (студенти, предмети, групи) доцільно застосовувати нормалізацію, але уникати надмірної кількості JOIN-операцій, які можуть уповільнити виконання.

Асинхронна обробка запитів дозволяє системі обробляти кілька операцій одночасно, що особливо важливо для інтерактивних функцій, таких як редагування оцінок або завантаження списків студентів. На сервері, побудованому, наприклад, на Node.js, асинхронність реалізована через механізми async/await і Promise. Наприклад, коли викладач оновлює оцінку, система може асинхронно зберегти дані в базі,

одночасно оновлюючи кеш і повертаючи підтвердження користувачу. Це зменшує час очікування. Для фонових завдань, таких як генерація звітів про прогрес групи, доцільно використовувати черги повідомлень, наприклад, RabbitMQ. Завдання додається до черги, а окремий обробник виконує його без блокування основного потоку. Це забезпечує стабільність інтерфейсу навіть при високому навантаженні.

Оптимізація клієнтської частини також відіграє важливу роль. Сучасні фронтенд-фреймворки, такі як React, дозволяють зменшити кількість оновлень інтерфейсу через віртуальний DOM, але надмірна кількість компонентів або неефективне управління станом можуть уповільнити рендеринг. Для прискорення завантаження сторінок використовується ледаче завантаження (lazy loading) компонентів і зображень. Наприклад, таблиця оцінок може завантажувати дані частинами (пагінація), відображаючи лише 20 записів за раз замість усього семестру. Крім того, стиснення статичних ресурсів (CSS, JavaScript) і використання Content Delivery Network (CDN) зменшують час завантаження. Наприклад, бібліотеки React і Material-UI можна підключати через CDN, що знижує навантаження на сервер.

Для масштабованості системи доцільно розглянути горизонтальне масштабування. Замість одного потужного сервера використовується кілька серверів, між якими розподіляється навантаження за допомогою балансувальника, такого як Nginx. Це дозволяє системі обробляти тисячі одночасних запитів, що актуально для великих освітніх закладів. У хмарних середовищах, таких як AWS, можна налаштувати автоскейлінг, який додає нові сервери при зростанні трафіку. Наприклад, під час сесії, коли викладачі масово оновлюють оцінки, система автоматично розширюється, щоб уникнути затримок.

Інструменти моніторингу продуктивності, такі як Prometheus і Grafana, допомагають відстежувати метрики системи в реальному часі. Prometheus збирає дані про час виконання запитів, використання CPU і пам’яті, а Grafana візуалізує їх у вигляді графіків. Наприклад, якщо середній час відповіді API перевищує 200 мс, система може сповістити адміністратора про необхідність оптимізації. Регулярний

аналіз логів також дозволяє виявляти вузькі місця, такі як повільні запити до бази або надмірне використання ресурсів.

Практична реалізація оптимізації включає кілька кроків. Спочатку проводиться профілювання системи для визначення найбільш повільних операцій. Наприклад, інструмент New Relic може показати, що запит на вибірку оцінок займає 500 мс через відсутність індексу. Далі впроваджуються зміни, такі як додавання індексу чи кешування. Після цього проводяться навантажувальні тести за допомогою інструментів, таких як JMeter, які симулюють одночасну роботу сотень користувачів. Результати тестів порівнюються з базовими показниками, щоб оцінити покращення. Наприклад, після додавання Redis середній час завантаження списку студентів може скоротитися з 300 мс до 50 мс.

Оптимізація продуктивності також враховує енергоефективність. Скорочення кількості запитів і обчислень зменшує споживання серверних ресурсів, що важливо для хмарних систем, де плата залежить від обсягу використання. Наприклад, оптимізація SQL-запитів може знизити витрати на базу даних у AWS RDS на 10-15%. Крім того, швидка система покращує користувацький досвід, що сприяє її популярності серед викладачів і студентів.

2.6 Безпека та захист даних

Безпека та захист даних є ключовими аспектами розробки інформаційної системи, особливо в освітній сфері, де обробляються персональні дані викладачів і студентів, а також конфіденційна інформація, така як оцінки, розклади. Недостатній рівень безпеки може призвести до несанкціонованого доступу, втрати даних або їх компрометації, що негативно вплине на довіру користувачів і репутацію навчального закладу. Для забезпечення надійного захисту система використовує комплексний підхід, який охоплює автентифікацію, авторизацію, шифрування даних, захист від атак, моніторинг і відповідність нормативним вимогам.

Для безпечного доступу до системи реалізовано механізм автентифікації через email і пароль, які перевіряються сервером. Паролі зберігаються в зашифрованому вигляді за допомогою алгоритму bcrypt, що забезпечує стійкість до атак типу brute-force завдяки адаптивному хешуванню з 12 ітераціями, балансуючи між безпекою та продуктивністю. Після автентифікації користувачу видається JSON Web Token (JWT), який використовується для авторизації. JWT містить зашифровану інформацію про користувача, таку як ідентифікатор і роль (наприклад, викладач), і підписується секретним ключем, що зберігається на сервері. Токен діє протягом 1 години, після чого потрібна повторна автентифікація. Для зручності та безпеки реалізовано refresh tokens із терміном дії 7 днів, які зберігаються в MongoDB і перевіряються за IP-адресою та User-Agent, щоб запобігти перехопленню. Авторизація базується на ролях: викладач має доступ лише до даних своїх груп і дисциплін, а адміністративні функції обмежено. У NestJS використано middleware для перевірки JWT і ролі перед обробкою запитів.

Дані захищаються як під час передачі, так і під час зберігання. Для передачі використовується протокол HTTPS із сертифікатом SSL/TLS, що шифрує запити та відповіді, запобігаючи перехопленню в незахищених мережах. Сертифікати генеруються через Let’s Encrypt і оновлюються кожні 90 днів за допомогою Certbot. У MongoDB персональні дані, такі як ПІБ та email, зберігаються без шифрування для швидкого доступу, але захищені конфігурацією ролей. Оцінки шифруються на рівні програми за алгоритмом через бібліотеку crypto у Node.js і розшифровуються лише для авторизованих користувачів. Для запобігання втрати даних реалізовано щоденне резервне копіювання за допомогою mongodump, із зберіганням копій протягом 30 днів в ізольованому сховищі.

Система захищена від поширених атак, таких як міжсайтові сценарії (XSS) і підробка міжсайтових запитів (CSRF). Для захисту від XSS у Next.js застосовується автоматична екранізація в JSX, а введені дані санітизуються через DOMPurify. На сервері реалізовано Content Security Policy (CSP), що обмежує джерела скриптів. Для

запобігання CSRF кожен запит, що змінює дані, супроводжується унікальним CSRF-токеном, який передається в заголовку X-CSRF-Token. Вхідні дані валідуються через class-validator у NestJS, що унеможливлює ін’єкції в запити до MongoDB. Для захисту від DDoS-атак планується інтеграція з Cloudflare, який фільтрує шкідливий трафік.
Для виявлення підозрілих дій реалізовано логування всіх операцій, таких як автентифікація чи зміна оцінок. Логи зберігаються в MongoDB у колекції logs, фіксуючи користувача, IP-адресу, час і деталі дії. Наприклад, запис про зміну оцінки містить ідентифікатор студента та новий бал. Логи аналізуються через Winston, що дозволяє фільтрувати записи за рівнем (info, warn, error). Для моніторингу використовуються Prometheus для збору метрик, таких як кількість запитів, і Grafana для їх візуалізації, що допомагає виявляти аномалії, наприклад, сплеск невдалих автентифікацій.

Користувачі погоджуються з політикою конфіденційності під час реєстрації, а функція видалення облікового запису дозволяє виконати вимоги щодо захисту даних. У разі розширення системи за межі України враховуватимуться вимоги GDPR, включаючи прозору обробку даних і права користувачів на їх видалення. Дії адміністраторів логуються для забезпечення підзвітності.

Потенційні ризики, такі як перехоплення токенів, мінімізуються коротким терміном дії JWT і перевіркою refresh tokens. Помилки конфігурації уникаються завдяки автоматизованому тестуванню через CI/CD. У майбутньому планується впровадження двофакторної автентифікації для додаткового захисту. Ці заходи забезпечують надійний захист даних, підтримуючи довіру користувачів і стабільність системи.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Реалізація клієнтської частини системи

Клієнтська частина інформаційної системи була розроблена з використанням сучасного фреймворку Next.js, який базується на бібліотеці React і дозволяє створювати високопродуктивні та динамічні користувацькі інтерфейси. Next.js вирізняється підтримкою серверного рендерингу (SSR), зручною системою маршрутизації та оптимізацією продуктивності, що робить його ідеальним вибором для створення складних веб-додатків. У поєднанні з бібліотекою Mantine, яка надає широкий набір готових UI-компонентів, було створено адаптивний, функціональний і максимально інтуїтивно зрозумілий інтерфейс, спеціально розроблений для потреб викладачів.

Основною метою при розробці клієнтської частини було забезпечення викладача простим і зручним інструментом для ефективного управління навчальними предметами, групами студентів, оцінками та іншими аспектами навчального процесу. Особлива увага приділялася логічній організації елементів інтерфейсу, мінімалістичному дизайну та швидкому доступу до ключових функцій. Усі компоненти інтерфейсу були спроектовані з урахуванням адаптивності, що дозволяє системі однаково ефективно працювати як на настільних комп’ютерах, так і на мобільних пристроях, таких як смартфони чи планшети.

Функціонально кожна сторінка веб-додатку складається з окремих React-компонентів, які відповідають за управління власними станами, обробку подій і завантаження даних. Наприклад, компонент таблиці студентів отримує дані з серверного API у форматі JSON, відображає їх у структурованому та зрозумілому вигляді, а при зміні оцінки локально оновлює стан і надсилає відповідний запит на сервер для синхронізації даних. Такий підхід забезпечує високу реактивність інтерфейсу та мінімізує необхідність перезавантаження сторінок, що покращує користувацький досвід. Для асинхронного завантаження даних використовуються

сучасні інструменти, такі як функція fetch або бібліотека Axios, які забезпечують швидку та надійну взаємодію з сервером.

Кожна дія викладача в системі — від авторизації до вибору предмета чи редагування оцінки — супроводжується відповідною подією у фронтенді, яка ініціює запит до сервера. Локальне управління станом додатку дозволяє уникнути надлишкових запитів до сервера, що значно знижує його навантаження та підвищує швидкодію системи. Для створення візуальних елементів інтерфейсу активно використовуються готові компоненти бібліотеки Mantine, такі як таблиці, форми, модальні вікна, кнопки тощо. Це дозволило значно прискорити процес розробки, забезпечити єдиний стиль дизайну та підтримувати консистентність інтерфейсу на всіх сторінках додатку.

Особлива увага приділялася адаптивності дизайну. Усі компоненти інтерфейсу автоматично адаптуються до розміру екрана користувача, змінюючи свою структуру та розміри для забезпечення комфортної роботи як на великих моніторах, так і на компактних екранах мобільних пристроїв. Наприклад, таблиці автоматично оптимізуються для відображення на маленьких екранах, а елементи управління, такі як кнопки чи форми, залишаються доступними та зручними. Мінімалістичний підхід до дизайну також відіграє важливу роль: усі зайві елементи приховано, а акцент зроблено на ключових функціях, що дозволяє викладачу швидко орієнтуватися в системі.

Уся взаємодія з інтерфейсом була спроектована з урахуванням максимальної простоти та інтуїтивності, щоб викладач міг працювати з системою без необхідності додаткового навчання чи ознайомлення з інструкціями. Використання сучасних інструментів розробки, таких як Next.js і Mantine, дозволило досягти високої швидкодії, зручного користувацького досвіду та гнучкості інтерфейсу. Це створює міцну основу для подальшого розвитку інформаційної системи, зокрема для розширення її функціоналу, інтеграції нових можливостей і підтримки в умовах реального навчального процесу. Такий підхід забезпечує не лише ефективну роботу

викладачів, але й можливість масштабування системи в майбутньому, що робить її надійним інструментом для освітнього середовища.

3.2. Реалізація серверної частини системи

Серверна частина інформаційної системи була розроблена з використанням сучасного фреймворку NestJS, який базується на платформі Node.js і забезпечує модульну архітектуру, високий рівень абстракції, повну підтримку TypeScript, а також чіткий поділ функціоналу на окремі компоненти. Такий підхід дозволяє створювати структуровану, гнучку та масштабовану серверну частину, яку легко підтримувати, тестувати та розширювати новими функціями відповідно до потреб проєкту. Завдяки модульності та чіткій організації коду NestJS забезпечує зручність у розробці складних систем, мінімізуючи технічний борг і полегшуючи подальшу інтеграцію нових можливостей.

Вся логіка взаємодії між серверною та клієнтською частинами реалізована через REST API, що забезпечує стандартизований і зручний спосіб обміну даними. Кожен маршрут API відповідає конкретним діям користувача в інтерфейсі, таким як реєстрація нового викладача, отримання списку студентів, оновлення оцінок, завантаження розкладу чи управління іншими сутностями системи. Усі маршрути логічно згруповані в окремі контролери, які відповідають за обробку запитів, пов’язаних із певними сутностями: викладачами, студентами, предметами, оцінками та розкладом. Така організація дозволяє чітко розділити функціонал і спрощує підтримку коду.

Кожен контролер взаємодіє з відповідним сервісом, у якому зосереджена вся бізнес-логіка системи. Сервіси відповідають за обробку вхідних даних, їх перевірку, виконання необхідних операцій і взаємодію з моделями бази даних. Наприклад, сервіс оцінювання отримує дані від клієнтської частини, перевіряє їх на відповідність типам і допустимим значенням, оновлює відповідний запис у базі даних і формує відповідь

для клієнта. Такий підхід забезпечує ізоляцію бізнес-логіки від маршрутів, що сприяє кращій структурованості коду та полегшує його тестування.

Безпека системи є одним із ключових аспектів реалізації. Для захисту даних і контролю доступу було впроваджено механізм авторизації на основі токенів. Після успішного входу в систему викладач отримує унікальний токен, який додається до кожного подальшого запиту до захищених маршрутів. Цей токен перевіряється спеціальним middleware перед обробкою запиту, що гарантує, що доступ до конфіденційних даних, таких як оцінки чи персональна інформація студентів, мають лише автентифіковані користувачі. Усі захищені маршрути доступні виключно за наявності дійсного токена, що забезпечує високий рівень безпеки системи.

Особлива увага приділялася детальній перевірці прав доступу. Навіть якщо користувач має дійсний токен, він може переглядати або редагувати лише ті дані, які стосуються його діяльності. Наприклад, викладач має доступ лише до тих груп, предметів і студентів, які закріплені за ним. Це реалізовано через додаткову перевірку відповідності ідентифікаторів викладача та предмета під час обробки кожного запиту. Такий підхід забезпечує не лише автентифікацію, а й авторизацію дій, що дозволяє чітко обмежити доступ до даних у межах дозволених прав.

Ще однією важливою особливістю серверної частини є ретельна обробка помилок і валідація даних. Усі вхідні запити проходять перевірку на правильність, повноту та відповідність заздалегідь визначеним схемам. У разі виявлення помилок, наприклад, некоректного формату даних чи відсутності обов’язкових полів, система повертає користувачу чітке повідомлення про помилку з поясненням причини. Завдяки вбудованим засобам валідації NestJS ці перевірки інтегруються в кожен маршрут без необхідності дублювання коду, що спрощує розробку та підвищує надійність системи.

Структура обробки запитів у системі є типовою та послідовною: отримання запиту, перевірка токена, валідація вхідних даних, звернення до відповідного сервісу, взаємодія з моделлю бази даних, формування та відправлення відповіді клієнту. Такий підхід забезпечує чистоту коду, полегшує його підтримку та дозволяє швидко вносити

зміни чи масштабувати систему за потреби. Наприклад, додавання нового маршруту чи сервісу не вимагає значних змін у наявному коді.

Завдяки модульній архітектурі NestJS серверна частина залишається легкою для підтримки та розширення. У разі потреби систему можна доповнити новими функціями, такими як адміністративна панель для управління користувачами, автоматизована розсилка результатів студентам, вивантаження статистики у форматі PDF чи інтеграція з іншими зовнішніми сервісами. Модульність дозволяє ізолювати новий функціонал у окремі компоненти, що мінімізує вплив на вже наявну логіку.

Таким чином, серверна частина, побудована на NestJS, відповідає сучасним вимогам до безпеки, швидкодії та структурованості. Вона забезпечує надійну обробку даних, контроль доступу, ефективну взаємодію з базою даних і надає клієнтській частині стабільний і зручний API для роботи. Модульна архітектура та використання сучасних інструментів створюють міцну основу для подальшого розвитку системи, її масштабування та адаптації до нових вимог навчального процесу.

3.3. Взаємодія з базою даних

У розробленій системі базу даних реалізовано за допомогою MongoDB — сучасної документоорієнтованої системи керування базами даних, яка дозволяє зберігати дані у вигляді гнучких структур JSON-подібного формату. Такий підхід дозволяє легко адаптувати структуру даних до потреб системи, змінювати формат полів без необхідності повного рефакторингу та масштабувати сховище відповідно до збільшення обсягу даних.

Усі основні сутності системи представлені у вигляді окремих колекцій: викладачі, предмети, студенти, оцінки, розклад. Для опису структури кожного документа використовується бібліотека Mongoose, яка дозволяє створювати схеми з чітко визначеними типами даних, обов’язковими полями, зв’язками між сутностями та кастомними валідаціями. Такий підхід дозволяє поєднувати гнучкість MongoDB з передбачуваністю класичної реляційної моделі.

Кожна сутність має свою схему. Наприклад, викладач має поля: унікальний ідентифікатор, ПІБ, email, номер телефону, пароль та список предметів, до яких він прикріплений. Усі ці структури логічно взаємопов’язані: наприклад, запис студента містить лише ID групи, а оцінка — ID студента та предмета.

Запити до бази даних реалізовано через відповідні сервіси в NestJS. Отримання даних реалізовано за допомогою методів пошуку — find, findOne або findById, з додатковими фільтрами. Оновлення реалізується через методи updateOne або findByIdAndUpdate. Для створення нових об’єктів використовуються методи create. Усі запити обгорнуті у try-catch блоки, що дозволяє коректно обробляти помилки та виводити інформативні повідомлення користувачеві у разі помилкового введення або втрати зв’язку з базою.

Для збереження узгодженості даних використовуються перевірки на стороні сервісів. Наприклад, при спробі записати оцінку система перевіряє, чи належить студент до групи, що пов’язана з викладачем, та чи існує такий предмет. Усі взаємозв’язки побудовані на використанні ID-зв’язків, що забезпечує гнучкість у зміні структури документів без порушення логіки.

Особливу увагу під час проєктування структури бази даних приділено розмежуванню ролей та забезпеченню безпеки. Завдяки токен-ідентифікації, доступ до даних обмежується виключно межами дозволених операцій: викладач не може переглядати або змінювати дані іншого користувача, навіть маючи прямий ідентифікатор колекції. Це забезпечується перевірками на рівні сервісів та middleware.

У моделі оцінювання реалізовано гнучку структуру, яка дозволяє легко додавати нові типи робіт, наприклад, лабораторні, проєктні чи індивідуальні завдання. Усі поля мають числовий тип з обмеженням допустимих значень, що унеможливлює випадкове введення неправильних даних. Крім того, сумарна оцінка розраховується автоматично на клієнтському рівні, але також перевіряється на сервері.

Усі запити до бази даних логуються в системі, що дозволяє відстежувати активність користувачів. Це є важливою функцією для адміністрування — у разі

технічних збоїв або конфліктних ситуацій можна відновити повний ланцюг змін. У майбутньому планується впровадження збереження історії змін, щоб викладач міг відкотити оцінки або перевірити, хто і коли їх змінював.

Окремим компонентом є колекція розкладу занять, яка дозволяє зберігати дані про дати, час, дисципліни, аудиторії та групи. Це забезпечує інтеграцію навчального процесу з модулями оцінювання та дозволяє формувати повну картину академічної активності. Розклад може бути динамічним, з можливістю оновлення в режимі реального часу.

Крім того, структура бази даних розроблена з урахуванням потенційного розширення функціоналу системи. За потреби до існуючих моделей можуть бути додані нові поля або створені нові колекції. Гнучкість документоорієнтованого підходу у MongoDB дозволяє масштабувати проєкт без значних змін у логіці додатку, що робить обрану архітектуру перспективною для подальшого розвитку вебзастосунку.

З метою покращення продуктивності та масштабованості система підтримує можливість розділення бази даних на окремі кластери. Це дає змогу у майбутньому реалізувати горизонтальне масштабування, що є особливо актуальним при збільшенні кількості користувачів або обсягу навчальних даних. Завдяки вбудованим механізмам MongoDB, як-от шардинг і реплікація, система може зберігати високу швидкодію навіть за інтенсивного навантаження.

Для полегшення процесу розробки та тестування використовуються попередньо заповнені демо-дані, які дозволяють імітувати роботу з реальними студентами, викладачами та розкладом. Це дало змогу провести повноцінне моделювання поведінки системи без ризику для реального навчального процесу. Усі дані структуровано та документовано, що забезпечує зручність при внесенні змін і супроводі проєкту.

Нижче наведено узагальнену структуру основних колекцій бази даних таблиця 3.1.

Таблиця 3.1 – Структура основних колекцій у MongoDB

Колекція	Основні поля	Примітка
Викладач	ID, ПІБ, email, телефон, пароль, список предметів	Має доступ лише до закріплених предметів
Предмет	ID, назва, ID викладача, ID групи	Є зв’язковою ланкою між викладачем і групою
Студент	ID, ПІБ, email, номер у списку, ID групи, об’єкт оцінок	Прив’язаний до групи та предмета через оцінки
Оцінка	ID студента, ID предмета, практичні 1-3, колоквіум, модуль, сума	Вкладена структура у студента або окрема
Розклад	Дата, час, ID групи, ID предмета	Використовується для зберігання даних розкладу занять

Таким чином, взаємодія з базою даних у системі побудована на основі гнучких, але водночас чітко структурованих моделей. Це забезпечує ефективне зберігання інформації, її швидкий пошук, фільтрацію та редагування. Обраний підхід дозволяє масштабувати систему без зміни логіки взаємодії між компонентами, підтримувати її в актуальному стані та легко додавати нові сутності чи поля при розширенні функціоналу.

3.4. Інструкція для користувача

Для ефективної роботи з системою викладачеві необхідно пройти кілька основних кроків, які забезпечують доступ до функціональності вебзастосунку. Інтерфейс системи побудований таким чином, щоб забезпечити інтуїтивно зрозумілу взаємодію та мінімізувати час, необхідний для виконання типових дій: реєстрації, входу, перегляду студентів, виставлення оцінок.

Після відкриття головної сторінки користувач бачить екран авторизації. Якщо обліковий запис ще не створено, необхідно перейти до сторінки реєстрації Рисунок 3.1. Тут викладач заповнює просту форму, в якій вказує свій email, ПІБ, номер телефону, пароль та обирає предмет, за яким закріплюється за допомогою Multiselect Рисунок 3.2. Після підтвердження даних система створює обліковий запис викладача.

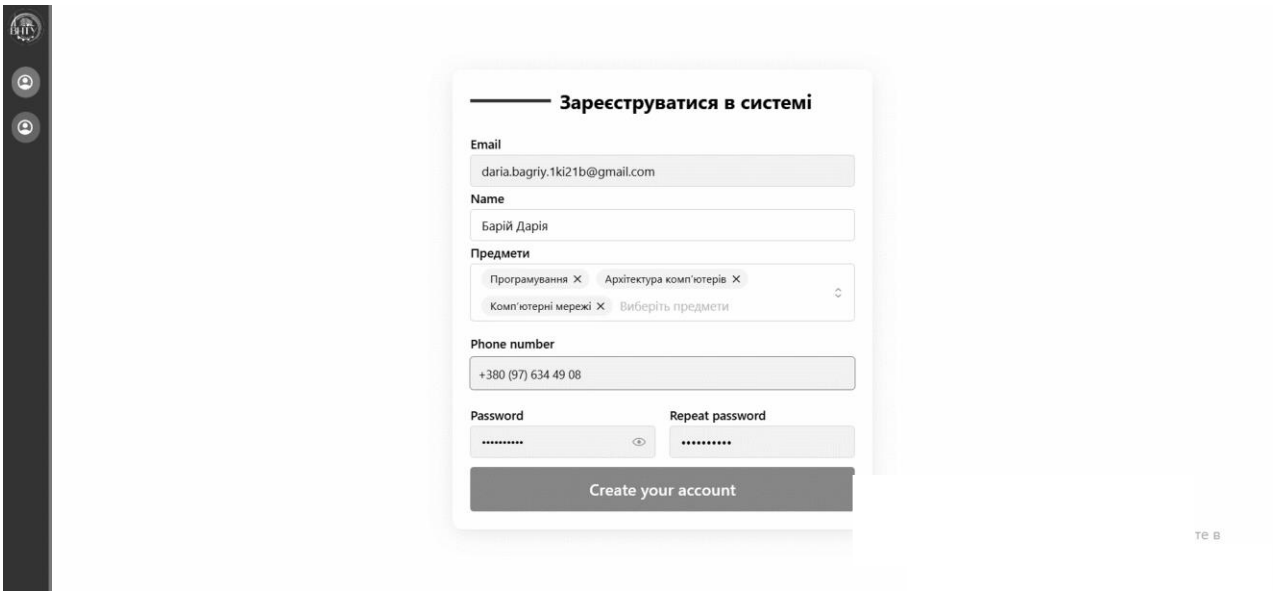


Рисунок 3.1 – Сторінка реєстрації користувача

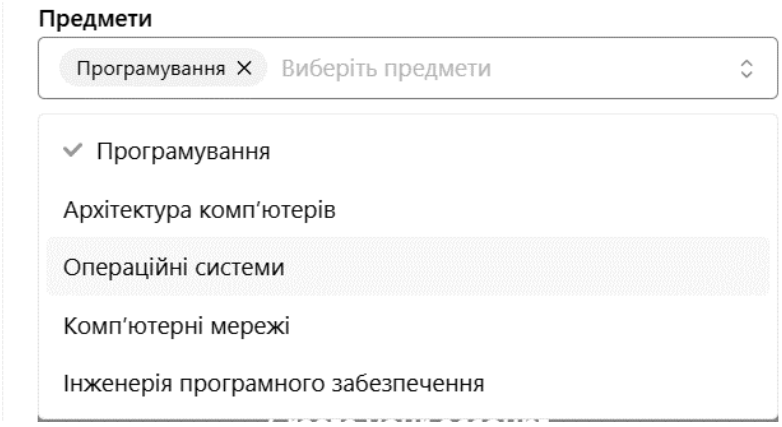


Рисунок 3.2 – Multiselect при виборі предмету

Для входу в систему користувачеві достатньо перейти на сторінку "Login" та ввести свій email та пароль на екрані авторизації Рисунок 3.3. Після введення даних та натискання кнопки "Log in to Jet" система перевіряє коректність введених даних. У

разі успішної перевірки користувач отримує доступ до персонального кабінету. Якщо введено неправильний email або пароль, система відображає відповідне повідомлення про помилку, пропонуючи повторити спробу.

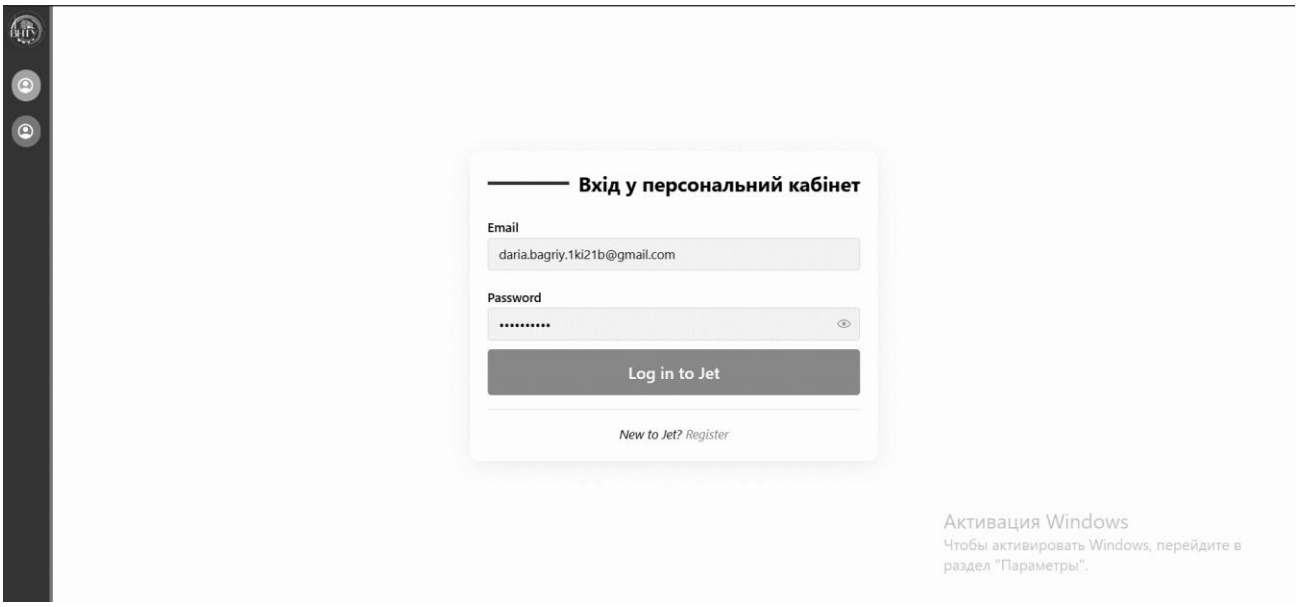


Рисунок 3.3 – Сторінка входу в систему(авторизація)

Після успішної авторизації користувач автоматично перенаправляється на сторінку профілю Рисунок 3.4. Цей розділ є персональним кабінетом викладача і слугує стартовою точкою для подальшої роботи в системі. На сторінці профілю відображається базова інформація про користувача, зокрема: повне ім’я викладача, електронна пошта та номер телефону, вказані під час реєстрації.

Окрім відображення персональних даних, інтерфейс сторінки профілю надає зручну навігацію до інших функціональних розділів системи. Звідси викладач може перейти до перегляду предметів, за якими він закріплений, керування навчальними групами, а також перегляду списків студентів, приєднаних до відповідних груп, предметів. Таким чином, сторінка профілю виступає центральним елементом взаємодії викладача з інформаційною системою та забезпечує швидкий доступ до основних інструментів для організації навчального процесу.

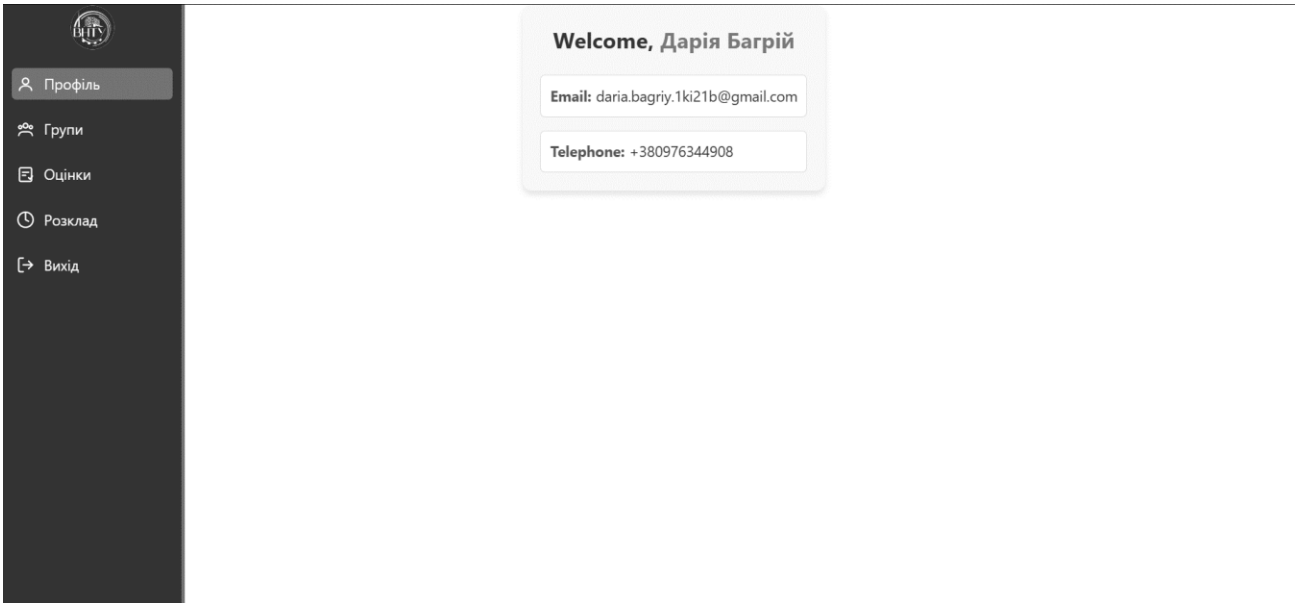


Рисунок 3.4 – Сторінка профілю викладача

Перейшовши на сторінку ’’Групи’’ у верхній частині екрана розміщено Multiselect у якому перелічено предмети, а саме програмування, архітектура комп’ютерів, операційні системи, комп'ютерні мережі, інженерія програмного забезпечення, точніше, які були прикріплені до викладача у виборі при реєстрації. Та містить такі групи як: КІ-21П, КІ-22А, КІ-22О, КІ-22К, КІ-22І. Після вибору одного з них з’являється список відповідних навчальних груп Рисунок 3.5. Обираючи групу з'являється таблиця відповідних студентів Рисунок 3.6. У ній відображається номер за списком, прізвище, ім’я, по батькові, електронна пошта, а також сумарна сума балів.

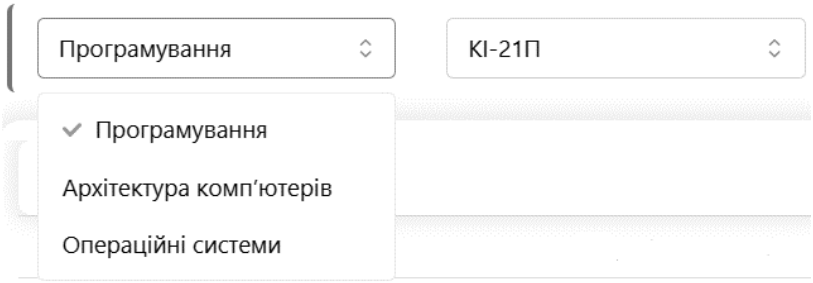
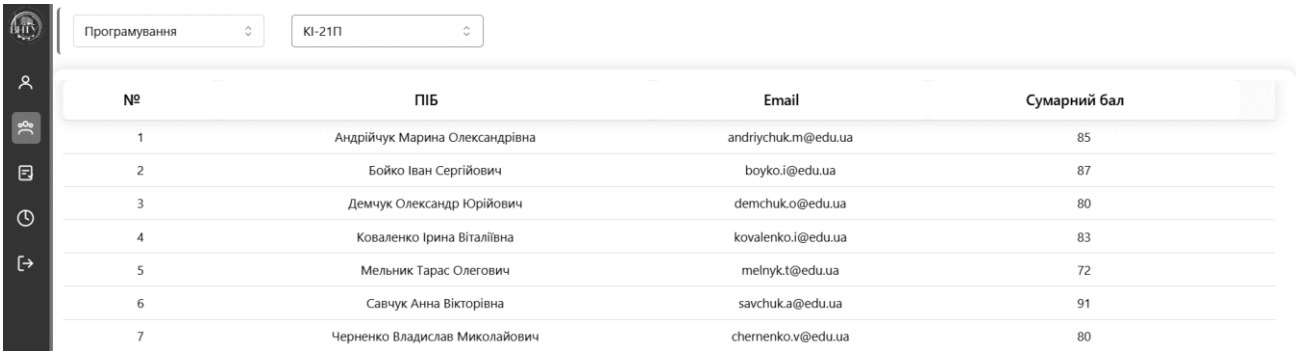


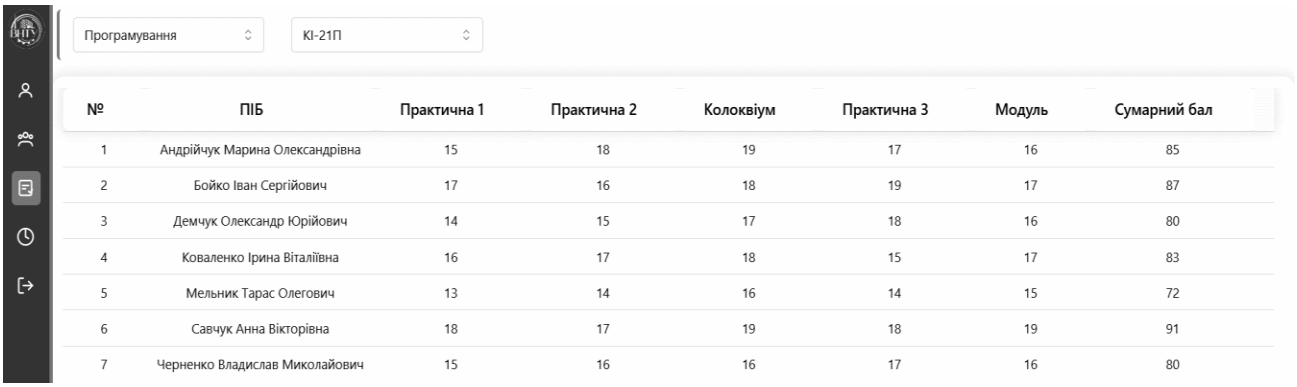
Рисунок 3.5 – Перелік предметів і груп



№	ПІБ	Email	Сумарний бал
1	Андрійчук Марина Олександрівна	andriychuk.m@edu.ua	85
2	Бойко Іван Сергійович	boyko.i@edu.ua	87
3	Демчук Олександр Юрійович	demchuk.o@edu.ua	80
4	Коваленко Ірина Віталіївна	kovalenko.i@edu.ua	83
5	Мельник Тарас Олегович	melnyk.t@edu.ua	72
6	Савчук Анна Вікторівна	savchuk.a@edu.ua	91
7	Черненко Владислав Миколайович	chernenko.v@edu.ua	80

Рисунок 3.6 – Таблиця студентів групи

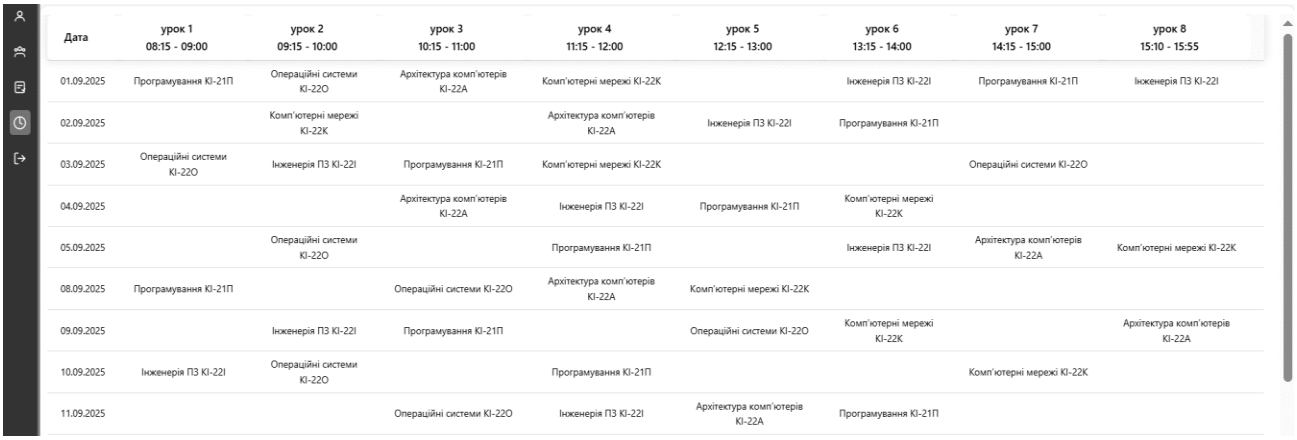
Для детального оцінювання викладач переходить у розділ виставлення балів, а саме ’’Оцінки’’. У цьому розділі формується таблиця, де кожен рядок відповідає одному студенту, а кожен стовпець — окремому виду роботи: практична 1, практична 2, колоквіум, практична 3, модуль, а також підсумковий бал. Усі бали є редагованими за допомогою бази даних без потреби перезавантаження сторінки.



№	ПІБ	Практична 1	Практична 2	Колоквіум	Практична 3	Модуль	Сумарний бал
1	Андрійчук Марина Олександрівна	15	18	19	17	16	85
2	Бойко Іван Сергійович	17	16	18	19	17	87
3	Демчук Олександр Юрійович	14	15	17	18	16	80
4	Коваленко Ірина Віталіївна	16	17	18	15	17	83
5	Мельник Тарас Олегович	13	14	16	14	15	72
6	Савчук Анна Вікторівна	18	17	19	18	19	91
7	Черненко Владислав Миколайович	15	16	16	17	16	80

Рисунок 3.7 – Таблиця оцінювання студентів

Для перегляду розкладу занять викладач може перейти до відповідного розділу в інтерфейсі. На цій сторінці відображаються дати проведення занять, час початку і закінчення, назва предмета та група, якій призначене заняття. Це дозволяє користувачу оперативно орієнтуватися у графіку проведення занять, планувати оцінювання та контролювати послідовність навчального процесу.



Дата	урок 1 08:15 - 09:00	урок 2 09:15 - 10:00	урок 3 10:15 - 11:00	урок 4 11:15 - 12:00	урок 5 12:15 - 13:00	урок 6 13:15 - 14:00	урок 7 14:15 - 15:00	урок 8 15:10 - 15:55
01.09.2025	Програмування КІ-21П	Операційні системи КІ-22О	Архітектура комп'ютерів КІ-22А	Комп'ютерні мережі КІ-22К		Інженерія ПЗ КІ-22І	Програмування КІ-21П	Інженерія ПЗ КІ-22І
02.09.2025		Комп'ютерні мережі КІ-22К		Архітектура комп'ютерів КІ-22А	Інженерія ПЗ КІ-22І	Програмування КІ-21П		
03.09.2025	Операційні системи КІ-22О	Інженерія ПЗ КІ-22І	Програмування КІ-21П	Комп'ютерні мережі КІ-22К			Операційні системи КІ-22О	
04.09.2025			Архітектура комп'ютерів КІ-22А	Інженерія ПЗ КІ-22І	Програмування КІ-21П	Комп'ютерні мережі КІ-22К		
05.09.2025		Операційні системи КІ-22О		Програмування КІ-21П		Інженерія ПЗ КІ-22І	Архітектура комп'ютерів КІ-22А	Комп'ютерні мережі КІ-22К
08.09.2025	Програмування КІ-21П		Операційні системи КІ-22О	Архітектура комп'ютерів КІ-22А	Комп'ютерні мережі КІ-22К			
09.09.2025		Інженерія ПЗ КІ-22І	Програмування КІ-21П		Операційні системи КІ-22О	Комп'ютерні мережі КІ-22К		Архітектура комп'ютерів КІ-22А
10.09.2025	Інженерія ПЗ КІ-22І	Операційні системи КІ-22О		Програмування КІ-21П			Комп'ютерні мережі КІ-22К	
11.09.2025			Операційні системи КІ-22О	Інженерія ПЗ КІ-22І	Архітектура комп'ютерів КІ-22А	Програмування КІ-21П		

Рисунок 3.7 – Сторінка розкладу

Після виконанної роботи користувач може вийти із системи, натиснувши відповідну кнопку виходу. Всі дані залишаються збереженими, і викладач може повернутись до них у будь-який момент.

Таким чином, взаємодія з системою побудована у максимально зручний спосіб, без перевантаження інтерфейсу зайвими елементами. Кожен етап роботи — від входу до виставлення оцінок — реалізований через просту й логічну послідовність дій. Це дозволяє викладачеві швидко освоїти роботу з платформою навіть без попередніх інструкцій або технічних навичок. Система забезпечує стабільність, оперативність і зручність оцінювання, що позитивно впливає на загальну ефективність контролю навчального процесу.

3.5 Тестування розробленої системи

Під час розробки вебзастосунку було проведено ретельне базове тестування з метою перевірки працездатності основного функціоналу, виявлення можливих логічних помилок, забезпечення стабільності користувацького інтерфейсу та коректності обробки даних на серверній частині. Тестування здійснювалося вручну на різних етапах розробки: спочатку для окремих компонентів системи, а згодом — після їх інтеграції в єдину цілісну систему. Основною метою тестування було гарантувати надійність, зручність і безпеку використання системи викладачами в реальних умовах

навчального процесу. Перевірялися ключові аспекти роботи системи, включаючи валідність даних, реакцію на помилкові сценарії, швидкість завантаження даних, стабільність роботи інтерфейсу та безпеку доступу до захищених ресурсів. Це дозволило переконатися, що система відповідає вимогам користувачів і може ефективно використовуватися в освітньому середовищі.

Особлива увага приділялася тестуванню основних функціональних блоків системи, які є критично важливими для роботи викладачів: реєстрація та авторизація користувачів, управління предметами, групами та списками студентів, редагування оцінок, автоматичне оновлення сумарного балу, обробка невалідних даних, перевірка доступу до захищених маршрутів і безпечний вихід із системи. Кожен із цих модулів був ретельно протестований для забезпечення стабільності, коректності та безпеки. Наприклад, під час тестування процесу реєстрації перевірялася валідність введених даних, таких як формат email, відповідність паролів встановленим вимогам безпеки (наприклад, мінімальна довжина, наявність спеціальних символів), а також реакція системи на спроби створення дублікатів акаунтів. Система коректно обробляла помилки, виводячи чіткі повідомлення без необхідності перезавантаження сторінки, а дані викладача успішно зберігалися в базі даних для подальшого використання під час авторизації.

Тестування авторизації включало перевірку коректності введення email і пароля, правильність видачі токена доступу, його збереження для подальших запитів, а також обробку сценаріїв із невірними даними. У разі введення неправильного пароля чи email система виводила зрозумілі повідомлення про помилку, що дозволяло користувачу швидко зрозуміти причину проблеми. Крім того, перевірялася стабільність роботи токена доступу під час виконання запитів до захищених маршрутів, а також коректність завершення сеансу користувача після виходу з системи. Це забезпечило надійність і безпеку авторизації, що є критично важливим для захисту конфіденційних даних.

Функціонал управління предметами, групами та списками студентів тестувався на точність відображення даних, коректність збереження зв’язків між викладачем, предметом і групою, а також швидкість завантаження інформації. Дані завантажувалися асинхронно через API, що забезпечувало швидке відображення без затримок, а інформація про студентів (ПІБ, email, порядковий номер, загальний бал) відповідала записам у базі даних. Окремо перевірялася навігація між сторінками системи, яка виявилася інтуїтивною, стабільною та зручною для користувача. Тестування також включало перевірку реакції системи на невалідні запити, такі як спроби несанкціонованого доступу до даних. У таких випадках система надійно блокувала доступ і виводила відповідні повідомлення, що підтвердило її стійкість до потенційних загроз.

Модуль редагування оцінок, який є ключовим елементом системи, зазнав особливо ретельного тестування, оскільки він безпосередньо впливає на навчальний процес. Перевірялася можливість змінювати оцінки в таблиці, автоматичне оновлення сумарного балу студента після внесення змін, а також коректне збереження оновлених даних у базі. Система успішно обробляла як валідні значення (наприклад, оцінки в межах допустимого діапазону), так і невалідні (наприклад, від’ємні значення чи текстові дані замість чисел), забезпечуючи цілісність даних і уникаючи помилок. Додатково тестувалася реакція інтерфейсу на швидкі послідовні зміни оцінок, що підтвердило стабільність роботи системи навіть за інтенсивного використання. Перевірка безпеки передачі та збереження даних також дала позитивні результати, гарантуючи захист інформації від несанкціонованого доступу.

Окремо тестувалася адаптивність інтерфейсу на різних пристроях — від настільних комп’ютерів до мобільних телефонів. Усі компоненти коректно відображалися на екранах різного розміру, а елементи управління залишалися зручними та доступними. Тестування також охоплювало сценарії з нестабільним інтернет-з’єднанням, де система демонструвала стабільність завдяки локальному

управлінню станом і асинхронній обробці запитів. Це забезпечило безперебійну роботу навіть у складних умовах.

Результати тестування підтвердили високу стабільність і працездатність вебзастосунку. Усі ключові функції — реєстрація, авторизація, обробка запитів, редагування оцінок, навігація між сторінками та взаємодія з базою даних — працюють без критичних помилок. Система забезпечує швидке завантаження даних, інтуїтивно зрозумілий інтерфейс, надійний захист інформації та стабільну роботу в різних сценаріях використання. Отримані результати свідчать про те, що вебзастосунок повністю готовий до використання викладачами в реальному навчальному процесі. Він відповідає поставленим вимогам щодо функціональності, безпеки, швидкодії та зручності, створюючи надійну основу для ефективного управління навчальними даними та подальшого розвитку системи.

ВИНОВКИ

У рамках даної бакалаврської роботи було розроблено інформаційну систему контролю навчального процесу, призначену для викладачів із дисциплін комп'ютерної інженерії. Основною метою проєкту було створення сучасного, зручного та функціонального вебзастосунку, який забезпечує автоматизацію процесів управління навчальними групами, предметами, студентами та їхніми оцінками. У результаті виконаної роботи створено повноцінну систему, яка відповідає ключовим потребам викладачів у структуризації та організації навчального процесу, забезпечуючи ефективну взаємодію з навчальною інформацією та спрощуючи виконання повсякденних завдань.

На початковому етапі виконання роботи було проведено детальний аналіз аналогічних інформаційних систем, доступних на ринку. Вивчення їхніх функціональних можливостей, переваг і недоліків дозволило сформувати чіткі вимоги до розроблюваного програмного забезпечення. Цей аналіз став основою для визначення функціональних і нефункціональних вимог, які враховували потреби кінцевих користувачів — викладачів. На основі отриманих даних було спроєктовано архітектуру системи, розроблено структуру бази даних, створено прототипи користувацького інтерфейсу та детально описано логіку взаємодії між усіма компонентами системи, включаючи клієнтську та серверну частини.

Клієнтська частина системи була реалізована з використанням сучасного фреймворку Next.js, який базується на бібліотеці React і забезпечує високу продуктивність, підтримку серверного рендерингу та зручну маршрутизацію. Для створення інтерфейсу використовувалася бібліотека Mantine, яка надала набір готових UI-компонентів, що дозволило значно прискорити розробку та забезпечити єдиний стиль дизайну. У результаті було створено адаптивний, інтуїтивно зрозумілий і зручний інтерфейс, який забезпечує комфортну роботу викладачів як на настільних комп'ютерах, так і на мобільних пристроях. Інтерфейс спроєктовано з акцентом на

мінімалізм і логічність розташування елементів, що дозволяє швидко виконувати основні дії без необхідності додаткового навчання.

Серверна частина системи розроблена на основі фреймворку NestJS, який працює на платформі Node.js і забезпечує модульну архітектуру, підтримку TypeScript, а також чіткий поділ функціоналу на компоненти. Використання NestJS дозволило створити структуровану, масштабовану та легку для підтримки серверну частину. Взаємодія з клієнтською частиною здійснюється через REST API, яке забезпечує стандартизований обмін даними. API включає маршрути для всіх ключових операцій, таких як реєстрація, авторизація, управління предметами, групами, студентами та оцінками. Безпека даних забезпечується за допомогою механізмів авторизації на основі токенів, а також додаткових перевірок прав доступу, що гарантує захист конфіденційної інформації.

Для зберігання даних у системі використовується база даних MongoDB у поєднанні з бібліотекою Mongoose. Це рішення забезпечує гнучке моделювання даних, дозволяючи легко описувати структури даних і підтримувати їх узгодженість. Mongoose спрощує взаємодію з базою даних, забезпечуючи зручну роботу з моделями та ефективну обробку запитів. Структура бази даних була спроєктована з урахуванням потреб системи, включаючи зберігання інформації про викладачів, студентів, предмети, групи, оцінки та розклад.

До складу розробленої інформаційної системи входять основні функціональні модулі: реєстрація та авторизація викладачів, управління предметами та групами, перегляд і редагування списків студентів, виставлення оцінок за різні види навчальної діяльності, а також модуль для роботи з розкладом. Кожен із цих модулів був ретельно протестований вручну для забезпечення працездатності, коректності логіки, правильної обробки помилок і адаптивності інтерфейсу. Тестування охоплювало перевірку валідності даних, реакцію системи на некоректні запити, швидкість обробки інформації та безпеку доступу до захищених ресурсів. Усі виявлені помилки були виправлені, що забезпечило стабільність і надійність системи.

Розроблене програмне забезпечення повністю відповідає поставленим завданням і вимогам. Воно є функціональним, зручним у використанні та придатним для застосування в реальному навчальному процесі як внутрішній інструмент викладача. Система забезпечує швидкий доступ до всіх необхідних функцій, інтуїтивну навігацію та безпечну обробку даних. Крім того, створений вебзастосунок має значний потенціал для подальшого розвитку. У майбутньому можливе розширення функціоналу, наприклад, додавання аналітичних інструментів для оцінки результатів навчання, підтримка акаунтів для студентів, інтеграція з іншими освітніми платформами чи впровадження автоматизованих функцій, таких як генерація звітів чи розсилка повідомлень.

Таким чином, у ході виконання бакалаврської роботи було досягнуто поставленої мети дослідження, успішно реалізовано всі заплановані задачі та створено програмний продукт, який має високу практичну цінність для освітнього середовища. Розроблена система є надійним і ефективним інструментом для викладачів, забезпечуючи автоматизацію ключових процесів і створюючи міцну основу для подальшого вдосконалення та масштабування. Перспективи використання системи в реальному навчальному процесі підтверджують її актуальність і відповідність сучасним вимогам до освітніх інформаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Braude E. Основи програмування та структури даних: пер. з англ. — К.: Видавництво «Лібра», 2010. — 432 с.

2. Стеценко А. В., Пашко В. О. Проектування інформаційних систем: навчальний посібник. — К.: КНУ, 2020. — 248 с.

3. Глушков В. М. Основи створення інформаційних систем. — К.: Наукова думка, 2009. — 366 с.

4. NestJS – офіційна документація [Електронний ресурс]. — Режим доступу: https://docs.nestjs.com

5. Next.js – офіційна документація [Електронний ресурс]. — Режим доступу: https://nextjs.org/docs

6. Mantine – бібліотека компонентів React [Електронний ресурс]. — Режим доступу: https://mantine.dev

7. MongoDB – офіційна документація [Електронний ресурс]. — Режим доступу: https://www.mongodb.com/docs

8. Фролов Є. П. Інтерфейс користувача: від архітектури до зручності. — Харків: ФОП Ранок, 2019. — 180 с.

9. Технології вебпрограмування: навч. посіб. / За ред. О. Ю. Гриценка. — К.: Ліра-К, 2021. — 212 с.

10. Ковальчук О. П. Технології розробки веб-застосунків: навчальний посібник. — Вінниця: ВНТУ, 2020. — 180 с.

11. Медведєва Ю. В., Некос А. М. Використання системи Moodle для контролю знань з екології учнів закладів загальної середньої освіти // Інформаційні технології і засоби навчання. — 2018. — №1(63). — С. 56 67. — Режим доступу: https://journal.iitta.gov.ua/index.php/itlt/article/view/1931/1304

12. Тимченко Г. М., Літвінова А. М. Зміна парадигми змішаного навчання в системі класичної освіти в умовах активного використання LMS Moodle та Google Classroom

// ResearchGate. — 2022. — Режим доступу: https://www.researchgate.net/publication/366613206

13. Пукач П. Ю. Досвід змішаного навчання інформатики студентів економічних спеціальностей з використанням засобів LMS Moodle та YouTube // Інформаційні технології і засоби навчання. — 2021. — №1(81). — С. 113–124. — Режим доступу: https://journal.iitta.gov.ua/index.php/itlt/article/view/3945/1808

14. Воронін І. В. Основи баз даних: навчальний посібник. — Вінниця: ВНТУ, 2017. — 150 с.

15. Ковальчук О. П. Системи управління базами даних: навчальний посібник. — Вінниця: ВНТУ, 2019. — 200 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н. проф._____Азаров О. Д.
«21» березня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання комплексної бакалаврської кваліфікаційної роботи
«Інформаційна система контролю навчального процесу з дисциплін комп’ютерної інженерії»

08-54.БКР.001.00.000 ПЗ

Керівник роботи д.т.н., проф. каф. ОТ

Крупель Л. В.

Студентка групи 1КІ-21б

Багрій Д. В.

ВНТУ 2025

1 Підстава для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність теми полягає в зростаючій потребі у цифрових інструментах для організації та моніторингу навчального процесу. Більшість існуючих систем є складними у використанні або не адаптованими до специфіки конкретних освітніх закладів. Розробка зручної та функціональної інформаційної системи для викладача дозволить оптимізувати керування навчальними групами, предметами, оцінюванням студентів, що є особливо актуальним у контексті цифровізації освіти.

1.2 Наказ ВНТУ №97 про затвердження теми БКР від 20.03.2025.

2 Мета БКР і призначення розробки

2.1 Мета роботи — створення вебзастосунку для викладачів, який дозволяє керувати навчальним процесом: обирати предмети та групи, переглядати списки студентів, виставляти оцінки, переглядати розклад занять і зберігати дані в базі MongoDB через захищені серверні API.

2.2 Призначення розробки — забезпечити інструмент для викладачів, що автоматизує ведення обліку оцінок, перегляду груп і дисциплін, що підвищує ефективність навчального процесу в закладах вищої освіти.

3 Вихідні дані для виконання БКР

3.1 Базові знання основ веброзробки: HTML, CSS, JavaScript, React, Node.js.

3.2 Основи роботи з базами даних: MongoDB, Mongoose.

3.3 Робота з фреймворками Next.js (Frontend) і NestJS (Backend).

3.4 Вимоги до безпеки, авторизації (JWT, токени).

3.5 Принципи REST-архітектури.

3.6 Розробка інтерфейсів користувача (UI/UX).

3.7 Практика ручного тестування вебзастосунків.

4 Вимоги до виконання БКР

4.1 Провести аналіз аналогічних систем (JetIQ, Google Classroom);

4.2 Сформувати вимоги до вебсистеми викладача;

4.3 Розробити архітектуру інформаційної системи;

4.4 Реалізувати клієнтську частину (Next.js, Mantine);

4.5 Реалізувати серверну частину з API (NestJS, MongoDB);

4.6 Забезпечити авторизацію користувачів;

4.7 Провести тестування системи та написати інструкцію користувача;

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКР

№ з/п	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз задачі	21.03.25	30.03.25	Огляд джерел
2	Аналітичний огляд систем моніторингу навчання	31.03.25	10.04.25	Розділ 1
3	Проєктування структури системи, моделей, БД	11.04.25	21.04.25	Розділ 2
4	Реалізація інтерфейсу та серверної логіки	22.04.25	30.04.25	Розділ 3
5	Тестування, інструкція користувача, підготовка графіки	31.04.25	02.05.25	Розділ 3
6	Оформлення пояснювальної записки та презентації	03.05.24	11.05.24	ПЗ, презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02- П.001.01:21.

ДОДАТОК Б Перевірка на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Інформаційна система контролю навчального процесу з дисциплін комп'ютерної інженерії

Тип роботи: _____ бакалаврська кваліфікаційна робота _____

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___0,5___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____

(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____

(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____

(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Крупельницький Л.В. _____

Здобувач _____ _____Багрій Д.В._____

ДОДАТОК В

Графічна частина

ІНФОРМАЦІЙНА СИСТЕМА КОНТРОЛЮ НАВЧАЛЬНОГО ПРОЦЕСУ ДИСЦИПЛІН КОМП’ЮТЕРНОЇ ІНЖЕНЕРІЇ

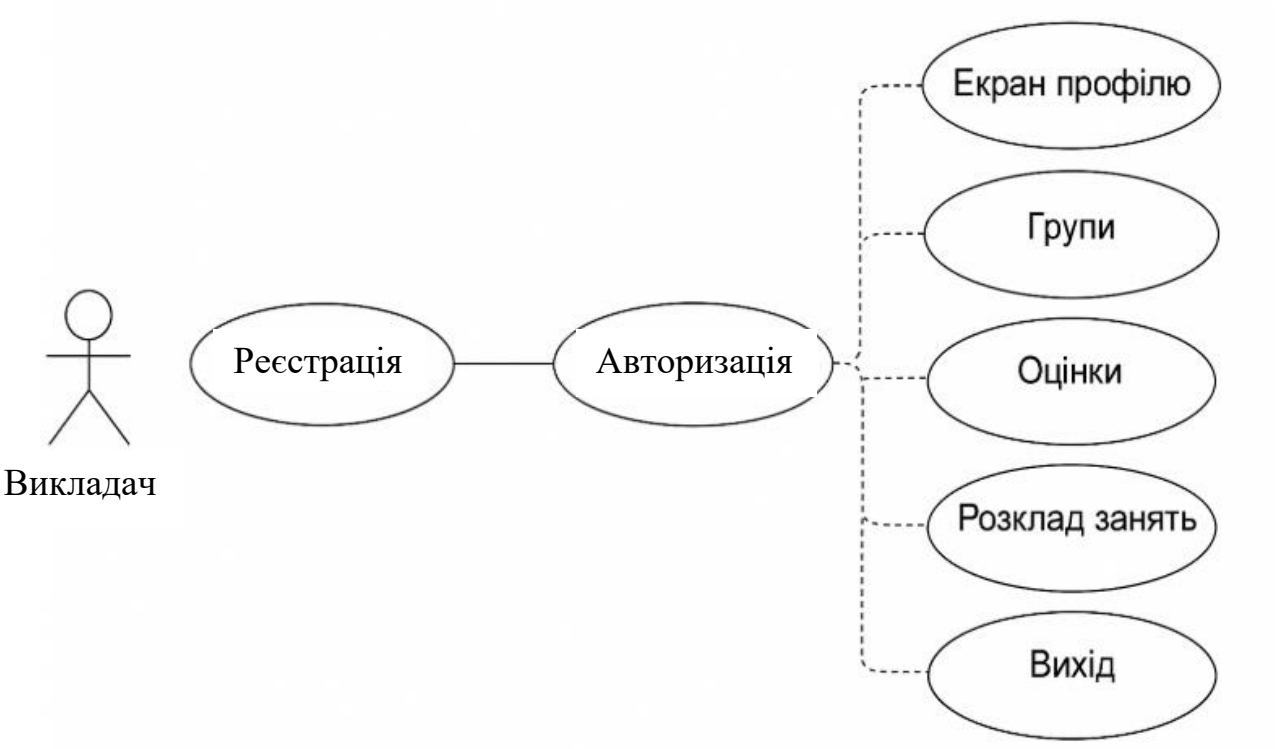


Рисунок В1 – Функціональна схема інтерфейсу вебзастосунку

ДОДАТОК Г

Лістинг програмного забезпечння

```
import {
  Body,
  Req,
  Controller,
  HttpCode,
  Post,
  UseGuards,
  UseInterceptors,
} from '@nestjs/common';
import { AuthenticationService } from './authentication.service';
import RegisterDto from './dto/register.dto';
import RequestWithUser from './requestWithUser.interface';
import { LocalAuthenticationGuard } from './localAuthentication.guard';
import JwtAuthenticationGuard from './jwt-authentication.guard';
import { User } from '../users/user.schema';
import MongooseClassSerializerInterceptor from
'../utils/mongooseClassSerializer.interceptor';

@Controller('authentication')
@UseInterceptors(MongooseClassSerializerInterceptor(User))
export class AuthenticationController {
  constructor(private readonly authenticationService: AuthenticationService) { }

  @Post('register')
  async register(@Body() registrationData: RegisterDto) {
    return this.authenticationService.register(registrationData);
```

```
  }

  @HttpCode(200)
  @UseGuards(LocalAuthenticationGuard)
  @Post('log-in')
  async logIn(@Req() request: RequestWithUser) {
    const { user } = request;
    const cookie = this.authenticationService.getCookieWithJwtToken(user.id);
    request.res?.setHeader('Set-Cookie', cookie);
    return user;
  }
  @Post('log-out')
  @HttpCode(200)
  async logOut(@Req() request: RequestWithUser) {
    request.res?.setHeader(
      'Set-Cookie',
      this.authenticationService.getCookieForLogOut(),
    );
  }
}

import { MailerService } from '@nestjs-modules/mailer';
import { Injectable } from '@nestjs/common';
import { User } from './../users/user.schema';
import { ConfigService } from '@nestjs/config';

@Injectable()
export class MailService {
```

```
    constructor(
      private readonly configService: ConfigService,
      private mailerService: MailerService,
    ) {}

    async sendEmailConfirmation(user: User, code: string) {
      await this.mailerService.sendMail({
        to: user.email,
        from: '"Jet Support Team" <support@jet.com>',
        subject: 'Welcome to Jet App! Confirm your Email',
        template: 'confirmationEmail.hbs',
        context: {
          code,
          frontendUrl: this.configService.get('FRONTEND_URL'),
        },
      });
    }
}

    import {
      Controller,
      Get,
      Query,
      Req,
      UseGuards
    } from '@nestjs/common';
    import { StudentService } from './student.service';
```

```
@Controller('student')
export class StudentController {
  constructor(private readonly studentService: StudentService) { }

  @Get()
  async getAll(@Query('groupId') groupId?: string, @Query('subjectId') subjectId?:
string) {
    return this.studentService.findAll(groupId, subjectId);
  }
}
```

```
"use client";

import { useEffect, useState } from "react";
import { TableHeader } from "../../widgets/TableHeader";
import { useGetMySubject } from "@/helpers/requests/subject/get-my";
import { TableOwn } from "../../ui/Tables/TableOwn";
import { useGetGroupStudent } from "@/helpers/requests/studet/get-group-student";

export const GroupScreen = () => {

  const { data: subject, isLoading } = useGetMySubject();
  const [selectedSubjectId, setSelectedSubjectId]: any = useState();
  const [group, setGroup]: any = useState(null);
  const [selectedGroupId, setSelectedGroupId]: any = useState();

  const {
    data: student = [],
```

```
    isLoading: isLoadingStudent,
  } = useGetGroupStudent({ groupId: selectedGroupId, subjectId: selectedSubjectId
});


  useEffect(() => {
    if (subject) {
      setSelectedGroupId(null)
      setGroup(null);
      const group = subject.find((item: any) => item.id === selectedSubjectId)?.group;
      setGroup(group);
    }
  }, [selectedSubjectId]);



  const columns = [
    {
      key: "num",
      title: "№",
    },
    {
      key: "fullName",
      title: "ПІБ",
    },
    {
      key: "email",
      title: "Email",
    },
    {
```

```
      key: "totalScore",
      title: "Сумарний бал"
    },
  ];

  return (
    <>
      <TableHeader
        setSelectedGroupId={setSelectedGroupId}
        title={"Предмети"}
        userSubject={subject || []}
        setSelectedSubjectId={setSelectedSubjectId}
        group={group}
        selectedGroupId={selectedGroupId} />
      <TableOwn
        data={student || []}
        columns={columns}
        isLoading={isLoadingStudent}
      //  dotsMenuItems={dotsMenuItems}
      />
    </>
  );
};

export default GroupScreen;

import {
  Injectable,
```

```
  NotFoundException,
} from '@nestjs/common';
import { InjectModel } from '@nestjs/mongoose';
import { Model } from 'mongoose';
import mongoose from 'mongoose';
import { Student, StudentDocument } from './student.schema';
import { Group, GroupDocument } from 'src/group/group.schema';
import { Grades, GradesDocument } from 'src/grades/grades.schema';

@Injectable()
export class StudentService {
  constructor(
    @InjectModel(Student.name) private studentModel: Model<StudentDocument>,
    @InjectModel(Group.name) private groupModel: Model<GroupDocument>,
    @InjectModel(Grades.name) private gradesModel: Model<GradesDocument>,
  ) { }

  async findAll(groupId?: string, subjectId?: string) {
    const group = await this.groupModel.findById(groupId)
    const student = await this.studentModel.find({ _id: group.student });

    const studentWithGrades = await Promise.all(student.map(async (student) => {
      console.log(student, subjectId)
      const studentGrades = await this.gradesModel.find({
        student: student._id,
        subject: subjectId,
      });
      return {
```

```
            ...student.toObject(),
            practical1: studentGrades.find((grade) => grade.title === 'practical1')?.grade,
            practical2: studentGrades.find((grade) => grade.title === 'practical2')?.grade,
            practical3: studentGrades.find((grade) => grade.title === 'practical3')?.grade,
            colloquium: studentGrades.find((grade) => grade.title === 'colloquium')?.grade,
            module: studentGrades.find((grade) => grade.title === 'module')?.grade,
            totalScore: studentGrades.reduce((acc, grade) => acc + grade.grade, 0),
          };
        }));

        console.log(studentWithGrades)


        return studentWithGrades
      }

      async findOne(id: string) {
        const point = await this.studentModel.findById(id).exec();
        if (!point) {
          throw new NotFoundException();
        }

        return point;
      }
}

    "use client";
```

```
import { use, useEffect, useState } from "react";
import { TableHeader } from "../../widgets/TableHeader";
import { Tabs } from "@mantine/core";
import { useGetMySubject } from "@/helpers/requests/subject/get-my";
import { TableOwn } from "../../ui/Tables/TableOwn";
import { useGetGroupStudent } from "@/helpers/requests/studet/get-group-student";

export const GradesScreen = () => {

  const { data: subject, isLoading } = useGetMySubject();
  const [selectedSubjectId, setSelectedSubjectId]: any = useState();
  const [group, setGroup]: any = useState(null);
  const [selectedGroupId, setSelectedGroupId]: any = useState();

  const {
    data: student = [],
    isLoading: isLoadingStudent,
  } = useGetGroupStudent({ groupId: selectedGroupId, subjectId: selectedSubjectId
});

  useEffect(() => {
    if (subject) {
      setGroup(subject.find((item: any) => item.id === selectedSubjectId)?.group);
    }
  }, [selectedSubjectId]);


  const columns = [
```

```
{
  key: "num",
  title: "№",
},
{
  key: "fullName",
  title: "ПІБ",
},
{
  key: "practical1",
  title: "Практична 1",
},
{
  key: "practical2",
  title: "Практична 2",
},
{
  key: "colloquium",
  title: "Колоквіум",
},
{
  key: "practical3",
  title: "Практична 3",
},
{
  key: "module",
  title: "Модуль",
},
```

```
        {
          key: "totalScore",
          title: "Сумарний бал"
        },
      ];

      return (
        <>
          <TableHeader  setSelectedGroupId={setSelectedGroupId}  title={"Предмети"}
userSubject={subject || []} setSelectedSubjectId={setSelectedSubjectId} group={group} />
          <TableOwn
            data={student || []}
            columns={columns}
            isLoading={isLoadingStudent}
          />
        </>
      );
    };

export default GradesScreen;
```