Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі»

(тема роботи)

Виконав: студент 2-ого курсу групи 1ІСТ-22м
(шифр групи)
спеціальності 126 – Інформаційні системи та технології
(шифр та назва спеціальності)
Олександр САЙ
(ім'я та ПРІЗВИЩЕ студента)
Керівник: к.т.н., проф. каф. АІІТ
Євген ПАЛАМАРЧУК
(науковий ступінь, вчене звання/посада, ім'я та ПРІЗВИЩЕ керівника)
«4» грудня 2023 р.
Опонент: д.т.н., проф. каф. КСУ
Володимир ДУБОВОЙ
(науковий ступінь, вчене звання/посада, ім'я та ПРІЗВИЩЕ опонента)
«8» грудня 2023р.

Допущено до захисту

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)
«11» грудня 2023 р.

Вінниця ВНТУ – 2023рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 126 – Інформаційні системи та технології
Освітня програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«20» 09 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Саю Олександру Олеговичу
(ПІБ автора повністю)

1. Тема роботи: Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі
   Керівник роботи: к.т.н., проф. каф. АІІТ Євген ПАЛАМАРЧУК
   Затверджені наказом ВНТУ від «18» 09 2023 року № 247.
2. Строк подання роботи студентом: до «05» 12 2023 року.
3. Вихідні дані до роботи: Клієнт-серверна WEB-система; курси; користувачі; групи; оцінки; критерії оцінювання; матеріали; повідомлення. Вимоги до техніки: Процесор Intel Core I5 3 GHz або вище. Об'єм оперативної пам'яті 8 Gb або більше. Жорсткий диск 120 Gb або вище.
4. Зміст текстової частини: Розглянути існуючі методи комунікації та системи, де відбувається комунікація в електронному освітньому середовищі; проаналізувати технології для розробки подібних систем; на основі розглянутих технологій та підходів розробити систему для комунікацій в електронному освітньому середовищі

5. Перелік ілюстративного (або графічного) матеріалу: Діаграма прецед тів системи, схеми алгоритмів процесів системи, схема структури б даних

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Євген ПАЛАМАРЧУК к.т.н., проф. каф. АІІТ		
5	Володимир КОЗЛОВСЬКИЙ к.е.н. проф. каф. ЕПВМ		

7. Дата видачі завдання «20» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз існуючих методів комунікації та систем, де відбувається комунікація в електронному освітньому середовищі	20.09 – 25.09	вик.
2	Аналіз технологій для розробки системи із впровадженням методів та засобів комунікації в електронному освітньому середовищі	26.09 – 30.09	вик.
3	Програмна реалізація системи	01.10 – 30.10	вик.
4	Тестування системи	31.10 – 02.11	вик.
5	Підготовка економічного розділу	03.11 – 09.11	вик.
6	Оформлення пояснювальної записки і графічного матеріалу	10.11 – 20.11	вик.
7	Попередній захист роботи	21.11	вик.
8	Остаточний захист роботи	18.12	

Студент _____ (підпис) Олександр САЙ (ім'я та ПРІЗВИЩЕ)

Керівник роботи _____ (підпис) Євген ПАЛАМАРЧУК (ім'я та ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.4

Сай О. О. Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 118 с.

На укр. мові Бібліогр.: 30 назв; рис.: 53; табл.: 6.

Метою роботи є удосконалення методів та засобів реалізації комунікації в електронному освітньому середовищі для викладачів, студентів та інших учасників навчального процесу.

Проаналізовано існуючі методи, підходи, а також інструменти для організації комунікацій в освітніх середовищах.

У даній магістерській кваліфікаційній роботі розглянуто основні аспекти розроблення клієнт-серверної системи для реалізації комунікацій в електронному освітньому середовищі та технології, які були застосовані. Проаналізовано архітектуру серверної та клієнтської частин, розглянуто вимоги до інтерфейсу користувача. В результаті роботи було представлено серверну та клієнтську частини системи.

Ключові слова: клієнт-серверна архітектура, комунікація, студент, викладач, електронне освітнє середовище.

ANNOTATION

Sai O.O. Improvement of Methods and Tools for Implementing Communication in the Electronic Educational Environment. Master's Qualification Thesis in the field of 126 - Information Systems and Technologies, Educational Program - Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2023. 118 p.

In Ukrainian language. Bibliography: 30 titles; fig.: 30; tabl.: 6.

The aim of the thesis is to enhance methods and tools for implementing communication in the electronic educational environment for teachers, students, and other participants in the educational process.

The existing methods, approaches, and tools for organizing communication in educational environments have been analyzed.

This master's qualification work explores the fundamental aspects of developing a client-server system for implementing communication in an electronic educational environment and the technologies applied. The architecture of the server and client components has been analyzed, and user interface requirements have been considered. The outcome of the work includes the presentation of both the server and client parts of the system.

Keywords: client-server architecture, communication, student, teacher, electronic educational environment.

ЗМІСТ

ВСТУП

Актуальність і важливість комунікацій в електронному освітньому середовищі не можуть бути переоцінені, оскільки вони впливають на якість навчання, розвиток учнів та ефективність викладання. Електронне освітнє середовище дозволяє студентам з різних частин світу отримувати якісну освіту, не виходячи з дому [1]. Ефективна комунікація допомагає зменшити географічні обмеження та забезпечує доступ до ресурсів та експертів з будь-якої точки світу.

Комунікація в електронному освітньому середовищі сприяє активному навчанню, де студенти можуть взаємодіяти з матеріалами, викладачами та одне з одним [2]. Це сприяє поглибленню розуміння та запам'ятовуванню матеріалу. Електронні платформи надають можливість індивідуалізувати навчання, а ефективна комунікація дозволяє викладачам і студентам адаптувати підхід до конкретних потреб та можливостей кожного учасника навчального процесу.

Ефективна комунікація сприяє залученості студентів до дискусій, обговорень та колективних проектів. Вона створює зручний механізм для вираження власних думок, задавання питань та розуміння матеріалу [3]. Комунікація у електронному освітньому середовищі дозволяє вдосконалити процес оцінювання завдяки зручній системі відстеження успішності студентів, після чого можна надати своєчасний та обґрунтований фідбек. В електронному освітньому середовищі студенти навчаються ефективно спілкуватися в онлайн, що стає надзвичайно важливим в цифровому світі та на ринку праці.

З огляду на ці аспекти, можна визначити, що комунікації в електронному освітньому середовищі є необхідним елементом сучасної освіти, що сприяє якості навчання та розвитку студентів та викладачів.

Метою даної роботи є удосконалення методів та засобів реалізації комунікації в електронному освітньому середовищі для викладачів, студентів та інших учасників навчального процесу.

Для того, щоб досягнути вищевказаної мети потрібно розв’язати низку таких **задач**:

- розглянути методи комунікації у електронному освітньому середовищі;
- здійснити порівняльний аналіз методів та аналогів систем для здійснення комунікації, визначити переваги та недоліки;
- проаналізувати технології та підходи для програмної реалізації системи для комунікації в електронному освітньому середовищі;
- на основі розглянутих підходів та технологій програмно реалізувати систему для комунікації в електронному освітньому середовищі.

Об’єктом дослідження є процес комунікації викладачів, студентів та інших учасників навчального процесу в електронному освітньому середовищі.

Предметом дослідження є методи комунікації у електронному освітньому середовищі.

Методи дослідження. Дослідження наукової літератури та існуючих систем для розуміння їхніх переваг і недоліків. Порівняння функціоналу, архітектури та інших аспектів існуючих систем. Контрольоване спостереження та тестування для вивчення причинно-наслідкових зв’язків.

Основні науково-технічні результати полягають у розробці та впровадженні нових підходів, які враховують сучасні вимоги та тенденції в галузі електронної освіти. Зокрема, впровадження чату як ефективного інструменту комунікації в електронному освітньому середовищі. Кожен навчальний курс має власний чат, що розширює можливості взаємодії. Студенти та викладачі можуть спілкуватися як особисто, так і у груповому чаті, що створює багатофункціональне середовище для взаємодії всередині кожного курсу.

Практична цінність. Проведені дослідження надають змогу впроваджувати автоматизовану систему для забезпечення комунікації в електронному освітньому середовищі у навчальних закладах. Це сприяє оптимізації взаємодії між студентами, викладачами та іншими учасниками навчального процесу.

Апробація. Результати теоретичних і експериментальних досліджень викладені у доповіді на науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації у 2023 році [4]. Результати роботи були використані в експериментальній частині електронної системи управління освітнім процесом ВНТУ JetIQ. Акт впровадження № 3/ 04.12.2023, продемонстровано в додатку Д.

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1 Актуальність та важливість ефективної комунікації в електронному освітньому середовищі

У сучасний період часу важко уникнути впливу Інтернету на всі сфери нашого життя, оскільки цей мережевий простір вже давно став необхідною складовою частиною нашого повсякденного існування. За останні роки спостерігається, що Інтернет впливає на демократизацію суспільства, відкриваючи користувачам доступ до різноманітного культурного середовища, де кожен може знаходити спільноти за інтересами та спілкуватися з людьми з усього світу [5].

Демократизація не обмежується лише розвагами та спілкуванням, але також має великий вплив на освіту. Інтернет відкриває доступ до освіти для всіх, незалежно від місця проживання, соціального статусу чи фінансових можливостей. Студенти можуть навчатися вдома, в офісі або в будь-якому іншому місці, де є доступ до Інтернету. Вони можуть вибирати з широкого спектру навчальних матеріалів та програм, які відповідають їхнім потребам і інтересам [6].

Оскільки Інтернет став невід'ємною частиною нашого життя, відмова від його використання для освіти та самовдосконалення вважається неправильною та кроком назад. Інформаційні технології в освіті є важливими для успішного розвитку сучасного суспільства. Інтернет забезпечує швидкий доступ до інформації та обміну знаннями, що є необхідною умовою для навчання та професійного розвитку.

Міжнародне суспільство вже давно визнало важливість використання Інтернету в освіті. У 2015 році ЮНЕСКО прийняла резолюцію, в якій закликала країни-члени сприяти розвитку дистанційного навчання та інших форм навчання, заснованих на Інтернеті [7].

Реалізація ефективної комунікації між викладачем та студентом в електронному освітньому середовищі є важливим завданням, яке вимагає використання різних методів та засобів. Одним із таких методів є оцінювання діяльності студента в режимі онлайн. Чітке, своєчасне та справедливе оцінювання може слугувати джерелом мотивації для студентів. Воно дозволяє їм зрозуміти свої сильні та слабкі сторони, а також визначити, над чим потрібно працювати. Це також може мотивувати студентів до більш високих досягнень і більш активного вивчення матеріалу.

Інтернет має позитивний вплив на освіту, демократизуючи її та надаючи нові можливості для навчання. Використання Інтернету в освіті є важливим для успішного розвитку сучасного суспільства.

1.2 Порівняльна характеристика існуючих методів комунікації та систем, де відбувається комунікація в електронному освітньому середовищі

Ефективна комунікація в електронному освітньому середовищі є важливою умовою для успішного навчання. Вона дозволяє викладачам і студентам обмінюватися інформацією, ідеями та думками, а також співпрацювати над спільними проектами [8].

У сфері електронної освіти існує багато методів комунікації, які можна використовувати, а також різні системи та їх аналоги для забезпечення ефективної взаємодії в електронному освітньому середовищі.

Одним із найпоширеніших та найстаріших методів комунікації в електронному освітньому середовищі є електронна пошта. Він дозволяє обмінюватися текстовими повідомленнями, а також документами, зображеннями та іншими файлами.

Даний метод використовується для спілкування між викладачами і студентами, а також для розсилки важливих повідомлень, розкладів та документів.

До переваг електронної пошти можна віднести універсальність, адже вона доступна майже для всіх користувачів з підключенням до Інтернету. Також перевагою є те, що користувачі можуть отримувати повідомлення та документи у будь-який час та з будь-якого пристрою. Поштою користуватись зручно, відправники можуть створювати довгі повідомлення та додавати великі файли, що є необхідним для кращого розуміння будь-якого матеріалу [9].

До недоліків даного методу комунікації можна віднести те, що електронна пошта не передає невербальних сигналів, таких як міміка, жести та інтонація, які можуть бути важливими для розуміння повідомлення. Також те, що відрізняє даний метод від чатів, та є недоліком, що електронна пошта є асиметричним методом комунікації. Це означає, що відправник не може бачити, коли отримувач прочитав повідомлення.

Електронна пошта виступає в ролі засобу комунікації в освітньому процесі, але не є повноцінною системою навчання, оскільки через неї лише здійснюється обмін інформацією. У ній відсутні ключові елементи навчального процесу, такі як система оцінювання, можливість участі в спільних проєктах та обговореннях.

Наступний метод комунікації в електронному освітньому середовищі – це онлайн-чати та миттєві повідомлення. Даний метод являється динамічним інтерактивним засобом електронної комунікації, який надає можливість обмінюватися текстовими та мультимедійними повідомленнями в реальному часі. Він є популярними в електронному освітньому середовищі завдяки своїй швидкості та спрощеній взаємодії [10].

Онлайн-чати та миттєві повідомлення використовуються для приватного та групового спілкування між студентами, викладачами та іншими учасниками навчального процесу. Вони також дозволяють вести діалог з користувачами з усього світу.

Позитивні аспекти використання онлайн-чатів включають синхронність, що означає можливість спілкування в режимі реального часу між відправником і отримувачем. Цей метод комунікації також сприяє швидкому та легкому обміну

інформацією. Крім того, онлайн-чати можуть бути використані для групової роботи та співпраці.

Недоліки використання онлайн-чатів включають неможливість передачі невербальних сигналів, таких як міміка, жести та інтонація, що можуть бути важливими для повноцінного розуміння повідомлення. Також може виникнути проблема з контролюванням, особливо якщо в онлайн-чаті бере участь велика кількість учасників.

В загальному, цей метод надає можливість лише для спілкування в освітньому середовищі в реальному часі, а не для проведення повноцінної навчальної діяльності.

Ще одним засобом комунікації є відеоконференції. Він дозволяє взаємодіяти учасникам в реальному часі за допомогою аудіо та відео на віддалених місцях. Цей метод особливо корисний у сфері електронної освіти, оскільки він створює можливість для віртуального спілкування та взаємодії між викладачами та студентами, незалежно від їхнього географічного розташування.

Відеоконференції застосовуються для проведення віртуальних лекцій, семінарів, засідань та інших заходів, які передбачають віддалену участь. Цей метод дозволяє взаємодіяти з іншими учасниками через відео та аудіо, а також обмінюватися інформацією та матеріалами [11].

Переваги використання відеоконференцій включають можливість відтворити атмосферу фізичних зустрічей, забезпечуючи змогу бачити та чути співрозмовників. Також цей метод дозволяє студентам та викладачам приєднуватися до занять з будь-якого місця, що особливо важливо для дистанційного навчання. Відеоконференції можуть бути записані для подальшого перегляду, що дозволяє студентам повертатися до матеріалів та лекцій.

З іншого боку, існують певні недоліки використання відеоконференцій. Наприклад, для забезпечення якісного відео та аудіо необхідне стабільне Інтернет-з'єднання. Помилки чи збій в програмному забезпеченні або обладнанні можуть вплинути на якість проведених відеоконференцій. Крім того, для участі

в них необхідні не лише пристрої для з'єднання, такі як ПК, ноутбук, планшет чи смартфон, але й наявність мікрофона та камери.

Ще одним методом комунікації між викладачами та студентами є використання електронних систем оцінювання. Ці інструменти та платформи дозволяють викладачам проводити оцінку студентів, ставити їм оцінки за виконані роботи та взаємодіяти з ними в віртуальному навчальному середовищі. Електронні системи сприяють автоматизації процесів оцінювання, зберігання та звітування щодо оцінок.

Ці системи використовуються для оцінки різних видів студентських робіт, таких як завдання, тести, лабораторні роботи та інші, і широко застосовуються в освітніх установах та при дистанційному навчанні.

Переваги використання електронних систем оцінювання включають можливість автоматизації оцінювання завдань та робіт студентів, швидке та легке виставлення оцінок та надання коментарів за допомогою електронних інструментів, а також можливість ведення детальних записів про оцінки та аналізу статистики студентських досягнень. Крім того, студенти можуть відстежувати свої успіхи та звертатися до викладачів з питаннями щодо оцінок.

З іншого боку, використання електронних систем вимагає підготовки та навчання викладачів щодо їх користування. Також для ефективного використання таких систем необхідна наявність технічних засобів та стабільного Інтернет-з'єднання.

Взявши результати дослідження проведеного компанією Pearson в 2022 році, в якому взяли участь понад 10 000 викладачів і студентів з усього світу, можемо побачити, що онлайн-форуми є найпопулярнішим методом комунікації в електронному освітньому середовищі (90%) [12]. Вони є популярними, оскільки дозволяють викладачам і студентам обговорювати різні теми, пов'язані з навчанням.

Чати є другим за популярністю методом комунікації. Вони дозволяють викладачам і студентам спілкуватися в режимі реального часу. Даний метод використовують 85% опитаних викладачів та студентів.

Електронна пошта є третім за популярністю методом комунікації (80%). Вона є популярним методом для відправлення та отримання повідомлень, надання відгуку на роботу студентів та запитування допомоги або інформації.

Відеоконференції є четвертим за популярністю методом комунікації (75%). Вони дозволяють викладачам і студентам спілкуватися один з одним в режимі реального часу за допомогою відео та звуку.

Ці дані свідчать про те, що онлайн-форуми, чати та електронна пошта є основними методами комунікації в електронному освітньому середовищі. Ці методи пропонують ряд переваг, включаючи доступність, зручність та гнучкість.

Онлайн-платформи для навчання - це спеціалізовані веб-сайти або додатки, які дозволяють викладачам і студентам навчатися та взаємодіяти один з одним в Інтернеті [13].

Дані платформи для навчання розширюють свої можливості через надання викладачам можливості створювати та публікувати різноманітний навчальний контент, такий як відео, презентації, текстові матеріали та завдання. Окрім цього, ці платформи також сприяють ефективному управлінню процесом навчання, включаючи планування уроків, надання зворотного зв'язку студентам та проведення оцінювання. Додатково, вони обладнані різноманітними інструментами для зручного спілкування між викладачами та студентами, такими як форуми, чати та відеоконференції.

Однією з найпопулярніших платформ для онлайн-навчання є Google Classroom, яка взаємодіє з іншими сервісами Google, такими як Google Drive та Google Docs [14].

За допомогою цієї платформи студенти та викладачі можуть обмінюватися інформацією через коментарі до завдань та електронну пошту. Також надається можливість проведення відеоконференцій через Google Meet, а також створення завдань, тестів та опитувань. Оцінки автоматично фіксуються у зошиті Google.

Важливо відзначити, що Google Classroom має обмеження, оскільки автоматизоване оцінювання застосовується лише до попередньо налаштованих тестів. Такий підхід може бути недостатнім, оскільки навчальний процес

включає не лише тести, але й завдання, що вимагають перевірки викладача щодо правильності виконання та відповідності всім вимогам.

Наступна у списку платформа - Classtime, спеціалізується на інтерактивних заняттях та онлайн-тестуванні [15]. Ця онлайн-платформа дозволяє викладачам та студентам обмінюватися повідомленнями та взаємодіяти під час проведення тестування. Основний акцент робиться на створенні та проведенні тестів, а також надає інструменти для їх оцінювання.

Однак слабкістю даної платформи є обмеженість в оцінці, оскільки можливість виставлення оцінок існує лише для тестів. Роботи, які потребують перевірки викладачем на відповідність завданню та правильність виконання, залишаються без оцінки, хоча вони є невід'ємною частиною навчального процесу.

Навчальна платформа N-Code є онлайн-ресурсом для вивчення IT-технологій [16]. Взаємодія між студентами та викладачами здійснюється за допомогою великого календаря, на якому відображаються розклад уроків та доступні часові інтервали для бронювання уроків з викладачем. Також, студенти мають можливість залишати коментарі до кожного уроку. Взаємодія з платформою також можлива через Telegram-бота, який відстежує всі процеси в системі.

Однак недоліком є відсутність будь-якого оцінювання роботи студента у функціоналі платформи N-Code. Ефективне та своєчасне оцінювання є важливою складовою навчального процесу та мотиваційним фактором для студентів. Надання відповідної оцінки сприяє покращенню навчальної динаміки та підтримує студентський інтерес у досягненні академічних цілей.

Інструменти розробки платформи N-Code, Front End:

- HTML
- CSS
- JavaScript
- AngularJS

Інструменти розробки платформи N-Code, Back End:

— Node.js

— Express.js

— MongoDB

1.3 Висновки до розділу

У даному розділі розглядалась актуальність та важливість ефективної комунікації в електронному освітньому середовищі. Були розглянуті такі методи як електронна пошта, онлайн-чати, відеоконференції та онлайн системи оцінювання. Всі вони є лише методами комунікації в електронному освітньому середовищі, і не можуть бути повноцінною платформою для онлайн навчання.

Були наведені результати дослідження, проведеним компанією Pearson в 2022 році, для виявлення найпопулярнішого методу комунікації в електронному освітньому середовищі.

Результати:

— Онлайн-форуми (90%);

— Чати (85%);

— Електронна пошта (80%);

— Відеоконференції (75%).

Також в розділі було проведено аналіз повноцінних навчальних платформ, таки як Google Classroom, Classtime, N-Code. При аналізі платформи N-Code були також наведені інструменти розробки. Основний недолік даних платформ – неефективна система оцінювання студентів.

2. ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ СИСТЕМИ ІЗ ВПРОВАДЖЕННЯМ МЕТОДІВ ТА ЗАСОБІВ КОМУНІКАЦІЇ В ЕЛЕКТРОННОМУ ОСВІТНЬОМУ СЕРЕДОВИЩІ

2.1 Вибір архітектури системи

Локальні застосунки включають у себе сервісні програми, системні утиліти, текстові та графічні редактори, компілятори та прості корпоративні програми. Розробка розширених корпоративних інформаційних систем передбачає, що вони не можуть бути складеними з окремих, не пов'язаних між собою компонентів.

Таким чином, сучасні розподілені інформаційні системи будуються з використанням розподілених архітектур, серед яких найбільшого поширення набули:

— Монолітна архітектура

— Клієнт-серверна архітектура

— Розподілена архітектура

— Мікросервісна архітектура

Монолітна архітектура веб-додатків, найбільш поширена та проста, передбачає, що всі компоненти, включаючи клієнтську та серверну частини, розташовані в одному модулі (рис. 2.1) [17]. Цей модуль може бути написаний на різних мовах програмування, таких як PHP, Java, Python або JavaScript.

Монолітна архітектура має свої переваги. Процес розробки та розгортання додатку є відносно простим, оскільки всі компоненти знаходяться в одному модулі. Оптимізація продуктивності відбувається завдяки ефективній взаємодії всіх компонентів, які працюють на одному сервері. Легкість обслуговування досягається через концентрацію всіх компонентів в одному модулі.

Однак монолітна архітектура має і свої недоліки. Складність масштабування виникає через концентрацію всіх компонентів в одному модулі. Розробка нових функцій може бути ускладненою через взаємозв'язок

компонентів. Ризик збоїв підвищується, оскільки збій в одному компоненті може призвести до збою всього додатку.

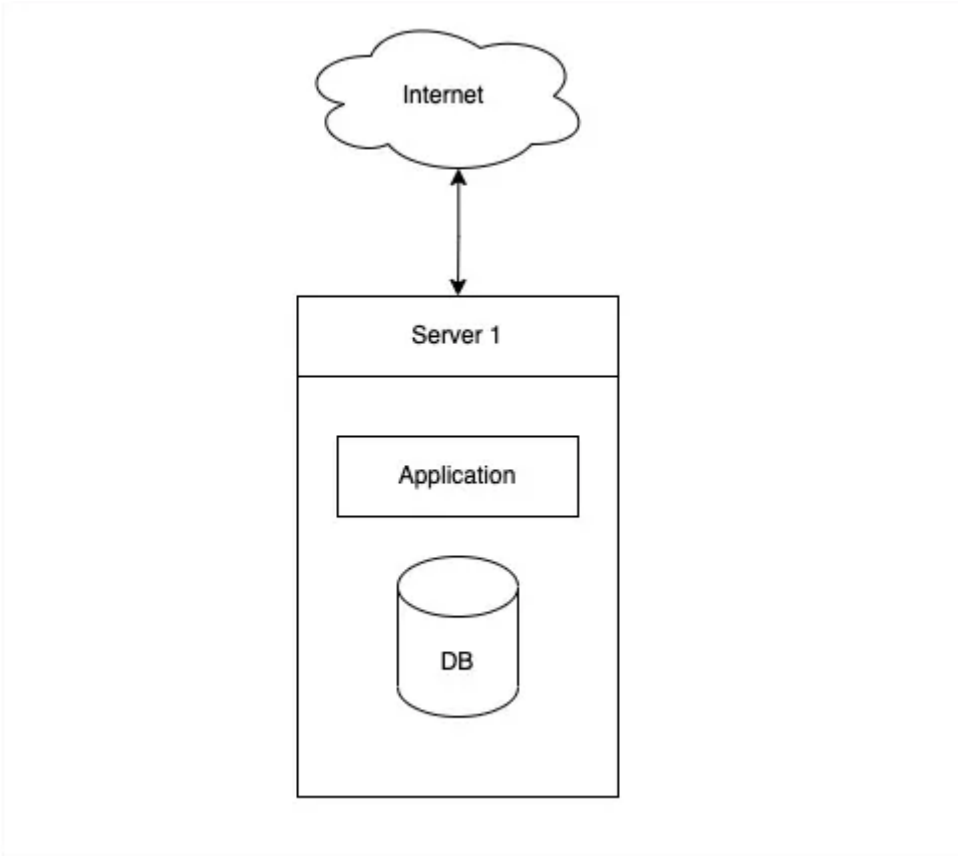

Рисунок 2.1 – Компоненти монолітної архітектури

Монолітну архітектуру рекомендується використовувати для невеликих, простих додатків, які не потребують високої масштабованості та гнучкості.

Архітектура веб-додатків типу "клієнт-сервер" визначається розділенням додатку на дві ключові частини: клієнтську та серверну (рис. 2.2) [18]. Клієнтська частина відповідає за інтерфейс користувача та взаємодію з ним і може бути реалізована на різних мовах програмування для веб-розробки, таких як HTML, CSS, JavaScript або React.

Серверна частина, в свою чергу, функціонує на сервері і забезпечує обробку запитів користувачів та відображення даних. Зазвичай вона реалізована мовами програмування, такими як PHP, Java, Python або Ruby.

Перевагами клієнт-серверної архітектури є те, що вона забезпечує розподіл відповідальності, де клієнтська частина відповідає за інтерфейс користувача, а серверна - за обробку запитів і відображення даних, спрощуючи розробку. Легкість масштабування досягається незалежністю масштабування клієнтської та серверної частини. Зручно додавати новий функціонал у одну із частин, при цьому не втручаючись в іншу.

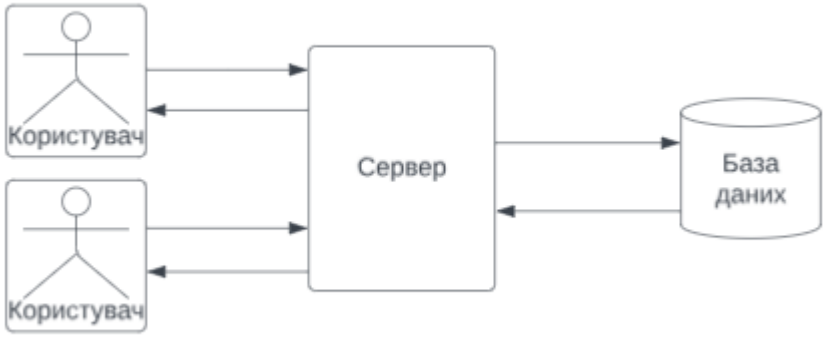

Рисунок 2.2 – Компоненти клієнт-серверної архітектури

Однак клієнт-серверна архітектура має і свої недоліки. Вона вимагає більшої складності розробки порівняно з монолітною архітектурою, і існує ризик збоїв, де поломка серверної частини призводить до паралізації клієнтської.

Клієнт-серверна архітектура рекомендована для середньорозмірних додатків з високою масштабованістю та гнучкістю, таких як електронні магазини, CMS, СУБД та веб-додатки для соціальних мереж.

Архітектура розподілених веб-додатків визначається розташуванням компонентів додатку на різних серверах, що сприяє розподілу навантаження та підвищує масштабованість системи.

Переваги такої архітектури включають високу масштабованість, оскільки додаток може зростати, додаючи нові сервери, зручність обслуговування, оскільки компоненти можуть обслуговуватися незалежно, і зменшення вартості розробки та розгортання.

Однак розподілена архітектура також має недоліки, такі як складність розробки та розгортання порівняно з іншими типами архітектур, ускладнене управління додатком і збільшений ризик випадкових збоїв через розділений характер системи.

Розподілена архітектура може бути вибором для великих, масштабованих додатків, які вимагають високої надійності та стійкості, таких як системи електронної комерції, системи управління контентом (CMS), системи управління базами даних (СУБД) та веб-додатки для соціальних мереж. До прикладів успішних розподілених веб-додатків належать такі великі корпорації, як Amazon, Google, Facebook і Netflix.

Мікросервісна архітектура представляє собою організацію додатку у вигляді невеликих і незалежних сервісів, кожен з яких відповідає за конкретну функціональність (рис. 2.3) [19].

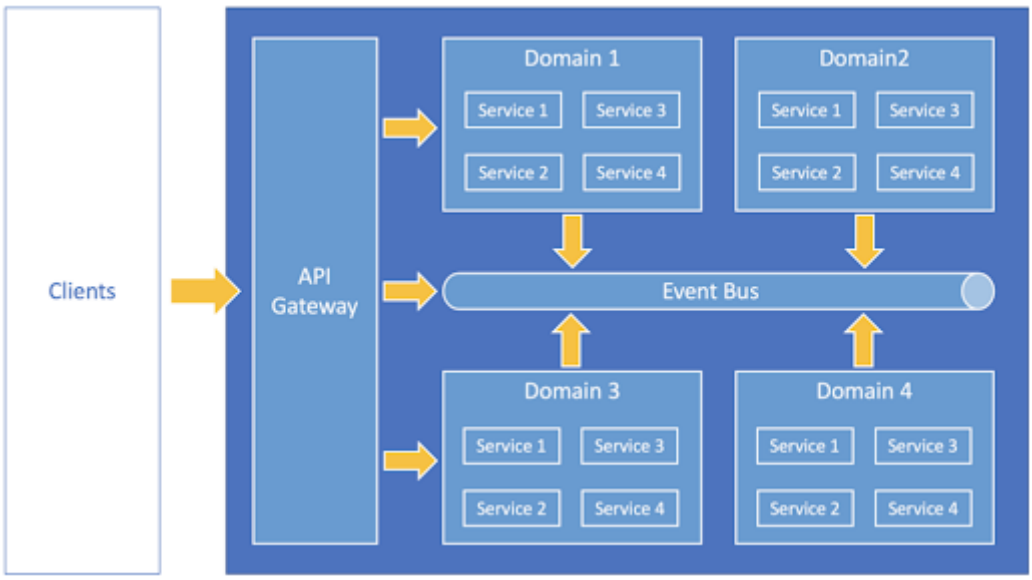

Рисунок 2.3 – Компоненти мікросервісної архітектури

Серед переваг такої архітектури можна відзначити високу масштабованість, оскільки можливе додавання або видалення сервісів для регулювання розмірів додатку. Легкість додавання нових функцій без впливу на інші сервіси і зручність обслуговування, оскільки кожен сервіс можна обслуговувати індивідуально, також вважаються перевагами мікросервісної

архітектури. Крім того, вона підвищує стійкість додатку, оскільки поломка одного сервісу не призводить до зупинки всього додатку.

З іншого боку, мікросервісна архітектура супроводжується складністю розробки та розгортання порівняно з іншими типами архітектур. Ускладнене управління додатком та збільшений ризик помилок через велику кількість сервісів розглядаються як недоліки цієї архітектури.

Мікросервісна архітектура може бути ефективним вибором для додатків, які вимагають високої масштабованості, гнучкості та адаптивності. До прикладів успішних мікросервісних веб-додатків належать такі великі компанії, як Amazon Web Services (AWS), Netflix, Twitter і Uber.

2.2 Аналіз технологій для розробки серверної частини системи

Веб-сервер - це програмне забезпечення, яке слугує доступом до веб-сторінок та інших веб-ресурсів. Веб-сервери працюють на серверах, які є комп'ютерами із постійним підключенням до Інтернету.

Коли користувач вводить адресу веб-сторінки у веб-браузер, веб-браузер відправляє запит на веб-сервер, який отримує запит і відповідає на нього, представляючи веб-сторінку користувачеві. Веб-сторінка формується з HTML-коду, який веб-браузер інтерпретує та відображає на екрані користувача, включаючи текст, зображення, відео та інші елементи. Крім того, веб-сервер може надавати доступ до різних веб-ресурсів, таких як веб-ігри, веб-додатки та веб-сервіси.

Процес розробки веб-сервера включає в себе етапи отримання вимог, визначення функціональності, масштабованості та безпеки, а також архітектурне проектування, програмування, тестування та розгортання на сервері.

Серверні скрипти, зазвичай, взаємодіють із системами керування базами даних (СКБД), такими як Oracle, MySQL, Microsoft SQL Server, MongoDB та інші. На відміну від клієнтських скриптів, які виконуються безпосередньо при

завантаженні сторінки браузером клієнта, серверні скрипти вимагають попередньої компіляції. Для їхнього виконання потрібне спеціальне програмне забезпечення, яке встановлюється на серверах.

До технологій веб-програмування, призначених для створення серверних скриптів, відносять PHP, Node.js, Java, Ruby.

PHP представляє собою скриптову мову програмування, яку широко використовують для розробки веб-сайтів та веб-додатків [20]. Ця динамічна мова може легко інтегруватися з HTML. Ця мова також є придатною для створення різноманітних проектів, включаючи графічні інтерфейси користувача (GUI).

Інтерпретується PHP веб-сервером у HTML-код, який передається на сторону клієнта. Різниця PHP та скриптової мови JavaScript полягає у тому, що користувач не бачить PHP-коду, тому що браузер отримує готовий HTML-код. З точки зору безпеки це є перевагою, але водночас погіршує інтерактивність сторінок. З іншої сторони ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

PHP має кілька переваг, таких як простий синтаксис, відкрита ліцензія та велика активна спільнота розробників, які створюють різноманітні ресурси для цієї мови. З іншого боку, PHP також має свої недоліки, такі як питання безпеки та можлива менша продуктивність та масштабованість порівняно з іншими мовами програмування, наприклад, Node.js чи Java.

Node.js представляє собою відкриту платформу, яка надає можливість розробляти веб-сервери, веб-додатки та інші програми, використовуючи JavaScript [21]. Базується Node.js на ядрі V8, яке виступає рушієм JavaScript для браузера Chrome.

Застосування Node.js охоплює створення різноманітних веб-додатків, таких як веб-сайти з динамічним контентом, веб-додатки (CMS, СУБД), чат-боти, потокове відео та штучний інтелект.

Маючи велику продуктивність, гнучкість та розширюваність, Node.js видається перспективним вибором для веб-розробки. Однак його складна архітектура може викликати труднощі в освоєнні. Порівнюючи Node.js з іншими

популярними технологіями веб-програмування, такими як PHP, Java та Ruby, можна визначити їхні основні відмінності та переваги.

Node.js, який використовує однопоточний асинхронний I/O, надає високу продуктивність та гнучкість. З іншого боку, PHP відзначається легкістю вивчення, відкритою ліцензією та великою спільнотою розробників. Java, яка використовує блокуючий I/O, має переваги в портативності, масштабованості та безпеці. Ruby, яка використовує блокуючий I/O, славиться простим синтаксисом, гнучкістю та розширюваністю.

Кожна з цих технологій має свої переваги та недоліки, тож вибір між ними залежить від конкретних потреб і вимог до проекту. Node.js, завдяки своїм характеристикам, може бути ефективним рішенням для веб-додатків, які вимагають високої продуктивності, таких як чат-повідомлення чи потокове відео. Однак, якщо важливіші легкість вивчення та велика спільнота, то розглянуті альтернативи, такі як PHP чи Ruby, також варто врахувати при виборі.

Java - це об'єктно-орієнтована мова програмування, яка часто використовується для розробки масштабованих веб-додатків і є мовою, яку є можливість використовувати на різних платформах [22].

Java використовується для створення різних типів веб-додатків, таких як веб-сайти з динамічним контентом (блоги, новинні сайти, магазини електронної комерції), веб-додатки (системи управління контентом, системи управління базами даних, ігри) і не веб-додатки (корпоративні додатки, мобільні додатки, додатки для вбудованих систем).

Java має кілька переваг, таких як портативність (використання на різних платформах), масштабованість (здатність працювати з великими і складними веб-додатками) і вбудовані функції безпеки для захисту від атак. Проте Java також має свої недоліки, такі як складність вивчення та ускладнена інтеграція з HTML.

Ruby - це динамічна, об'єктно-орієнтована мова програмування, яка широко використовується для розробки веб-додатків. За своїм призначенням

Ruby спрощує процес створення програмного забезпечення завдяки простому синтаксису та гнучкості використання [23].

Ruby може бути використаний для створення різноманітних типів веб-додатків, включаючи веб-сайти з динамічним контентом (блоги, новинні сайти, магазини електронної комерції), веб-додатки (системи управління контентом, системи управління базами даних, ігри).

Ruby володіє численними перевагами, зокрема:

— Ruby відзначається простим синтаксисом, що полегшує процес навчання та використання.

— Як динамічна мова, Ruby дозволяє розробникам легко змінювати код, надаючи їм велику гнучкість.

— Наявність бібліотеки модулів у Ruby дозволяє розробникам додавати нові функції та можливості до своїх додатків.

Проте Ruby має свої недоліки, включаючи:

— Якщо Ruby не використовується належним чином, можуть виникати проблеми з безпекою.

— У порівнянні з деякими іншими мовами програмування, такими як Node.js, Ruby може бути менш продуктивним.

— У порівнянні з деякими іншими мовами програмування, такими як Java, масштабованість Ruby може бути меншою.

Ruby є відмінним вибором для розробки веб-додатків, які вимагають простоти вивчення, гнучкості та розширюваності. Однак, для тих, хто прагне високої продуктивності, масштабованості чи безпеки, може бути розглянута альтернатива, така як Node.js або Java.

2.3 Аналіз систем керування базами даних

При виборі бази даних ключовим рішенням є вибір між реляційною (SQL) та нереляційною (NoSQL) структурами даних. Ці дві структури є життєздатними

варіантами, проте між ними існують певні суттєві відмінності, які користувачам слід враховувати при прийнятті рішення. Щодо типу, реляційні бази даних (RDBMS) визначаються як SQL, тоді як бази даних NoSQL в першу чергу асоціюються із нереляційною або розподіленою базою даних.

У визначенні мови, реляційні бази даних SQL оперують мовою структурованих запитів на основі даних (SQL) та контролюють їх. Ця мова є потужною, бо SQL є одним з найбільш універсальних та широко використовуваних інструментів, зокрема для виконання великих складних запитів [24]. Однак використання SQL може мати свої обмеження. Від користувача вимагається визначення попередньо встановлених схем для структури даних перед роботою з ними. Крім того, всі дані повинні мати однакову структуру, що може вимагати значної підготовки. Зміна структури може бути складною та руйнівною для всієї системи.

База даних NoSQL має динамічну схему для неструктурованих даних. Вони зберігаються багатьма способами, це означає, що вони можуть бути орієнтованими на документ, орієнтовані на стовпці, на основі графіків або організовані як сховище KeyValue. Ця гнучкість означає, що документи можна створювати без попереднього визначення структури. Також кожен документ може мати свою унікальну структуру. Синтаксис залежить від бази даних, і можна додавати поля по ходу.

Бази даних SQL та NoSQL мають різні підходи до масштабування. Бази даних SQL зазвичай масштабуються вертикально, що означає, що один сервер може бути модернізований для підвищення продуктивності. Це можна зробити, додавши більше пам'яті, процесорів або інших ресурсів до сервера. Бази даних NoSQL, з іншого боку, зазвичай масштабуються горизонтально, що означає, що кілька серверів можуть бути об'єднані для підвищення продуктивності. Це можна зробити, розмістивши дані на різних серверах або додавши більше серверів до кластера.

Вертикальне масштабування має ряд переваг. Воно може бути відносно простим для реалізації, і воно може забезпечити значне підвищення

продуктивності. Крім того, вертикальне масштабування може бути більш економічно ефективним, ніж горизонтальне масштабування, оскільки не вимагає додаткової інфраструктури. Вертикальне масштабування також має ряд недоліків. Воно може бути дорогим, оскільки вимагає модернізації існуючих серверів. Крім того, вертикальне масштабування може бути обмеженим, оскільки існує фізичний ліміт на кількість ресурсів, які можна додати до одного сервера.

Горизонтальне масштабування може забезпечити необмежену масштабованість, оскільки немає фізичного обмеження на кількість серверів, які можна додати до кластера. Крім того, горизонтальне масштабування може бути більш надійним, ніж вертикальне масштабування, оскільки відмова одного сервера не призведе до повної відмови системи.

Горизонтальне масштабування також має ряд недоліків. Воно може бути складним для реалізації, і воно може вимагати додаткової інфраструктури, наприклад, балансування навантаження. Крім того, горизонтальне масштабування може бути менш економічно ефективним, ніж вертикальне масштабування, оскільки вимагає додаткових серверів.

Бази даних SQL та NoSQL мають різні структури. Бази даних SQL базуються на таблицях, які складаються з рядків та стовпців. Рядки представляють записи даних, а стовпці представляють атрибути цих записів. Бази даних NoSQL, з іншого боку, мають різні структури, включаючи:

— Пари ключ-значення - це найпростіша структура баз даних NoSQL. Вона складається з пари ключів та значень, де ключ є унікальним ідентифікатором запису, а значення є даними, що зберігаються для цього запису.
— Документи - це більш складна структура баз даних NoSQL, яка схожа на таблицю баз даних SQL. Однак документи можуть містити будь-яку структуру даних, включаючи ключі, значення, масиви та об'єкти.
— Графічні бази даних - це тип баз даних NoSQL, який використовується для зберігання даних у вигляді графів. Графи

складаються з вершин та ребер, де вершини представляють об'єкти, а ребра представляють зв'язки між цими об'єктами.

— Сховища з широкими стовпцями - це тип баз даних NoSQL, який використовується для зберігання даних, які мають стовпці різної довжини. Цей тип баз даних зручний для зберігання даних, які мають багато повторюваних значень.

Приклади використання структур баз даних із різними системами та сервісами:

Бази даних SQL:

— Система обліку;

— Система управління запасами;

— Система бронювання.

Бази даних NoSQL:

— Сервіси обміну повідомленнями;

— Сервіси аналізу даних;

— Сервіси ігор.

Для аналізу із реляційних баз даних взято PostgreSQL [25]. Це реляційна система управління базами даних (СУБД) з відкритим кодом. Вона була розроблена в 1986 році і з тих пір стала однією з найпопулярніших СУБД у світі. PostgreSQL підтримує широкий спектр функцій, включаючи:

— Всі стандарти реляційних баз даних, включаючи нормальні форми і обмеження цілісності.

— Таблиці як основну структуру для зберігання даних. Таблиці можуть містити будь-який тип даних, включаючи текст, числа, дати та час.

— Ключі для ідентифікації записів. Ключ - це атрибут, який є унікальним для кожного запису.

— Нормальні форми, які можна використовувати для оптимізації ефективності баз даних.

— Транзакції, які забезпечують цілісність даних.

— Кілька мов, включаючи англійську, французьку, німецьку та японську.

PostgreSQL відзначається низкою переваг, включаючи стабільність та надійність як системи управління базами даних (СУБД). Забезпечуючи широкий спектр функцій безпеки, таких як аутентифікація, авторизація та шифрування, PostgreSQL гарантує високий рівень безпеки даних. Можливість масштабування дозволяє використовувати PostgreSQL для потреб великих та складних систем, а його ефективність полягає в швидкому виконанні запитів. Крім того, відкрита ліцензія PostgreSQL робить її доступною для використання та модифікацій безкоштовно.

PostgreSQL також має деякі недоліки, включаючи:

— Є складною СУБД, яка вимагає певного рівня знань для її використання.

— PostgreSQL не підтримує деякі функції, які доступні в інших СУБД, наприклад, в Oracle або Microsoft SQL Server.

PostgreSQL базується на реляційній моделі даних. Реляційна модель даних визначає всі дані як набір пов'язаних записів та атрибутів у таблиці. Таблиця PostgreSQL складається з рядків і стовпців. Рядки представляють записи даних, а стовпці представляють атрибути цих записів.

PostgreSQL підтримує широкий спектр типів даних, включаючи:

— Текстові типи даних: CHAR, VARCHAR, TEXT

— Числові типи даних: INT, BIGINT, FLOAT, DOUBLE PRECISION

— Дата та час: DATE, TIME, TIMESTAMP

— Геометричні типи даних: POINT, LINE, POLYGON, GEOMETRY

— Символьні типи даних: UUID, BYTEA

PostgreSQL знаходить широке використання у різноманітних сферах, включаючи системи управління базами даних (СУБД), системи управління ресурсами (СУР), системи управління бізнес-процесами (СУП), системи електронної комерції (ЕК), інтернет-додатки та ігрові сервери.

Ця популярність PostgreSQL обумовлена його здатністю забезпечувати високу продуктивність та надійність у сфері веб-додатків, а також його використанням у різноманітних системах, де важливі аспекти, такі як управління базами даних та бізнес-процесами, вимагають надійності та ефективності.

Для аналізу із не реляційних баз даних взято MongoDB. Це документо-орієнтована система управління базами даних (СУБД) з відкритим кодом. Вона була розроблена в 2009 році і з тих пір стала однією з найпопулярніших СУБД у світі [26]. MongoDB підтримує широкий спектр функцій, включаючи:

— Зберігає дані у вигляді документів, які є об'єктами JSON.
— MongoDB зберігає документи у збірниках, які є аналогами таблиць у реляційних базах даних.
— Підтримує індекси для покращення продуктивності запитів.
— Підтримує транзакції, які забезпечують цілісність даних.
— MongoDB підтримує кілька мов, включаючи англійську, французьку, німецьку та японську.

MongoDB має ряд переваг:

— MongoDB не вимагає визначення схеми для документів, що робить її більш гнучкою, ніж реляційні бази даних.
— Є швидкою СУБД, яка може швидко виконувати запити.
— Можна масштабувати для задоволення потреб великих і складних систем.
— MongoDB має відкриту ліцензію, що дозволяє її безкоштовно використовувати та модифікувати.

Недоліком даної бази даних є те, що вона є складною СУБД, яка вимагає певного рівня знань для її використання.

MongoDB базується на документо-орієнтованій моделі даних. Документо-орієнтована модель даних визначає дані як набір документів, які є об'єктами JSON. Документи у MongoDB складається з набору ключів і значень. Ключі є унікальними ідентифікаторами для значень. Значення можуть бути будь-яким типом даних, включаючи текст, числа, дати та час.

MongoDB широко використовується в різноманітних галузях, включаючи веб-додатки, інтернет-додатки, ігрові сервери та обробку великих даних. У веб-додатках MongoDB використовується для зберігання різноманітних даних, таких як дані користувачів, продукти та замовлення. Також вона застосовується в інтернет-додатках для зберігання даних чату, аналітики та транзакцій. У сфері ігрових серверів MongoDB використовується для зберігання і управління даними гравців, ігрового процесу та лідерів. Крім того, MongoDB використовується в області великих даних, таких як логи, дані сенсорів і дані машинного навчання.

У всіх цих випадках MongoDB виділяється своєю високою продуктивністю і можливістю масштабування, що робить її популярним вибором у відповідних галузях.

2.4 Аналіз технологій для розробки клієнтської частини системи

Клієнтська частина системи представляє собою важливий аспект, який забезпечує взаємодію із користувачем. Вона відповідає за відображення інформації користувачеві та обробку його введення.

Процес розробки клієнтської частини системи включає кілька етапів, починаючи з аналізу вимог, де визначаються функціональні, нефункціональні та технічні вимоги. Далі йде етап дизайну, на якому розробляється інтерфейс та архітектура клієнтської частини. Після цього відбувається реалізація коду, його тестування на відповідність вимогам і впровадження в експлуатацію.

Існують різні підходи до розробки клієнтської частини системи, які можна класифікувати за кількома критеріями. Наприклад, залежно від способу доставки користувачеві (скачування, вбудована), типу інтерфейсу (текстовий, графічний) та технології розробки (HTML, CSS, JavaScript, Java, .NET, конструктори).

Особливу увагу слід приділити вибору підходу до розробки, орієнтуючись на конкретні вимоги системи, її цілі, бюджет і терміни розробки, а також на рівень володіння та досвід розробників. Наприклад, розробка клієнтської

частини системи самостійно надає повний контроль над процесом, але може бути складною та витратною. З іншого боку, використання конструкторів дозволяє швидко створити систему, але обмежує можливості кастомізації. Кінцевий вибір залежить від конкретних обставин та потреб проекту.

У процесі створення даної системи віддається перевага індивідуальному програмуванню без використання конструкторів, оскільки наявний достатній рівень досвіду, що компенсує недоліки цього підходу, і забезпечує повний контроль над усім процесом розробки.

Перед розробкою клієнтської частини системи буде проведено детальний аналіз основних бібліотек та фреймворків: React, Vue та Angular. Цей аналіз дозволить обґрунтувати вибір конкретних технологій для забезпечення оптимальної функціональності та ефективності системи.

React є бібліотекою JavaScript, спеціально розробленою для створення інтерфейсів користувача з використанням моделі компонентів [27]. Цей підхід дозволяє розробникам легко створювати складні інтерфейси з простих, повторюваних компонентів.

Переваги використання React включають в себе високу продуктивність, легкість використання та модульність. Її основною перевагою є використання віртуального DOM, що призводить до ефективнішої роботи. React легко інтегрується з роутерами, дозволяючи зручно створювати SPA, що дає можливість працювати із сторінками без їх перезавантаження. Розробка на React базується на компонентах, що полегшує управління станом та створення коду, який можна використовувати декілька разів. Також перевагою бібліотеки є її документована структура, що сприяє швидшому вивченню та впровадженню.

Незважаючи на ці переваги, React має свої недоліки. ЇЇ складність може стати викликом для розробників без попереднього досвіду в інтерфейсній розробці. Також, React залежить від сторонніх технологій, таких як Node.js і Babel, що може призводити до ускладнень у розробці та підтримці застосунків.

React широко використовується у створенні різноманітних веб-додатків, таких як соціальні мережі, інтернет-магазини та ділові застосунки. Крім того,

відомі компанії, такі як Facebook, Instagram, Airbnb та Netflix, використовують React для розробки своїх інтерфейсів користувача.

У висновку, React є потужною та ефективною бібліотекою для розробки інтерфейсів користувача, що знаходить застосування у широкому спектрі застосунків, включаючи веб-додатки, мобільні та десктопні застосунки.

У сучасному світі розробка клієнтської частини системи вимагає від розробників використання потужних інструментів, здатних забезпечити ефективність та продуктивність. Один із таких інструментів - Vue.js, JavaScript-фреймворк, призначений для розробки інтерфейсів користувача.

Vue.js визначається своєю легкістю використання та потужною моделлю компонентів, яка дозволяє розробникам легко створювати складні інтерфейси з простих, повторюваних компонентів [28]. Цей фреймворк надає ефективність та зручність в роботі, а також відповідає сучасним тенденціям в розробці інтерфейсів користувача.

Переваги Vue.js:

- Vue.js дозволяє реалізувати інтерфейси користувача з високою продуктивністю, забезпечуючи швидку відповідь на дії користувача.
- Легка у використанні бібліотека з добре документованою структурою, Vue.js спрощує розробку та навчання розробників.
- Vue.js є модульним фреймворком, що дозволяє легко створювати та підтримувати складні інтерфейси через використання компонентів.

Недоліки Vue.js:

- Для розробників без досвіду в інтерфейсній розробці, вивчення Vue.js може бути викликом.
- Vue.js використовує сторонній код, що може ускладнювати розробку та підтримку застосунків.

Vue.js широко використовується для розробки веб-додатків, включаючи соціальні мережі, інтернет-магазини та ділові застосунки. Також він застосовується в розробці мобільних та десктопних застосунків.

Приклади Використання Vue.js:

— Gitlab;

— AliExpress;

— Baidu;

— Ant Design.

Вибір фреймворка для розробки інтерфейсів користувача залежить від конкретних вимог до застосунку. Vue є хорошим вибором для застосунків, які вимагають високої продуктивності і легкості використання.

Angular, один із найбільш визнаних JavaScript-фреймворків, надає високий рівень функціональності та забезпечує стабільність розробки [29].

Angular вирізняється своєю модульністю та великим співтовариством розробників, що дозволяє розробникам створювати складні, але добре структуровані застосунки. Цей фреймворк володіє широким набором інструментів, які полегшують процес розробки та забезпечують високу якість вихідного коду.

Переваги Angular:

— Angular є модульним фреймворком, що дозволяє розробникам легко розділяти функціонал на невеликі та повторно використовувані модулі.

— За допомогою строгого стандарту вихідного коду та широкого набору тестів, Angular забезпечує високий рівень стабільності.

— Angular надає багатий функціонал, включаючи вбудовані інструменти для роботи з HTTP, формами, маршрутизацією та іншими ключовими аспектами розробки.

Недоліки Angular:

— За всією своєю потужністю, Angular може вимагати написання більшого обсягу коду порівняно з іншими фреймворками.

— Для новачків Angular може виявитися складним у вивченні через його обширну документацію та багатий функціонал.

Angular широко використовується для розробки веб-додатків різного роду, включаючи адміністративні панелі, та інтерактивні інтерфейси. Він також добре

підходить для великих та складних систем, де модульність та стабільність мають велике значення.

Приклади використання Angular:

— Google: Google використовує Angular для розробки внутрішніх та зовнішніх веб-додатків.
— Microsoft: Angular використовується у ряді продуктів Microsoft, зокрема, у веб-інтерфейсі Microsoft Office.
— IBM: Компанія IBM обирає Angular для створення додатків.

Angular є потужним і всебічним фреймворком, який може бути використаний для розробки клієнтської частини системи будь-якого типу.

2.5 Висновки до розділу

У даному розділі виконано аналіз різних типів архітектур систем, включаючи монолітну, клієнт-серверну, розподілену та мікросервісну архітектури. Розглянуті переваги та недоліки кожного типу, з метою визначення найбільш підходящого варіанту для розробки системи.

В результаті проведеного аналізу обрано клієнт-серверну архітектуру. Це обумовлено тим, що планується розробка середньорозмірного додатку, і монолітна архітектура виявилася непридатною для цієї мети. Водночас, відмовлено від розподіленої та мікросервісної архітектур через їхні недоліки, такі як складність розробки та розгортання, ускладнене управління додатком і підвищений ризик випадкових збоїв через розділений характер системи.

Вибір клієнт-серверної архітектури має на меті оптимально поєднати ефективність та надійність для успішної реалізації поставлених завдань у контексті даного дипломного проєкту.

Також у цьому розділі були розглянуті різноманітні технології для розробки серверної частини системи, такі як PHP, Node.js, Java та Ruby. Після уважного аналізу вибір був зроблений на користь Node.js та фреймворка Express

для реалізації серверної частини. Однією з ключових переваг Node.js є його асинхронний підхід, який дозволяє ефективно обробляти багато запитів паралельно, не блокуючи інші операції. Це надзвичайно важливо для високопродуктивних застосунків з великою кількістю одночасних підключень. Використання двигуна V8 від Google Chrome забезпечує високу швидкодію виконання коду в середовищі Node.js.

Окрім того, вибір Node.js обґрунтований можливістю простоти розробки та обміну кодом між клієнтом і сервером, оскільки обидва використовують мову програмування JavaScript. Це сприяє збільшенню ефективності та зручності в процесі розробки. Планується реалізація обміну повідомленнями в чаті, для чого Node.js виявляється особливо ефективним.

У порівнянні з альтернативними варіантами, такими як Java чи PHP, Node.js демонструє вагомі переваги, такі як висока швидкодія, масштабованість та ефективність обробки багатьох одночасних підключень. Наприклад, використання Java може призводити до збільшення часу розробки, а також до виділення значної кількості ресурсів через створення нового потоку на кожен запит. Недоліки PHP, такі як проблеми безпеки та обмеження у продуктивності порівняно з іншими мовами, призвели до рішення вибрати Node.js як оптимальний варіант для даного проєкту.

У даному розділі було також розглянуто та проведено аналіз різноманітних систем керування базами даних, з урахуванням специфіки потреб проекту. В результаті вивчення вибір був здійснений на користь MongoDB, яка є NoSQL базою даних. Однією з ключових переваг NoSQL баз даних, зокрема MongoDB, для даної системи є їхня гнучкість у роботі з даними, що можуть мати різну структуру та відсутність обов'язкових полів, що є важливим у контексті проєкту.

Особливий акцент був зроблений на важливості використання бази даних NoSQL у випадках, коли дані можуть мати різні форми та характеристики. У контексті сервісу обміну повідомленнями, який планується в рамках даного проєкту, особливо важливою виявляється можливість ефективного зберігання та обробки даних з різною структурою.

Зазначено, що MongoDB широко використовується для зберігання даних чату, що підкреслює його відповідність та виключну придатність для реалізації функціоналу обміну повідомленнями в даному контексті.

Також у цьому розділі були ретельно розглянуті та проаналізовані технології, використовувані для розробки клієнтської частини системи. У процесі створення даної системи віддається перевага індивідуальному програмуванню без використання конструкторів, оскільки наявний достатній рівень досвіду, що компенсує недоліки цього підходу, і забезпечує повний контроль над усім процесом розробки.

Після детального аналізу бібліотек та фреймворків, таких як React, Vue та Angular, для реалізації клієнтської частини системи було обрано JavaScript бібліотеку React. Основні переваги React включають використання віртуального DOM, що призводить до ефективної роботи та високої продуктивності. React також легко інтегрується з роутерами, що сприяє зручному створенню односторінкових додатків, дозволяючи змінювати сторінки без перезавантаження. Розробка на React ґрунтується на концепції компонентів, спрощуючи управління станом та забезпечуючи можливість повторного використання коду.

У порівнянні з React, Vue відрізняється меншою кількістю готових бібліотек і компонентів, а також менш активною спільнотою, що може впливати на зручність пошуку рішень та обговорення проблем. Angular, навпаки, вимагає більше ресурсів для роботи, особливо на стороні клієнта, що може впливати на продуктивність додатку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ІЗ ВПРОВАДЖЕННЯМ МЕТОДІВ ТА ЗАСОБІВ КОМУНІКАЦІЇ В ЕЛЕКТРОННОМУ ОСВІТНЬОМУ СЕРЕДОВИЩІ

В області розробки програмного забезпечення велике значення приділяється створенню та підтримці функціональних, якісних і надійних програм. Цей процес використовує технології, методології та практики з різних галузей, таких як інформатика, керування проектами, математика, інженерія та інші. Розробка програмного забезпечення, подібно до інших інженерних дисциплін, розглядає питання якості, вартості та надійності. Зазначено, що деякі програми складаються з мільйонів рядків вихідного коду, які повинні правильно функціонувати в змінних умовах.

3.1 Опис і аналіз процесів та ролей учасників системи

Розроблювана система електронного освітнього середовища включає три основні ключові ролі: студента, викладача та адміністратора. Кожна з цих ролей відповідає за конкретні процеси, які можуть бути або спільними, або унікальними для кожної з них.

З метою усвідомлення всіх ключових ролей учасників та їх взаємодії з відповідними процесами наведено Use Case діаграму на рисунку 3.1. Діаграма прецедентів (англ. Use case diagram) - це графічний інструмент UML, що використовується для визначення функціональності системи з погляду користувачів. Вона ілюструє, які функції системи повинні виконуватися та які користувачі будуть їх використовувати.

Після успішної авторизації користувача в системі відбувається його перенаправлення на відповідний маршрут для отримання переліку курсів. Залежно від ролі користувача відправляються різні запити до серверної частини системи.

Рисунок 3.1 – Діаграма прецедентів системи

Для адміністратора ініціюється запит на отримання повного переліку курсів у системі. Для ролі викладач надсилається запит для отримання переліку курсів, які він веде. У динамічному параметрі запиту передається ідентифікатор викладача для точного визначення його курсів.

Для студента генерується запит на отримання переліку курсів, які призначені його навчальній групі. Запит до серверної частини містить динамічний параметр із ідентифікатором навчальної групи для точного визначення курсів для даного студента.

Схема алгоритмів отримання курсів для всіх трьох ролей зображена на рисунку 3.2.

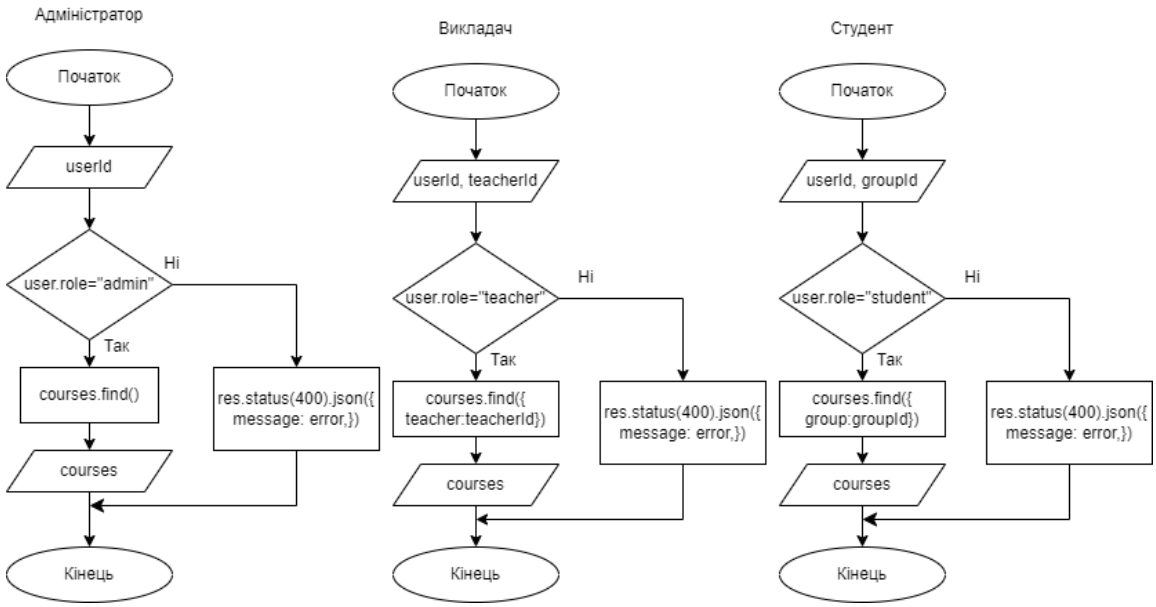

Рисунок 3.2 – Схема алгоритмів отримання курсів для всіх трьох ролей

Важливо відзначити, що для викладача та адміністратора на бічній панелі має бути доступна вкладка "Створити курс". Відкриваючи цю вкладку, користувачеві має відображатись спеціалізована форма для створення нового курсу. У формі повинні бути передбачені поля для введення назви курсу, а також випадаючі списки, що містять переліки всіх викладачів та груп у системі.

Після введення необхідних даних та натискання відповідної кнопки виконується запит до серверної частини системи. Цей запит має на меті створення нового курсу та включає в себе введені користувачем дані, такі як назва курсу, вибраний викладач та відповідна навчальна група. Схема алгоритму створення нового курсу зображена на рисунку 3.3.

У ролей викладача та адміністратора повинна бути в наявності вкладка "Студенти" на бічній панелі. Перехід на цю вкладку передбачає відображення списку всіх студентів у системі.

У випадку, коли у студента ще не визначена навчальна група, в системі повинні бути доступні поля з випадаючим списком усіх існуючих груп. Після

вибору групи і натискання відповідної кнопки, інформація про групу студента оновлюється.

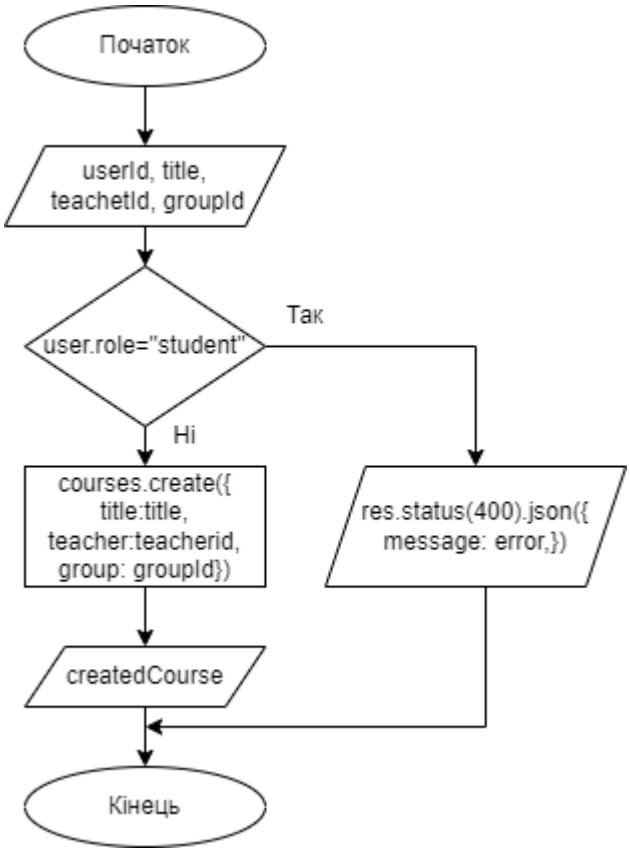

Рисунок 3.3 – Схема алгоритму процесу створення курсу

Якщо у студента вже є призначена група, поруч із нею має з'явитися поле з можливістю вибору нової групи. При редагуванні цього поля і натисканні кнопки оновлення групи студента, система відправляє запит на серверну частину, використовуючи метод для оновлення інформації про користувача. У тілі запиту передається ідентифікатор нової групи, а в динамічному параметрі — ідентифікатор студента, якому вносяться зміни.

Після оновлення початкової групи також необхідно ввести зміни у поле, що відображає кількість студентів у групі. У випадку, якщо до оновлення існувала попередня навчальна група, кількість студентів у цій групі повинна зменшитись, водночас у новій групі кількість студентів повинна збільшитись.

Схема алгоритму оновлення навчальної групи студента зображена на рисунку 3.4.

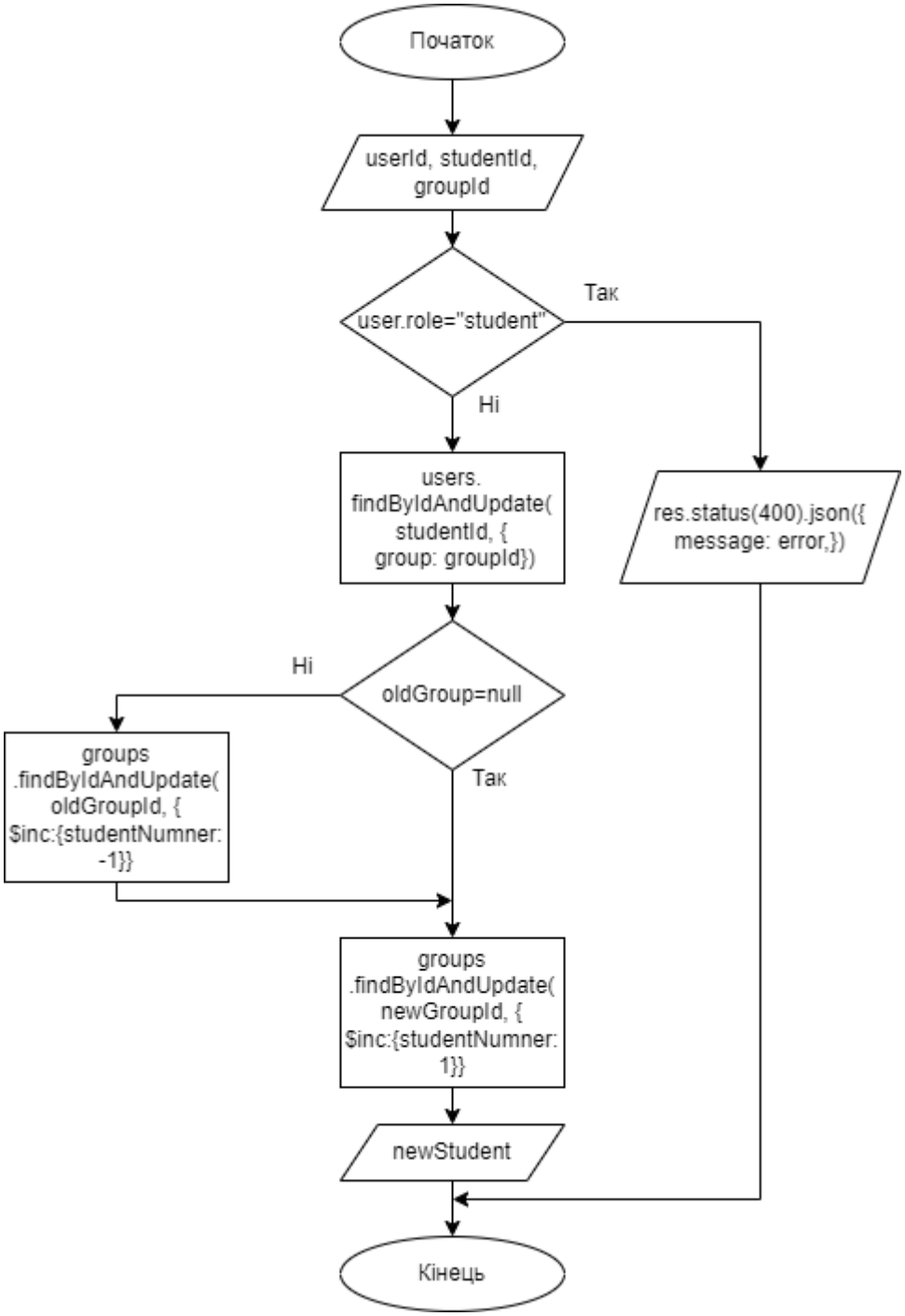

Рисунок 3.4 – Схема алгоритму оновлення навчальної групи студента

У ролі адміністратора на бічній панелі також повинна бути вкладка "Групи". Після переходу на цю вкладку відображатиметься форма для створення нової групи, що включає одне поле для введення - назву групи. Під цією формою будуть відображатися всі існуючі групи, разом з їхніми назвами та кількістю студентів у кожній з навчальних груп.

При введенні назви та натисканні кнопки для створення групи, система відправлятиме запит на сервер, передаючи лише назву нової групи в тілі запиту. При створенні групи на сервері, кількість студентів в групі буде автоматично встановлюватись на значення за замовчуванням, яке дорівнює нулю. Схема алгоритму створення навчальної групи зображена на рисунку 3.5.

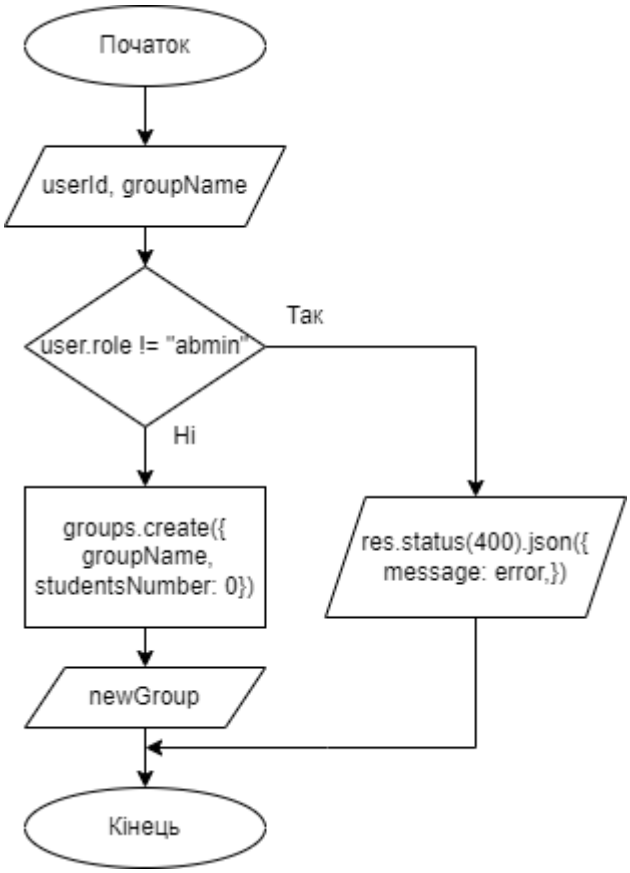

Рисунок 3.5 – Схема алгоритму створення навчальної групи

На вкладці "Мої курси" користувач має можливість натиснути на заголовок будь-якого курсу, що викликає перехід на відповідну сторінку курсу. На цій сторінці для всіх ролей буде розміщений блок із зазначенням заголовка курсу, інформації про викладача та навчальну групу, для якої призначений курс. За замовчуванням, бічна панель відкриватиметься на вкладці "Матеріали". Тут буде представлено усі матеріали, пов'язані із вибраним курсом.

Адміністратор та студент матимуть лише право перегляду матеріалів. На кожному із них є поле, яке показує кількість переглядів кожного матеріалу. Для

викладача на сторінці матеріалів буде доступна форма для створення нових матеріалів курсу, яка включатиме поля "Заголовок" та "URL", яким буде визначено місце переадресації при кліку на матеріал. Кожен матеріал буде супроводжуватись кнопками для оновлення та видалення, доступними тільки викладачу.

При активації кнопки для створення нового матеріалу, відбувається відправка запиту на сервер, в якому вказані дані нового матеріалу - заголовок, URL та ідентифікатор курсу, до якого належить матеріал. В результаті цього запиту отримується відповідь від сервера, що містить інформацію про новий матеріал, щойно створений. Схема алгоритму створення навчального матеріалу зображена на рисунку 3.6.

Після натискання на вкладку "Чати" на бічній панелі відбувається відкриття чатів, пов'язаних із курсом. Роль адміністратора має доступ лише до групового чату для всього курсу. Студент може користуватись груповим чатом для всього курсу та особистим чатом із викладачем. Викладач, з свого боку, має доступ до групового чату для всього курсу та особистих чатів із кожним студентом курсу.

Всі доступні чати для кожної ролі відображаються у лівій бічній панелі. За замовчуванням на головному блоку чату відображається груповий чат для всього курсу. Повідомлення поточного користувача розташовуються праворуч, в той час як повідомлення інших користувачів відображаються ліворуч.

Якщо користувач натиснув на інший чат, головний блок чату має показувати повідомлення відповідно до кожного користувача. Крім того, в чаті має бути можливість надсилати посилання, які будуть розпізнані системою, підсвічуватись та ставати активними для натискання.

При введенні тексту повідомлення у відповідне поле та подальшому натисканні кнопки для відправки, ініціюється запит на сервер для створення нового повідомлення. У випадку, якщо повідомлення надсилається з групового чату, в тілі запиту передаються відповідні дані, такі як ідентифікатор курсу, ідентифікатор групи, ідентифікатор користувача, що відправляє повідомлення, а

також сам текст повідомлення. У випадку надсилання індивідуального повідомлення, ідентифікатор групи не відправляється, замість цього передається ідентифікатор іншого користувача, якому призначено дане повідомлення. Схема алгоритму створення повідомлення зображена на рисунку 3.7.

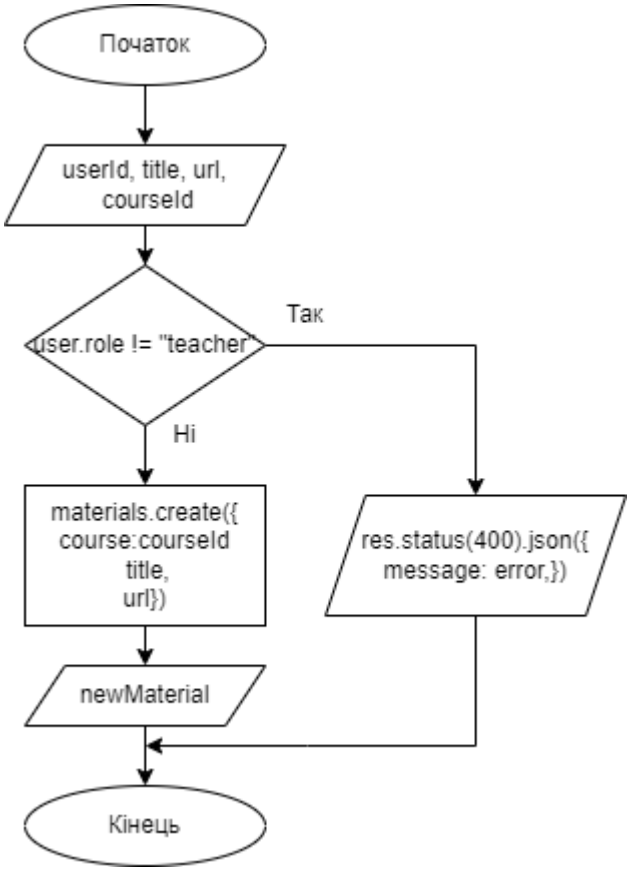

Рисунок 3.6 – Схема алгоритму створення навчального матеріалу

Для адміністратора та викладача передбачено вкладку "Критерії оцінювання" на бічній панелі. При переході на цю вкладку відображаються всі існуючі критерії оцінювання для даного курсу, що включають поля із назвою критерію та максимальною оцінкою, яку можна отримати за даним критерієм. Також відображається блок, в якому представлено загальну максимальну кількість балів за всіма існуючими критеріями оцінювання

Роль адміністратора обмежується лише переглядом вже створених критеріїв оцінювання. З іншого боку, викладач має додаткову можливість використовувати форму для створення нового критерію, де передбачені поля для

введення назви критерію та його максимального балу. Крім того, викладач може видаляти чи редагувати наявні критерії оцінювання.

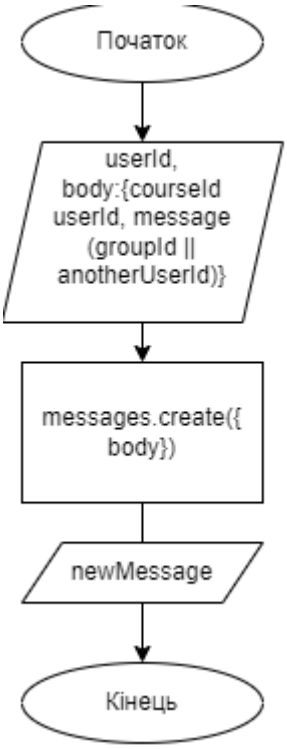

Рисунок 3.7 – Схема алгоритму створення повідомлення

Після створення нового критерію для даного курсу, автоматично формується оцінка за цим критерієм для кожного студента. За замовчуванням, ця оцінка приймає значення 0. Схема алгоритму створення критерію оцінювання зображена на рисунку 3.8.

Для всіх трьох ролей - адміністратора, викладача та студента, доступна вкладка "Результати" на бічній панелі. Після переходу на цю вкладку система відображає результати студентів даного курсу, які були оцінені за визначеними критеріями. Для ролі студента відображаються лише його власні результати для перегляду. Адміністратор може переглядати результати всіх студентів даного курсу, але не може редагувати їх. Викладач, крім перегляду результатів всіх студентів, має можливість змінювати оцінки. Натискання на кнопку зміни оцінки дозволяє викладачу внести зміни, і після натискання кнопки "Зберегти", оцінка оновлюється в системі.

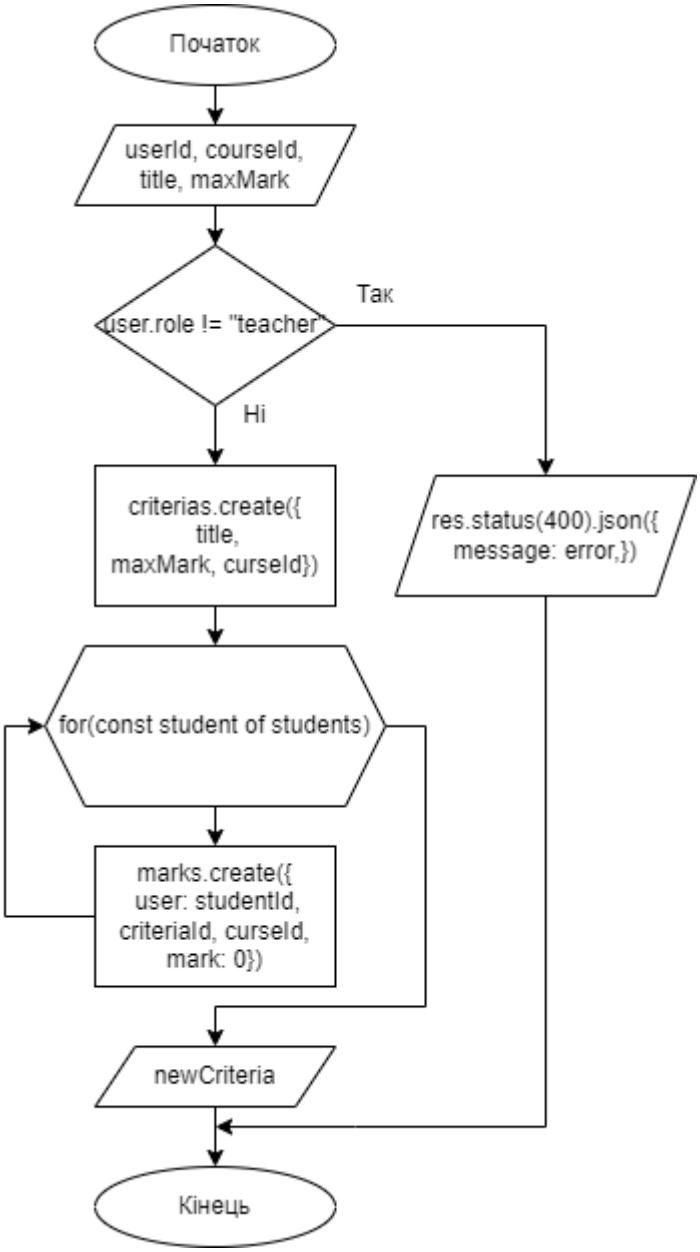

Рисунок 3.8 – Схема алгоритму створення критерію оцінювання

Після вилучення критерію оцінювання з курсу, необхідно також автоматично видалити всі оцінки всіх студентів, які були надані за цим конкретним критерієм.

3.2 Створення архітектури системи

Створення архітектури системи визначає ключовий етап у розробці клієнт-серверних систем, оскільки правильний вибір архітектури дозволяє легко масштабувати розміри системи та впроваджувати новий функціонал. Для досягнення необхідного результату слід провести аналіз різних варіантів реалізації архітектури системи. Взаємодія між клієнтською та серверною частинами здійснюється за допомогою архітектурного стилю REST.

REST (Representational State Transfer) – це архітектурний стиль для розробки розподілених гіпертекстових систем, який використовує HTTP і JSON для забезпечення інтерфейсу користувача до ресурсів системи. REST-системи часто реалізуються за допомогою веб-фреймворків, таких як Spring Boot, Django і Express.

Клієнтська частина системи розробляється з використанням JavaScript бібліотеки React, а для відправки запитів на серверну частину використовується бібліотека axios. Axios – це JavaScript-бібліотека для обміну HTTP-запитами, яка є популярною серед розробників веб-додатків завдяки своїй простоті, ефективності та надійності.

На серверній частині системи для обробки запитів використовується фреймворк Express. Express – це веб-фреймворк на основі Node.js, який використовується для створення RESTful API. Його популярність пояснюється його простотою використання, ефективністю і можливістю масштабування.

Для забезпечення взаємодії серверної частини системи з базою даних MongoDB використовується модуль mongoose. Mongoose – це JavaScript-бібліотека, спрямована на взаємодію з базами даних MongoDB, яка надає ряд корисних можливостей для оптимізації цього процесу.

До ключових переваг використання Mongoose можна віднести його простоту у використанні, оскільки він має інтуїтивно зрозумілий API, що спрощує процес розробки та навчання. Додатково, бібліотека використовує

оптимізовані алгоритми, що підвищують ефективність взаємодії з базою даних MongoDB, забезпечуючи швидкий доступ до неї.

Окрім того, Mongoose підтримує масштабованість за допомогою функцій, таких як прослуховування змін у базі даних. Це робить його відмінним інструментом для створення масштабованих веб-додатків, сприяючи зручній і надійній роботі з динамічною базою даних MongoDB.

3.3 Розробка структури бази даних системи

З метою забезпечення оптимальної функціональності системи виникає належна необхідність ефективного зберігання великих обсягів інформації в надійному сховищі. Оскільки обробка значущих масивів даних включає велику потужність системи, особливу увагу слід звертати на створення потужної і надійної бази даних.

Система повинна зберігати дані, пов'язані із користувачами та їх взаємодією в електронному освітньому середовищі, для забезпечення ефективної комунікації. З цією метою виникає необхідність в розробці структури бази даних, включаючи опис колекцій та полів у вигляді схеми, для оптимального управління та збереження важливих даних системи. На рисунку 3.9 зображена схема структури бази даних.

Оскільки база даних MongoDB є нереляційною, це означає, що не всі поля в кожній колекції є обов'язковими. Обов'язкові поля визначаються у схемі символом "*", розташованим праворуч від назви кожного поля. Також біля назви кожного поля в дужках зображено його тип.

Загальна структура бази даних включає такі колекції: "users", "messages", "materials", "marks", "groups", "criterias" та "courses". Кожна з цих колекцій грає важливу роль у забезпеченні функціональності системи та збереженні відповідних даних.

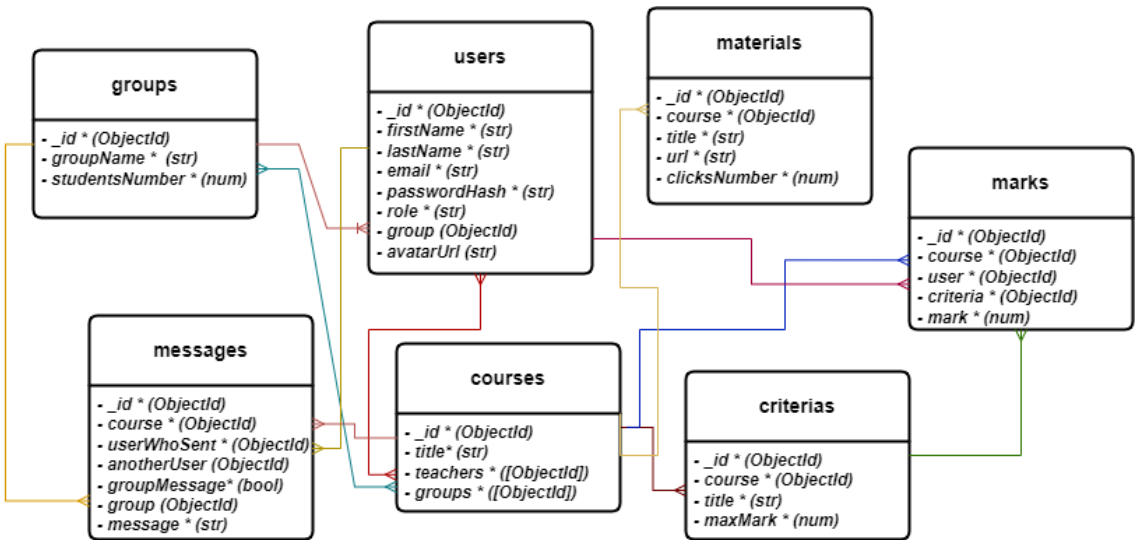

Рисунок 3.9 – Схема структури бази даних

У кожному документі колекції також присутнє обов'язкове поле "_id". Під час створення документу немає необхідності вказувати значення цього поля, оскільки база даних автоматично надає унікальний ідентифікатор для кожного об'єкту. Однак це поле було включено у схему кожної колекції для загального уявлення про структуру документів. Це сприяє кращому розумінню, як дані організовані в межах бази даних.

У колекції "users" наявні обов'язкові та необов'язкові поля. Серед обов'язкових полів - "_id", "firstName", "lastName", "email", "passwordHash", "role", а також необов'язкові поля "group" та "avatarUrl". Поле "group" визначено як необов'язкове, оскільки лише для ролі "студент" вказується навчальна група, а інші ролі не використовують це поле. Поле "avatarUrl" також вважається необов'язковим, оскільки, у випадку відсутності встановленого користувачем зображення профілю, автоматично присвоюється значення базового зображення.

Колекція "users" має вказівники, які використовуються в інших колекціях, зокрема, "messages" використовує посилання на неї двічі в таких полях як "userWhoSent" та "anotherUser", а також є посилання від "courses" та "marks" відповідно поля "teachers" та "user".

У колекції "materials" присутні лише обов'язкові поля, такі як "_id", "course", "title", "url", та "clicksNumber". Зауважимо, що інші колекції не мають

посилань на цю колекцію. Натомість, в полі "course" цієї колекції встановлено посилання на колекцію "courses".

У колекції "marks" всі поля є обов'язковими. Ця колекція включає в себе _id, course, user, criteria, та mark. Її призначення - зберігати оцінки студентів для конкретного курсу, оцінюваного за певним критерієм. Важливо відзначити, що інші колекції не мають посилань на "marks". У той же час, "marks" встановлює зв'язки з колекціями "users", "courses" та "criterias".

Колекція "criterias" включає чотири обов'язкові поля: _id, course, title, та maxMark. Ця колекція служить для зберігання інформації про критерії оцінювання в конкретному курсі. Кожен критерій має свій заголовок і максимальну оцінку, яку можна отримати за його виконання. Зазначимо, що "criterias" є джерелом посилань для колекції "marks". У свою чергу, "criterias" має зв’язок із колекцією "courses".

Колекція "courses" включає чотири обов'язкові поля для заповнення: "_id", "title", "teachers", та "groups". Особливу увагу слід приділити полям "teachers" та "groups", оскільки вони представляють масиви посилань. Поле "teachers" створено для посилання на колекцію "users", тоді як "groups" вказує на колекцію "groups". Це зумовлено необхідністю уможливлення кожному курсу мати більше одного викладача та навчальну групу. Зазначимо, що на колекцію "courses" посилаються інші колекції, такі як "materials", "marks", "criterias", та "messages".

Колекція "messages" обов'язково має заповнюватись полями: "_id", "course", "userWhoSent", "groupMessage", та "message". Також вона містить необов'язкові поля, такі як "anotherUser" та "group". Останні два поля дозволяють відзначити, відповідно, індивідуальні та групові повідомлення. Поле "anotherUser" заповнюється лише для індивідуальних повідомлень і вказує користувача, якому вони надсилаються. Поле "group" заповнюється лише для групових повідомлень і містить ідентифікатор групи, у якій відбувається обговорення. Колекція "messages" використовується для зберігання повідомлень у чатах різних курсів. Вона має посилання на такі колекції, як "courses", "users" та "groups".

Колекція "groups" складається з трьох обов'язкових для заповнення полів: "_id", "groupName" та "studentsNumber". Ця колекція призначена для зберігання інформації про назви груп та кількість студентів у кожній навчальній групі. Її дані використовуються в інших колекціях, зокрема, в "users", "courses", та "messages".

Для зберігання даних вибрана база даних MongoDB, оскільки вона повністю відповідає усім вимогам проекту та забезпечує підтримку всіх необхідних типів даних. Вибір MongoDB обґрунтований його здатністю ефективно взаємодіяти з великим обсягом різноманітних даних та забезпечувати гнучкість у відношенні до схеми даних.

3.4 Розробка клієнт-серверної частини системи

Для розробки клієнт-серверної частини системи вона повинна бути розділена на дві ключові частини: серверну та клієнтську. Серверна частина, або сервер, відповідатиме за всі процеси обробки та постачання даних у необхідному форматі. З іншого боку, клієнтська частина, або клієнт, забезпечить користувача інтерфейсом для зручного отримання інформації.

Розробка серверної частини буде здійснена з використанням Node.js та фреймворка Express. Це дозволить створити всі необхідні компоненти, враховуючи обране архітектурне рішення та забезпечити ефективну взаємодію між клієнтом та сервером. На сервері обробляються дані, які приходять із клієнтської частини, та виконуються певні запити до бази даних. Структура серверної частини системи зображена на рисунку 3.10.

Структура серверної частини системи організована на рівні компонентів, включаючи головний індекс-файл, контролери, моделі, утиліти та JSON-файл, де зберігаються налаштування сервера.

Головний індекс-файл відіграє роль центрального елемента, який приймає всі запити від клієнтської частини системи та направляє їх для подальшого

оброблення. Контролери включають файли обробників запитів для кожної колекції бази даних, а також один додатковий файл для обробки запитів на реєстрацію та авторизацію в системі. Вони містять основну логіку системи, виконуючи операції з базою даних, такі як отримання, створення, оновлення та видалення даних. Крім того, вони обробляють дані з бази даних та повертають їх клієнтській частині системи.

Моделі включають файли схем для кожної колекції бази даних, де описана структура кожного об'єкта, назви полів, їх типи, обов'язковість та значення за замовчуванням.

Утиліти включають лише один файл, який використовується для перевірки токену доступу перед тим як він відправиться у відповідний контролер. Ця утиліта визначає наявність токену доступу для виконання подальших дій. У випадку відсутності або невірності токену виконання запиту припиняється.

Для реалізації клієнтської частини системи використовується JavaScript-бібліотека React. Клієнтський інтерфейс включає систему авторизації та реєстрації, а також три ключові ролі, кожна з яких наділяє користувача конкретними повноваженнями. Серед цих ролей виділяються студент, викладач та адміністратор, кожен із яких має унікальні функціональні можливості в системі. Структура клієнтської частини системи зображена на рисунку 3.11.

Структура клієнтської частини системи включає ключові елементи, такі як файл запитів до серверної частини системи (fetch.js), файл axios, файл App, сторінки та компоненти.

У файлі запитів (fetch.js) реалізована логіка взаємодії з сервером, обробка вхідних та вихідних даних. Axios файл включає імпорт модуля axios та налаштування кожного запиту, включаючи прикріплення токену доступу користувача.

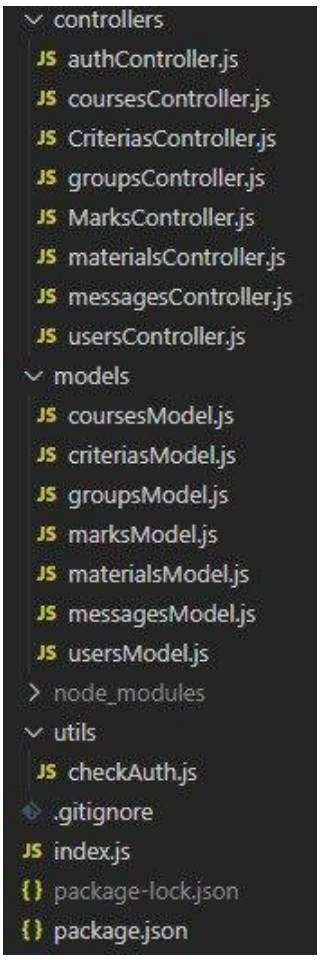

Рисунок 3.10 – Структура серверної частини системи

App файл виступає як контейнер для усіх сторінок, інтегруючи роути для забезпечення концепції односторінкового застосунку (SPA) та оновлення вмісту без перезавантаження сторінок. Сторінки представляють собою різні частини системи, такі як авторизація, реєстрація, домашня сторінка та сторінка курсу. Компоненти, розташовані у папці, використовуються для відображення даних з серверної частини та забезпечують перевикористання коду для полегшення розробки.

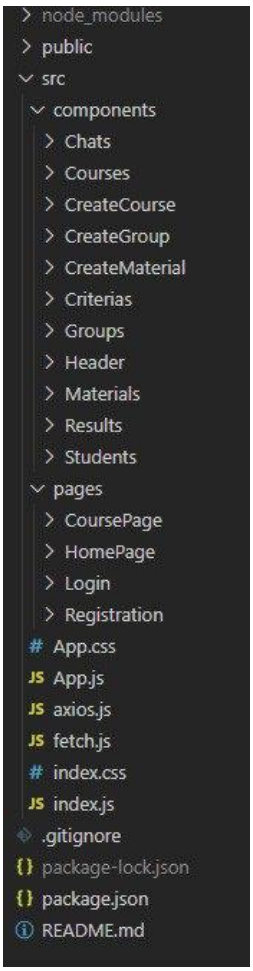

Рисунок 3.11 – Структура клієнтської частини системи

Через реєстраційну сторінку, зображену на рисунку 3.12, користувач може здійснити реєстрацію, заповнивши всі обов'язкові поля форми.

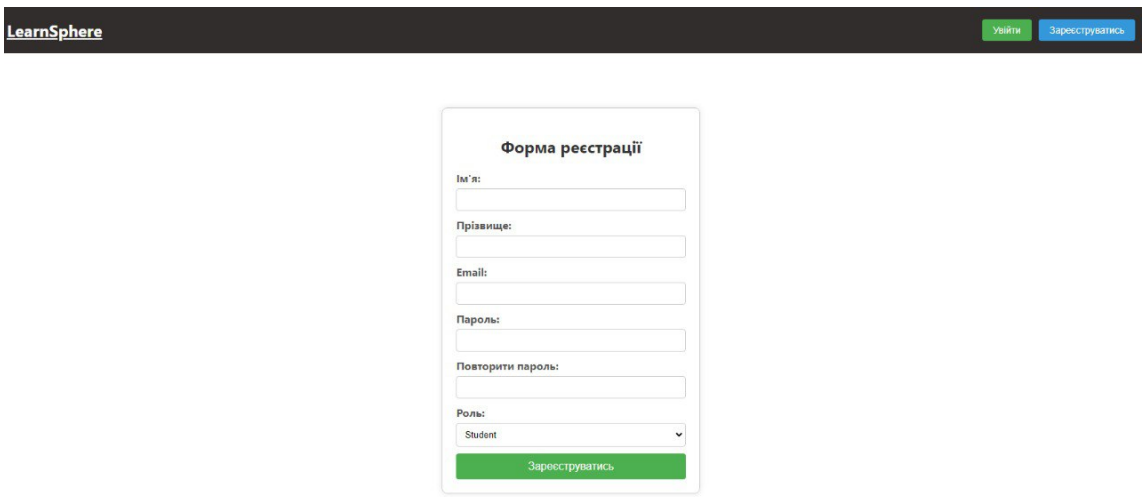

Рисунок 3.12 – Інтерфейс сторінки реєстрації

Ці поля включають ім'я, прізвище, електронну пошту, пароль, повторення пароля, обрання ролі з випадаючого списку та натискання кнопки реєстрації. Поміж усіма полями обов'язкового заповнення важливо відзначити, що при виникненні неспівпадань у полях пароля та його повторення, реєстраційний процес завершиться помилкою. Введені коректні дані призведуть до отримання позитивної відповіді від серверу, а токен доступу даного користувача буде збережено в локальному сховищі (рис. 3.13). Після чого користувачеві відкриється домашня сторінка платформи.

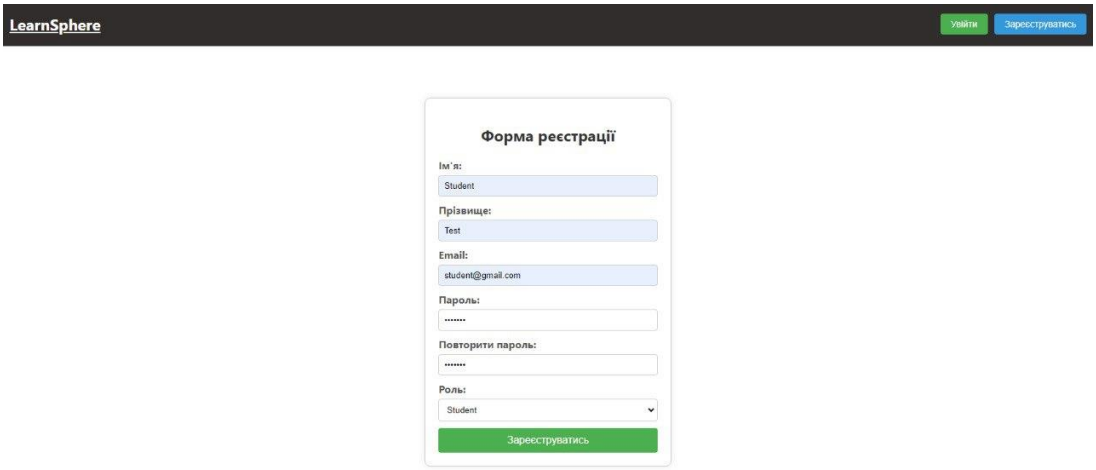

Рисунок 3.13 – Інтерфейс сторінки реєстрації із правильним заповненням полів

У випадку, якщо користувач вже здійснив реєстрацію в системі, на сторінці авторизації (рис. 3.14) він повинен ввести свою електронну пошту та пароль.

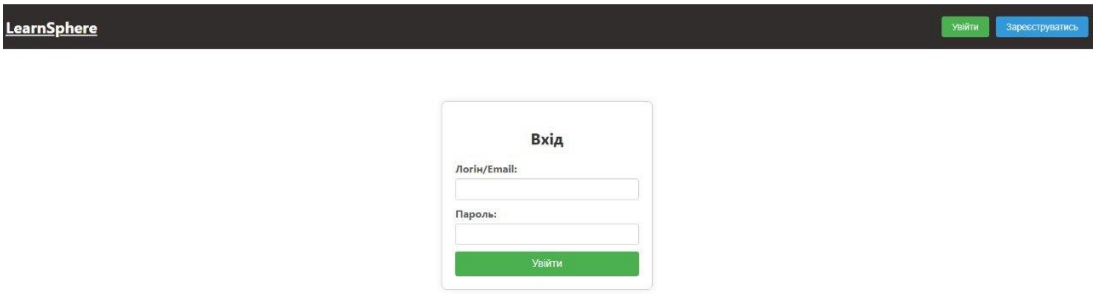

Рисунок 3.14 – Інтерфейс сторінки авторизації

На сервері проводиться валідація введених даних, де за введеною поштою з бази даних отримується користувач. Після цього порівнюється введений пароль із захешованим паролем у базі даних. У випадку співпадіння паролів, сервер повертає відповідь із даними авторизованого користувача, а також повертає токен доступу, який зберігається у локальному сховищі. Після успішної авторизації користувач отримує доступ до домашньої сторінки системи відповідно до його ролі.

Після проходження процедури авторизації, усі користувачі отримують доступ до домашньої сторінки. У правій бічній панелі домашньої сторінки кожній ролі користувача відведено свої вкладки. За замовчуванням, усі три ролі автоматично переходять на вкладку "Мої курси". Кожен курс має таку структуру: заголовок, на який можна натискати, викладач курсу, навчальна група курсу та кількості студентів у групі та на курсі.

У випадку авторизації в ролі адміністратора, за замовчуванням відображаються всі курси, що існують у системі, незалежно від викладача курсу чи навчальної групи. У бічній панелі адміністратора розташовані вкладки: "Мої курси", "Створити курс", "Групи", "Студенти". Інтерфейс домашньої сторінки ролі адміністратора зображено на рисунку 3.15.

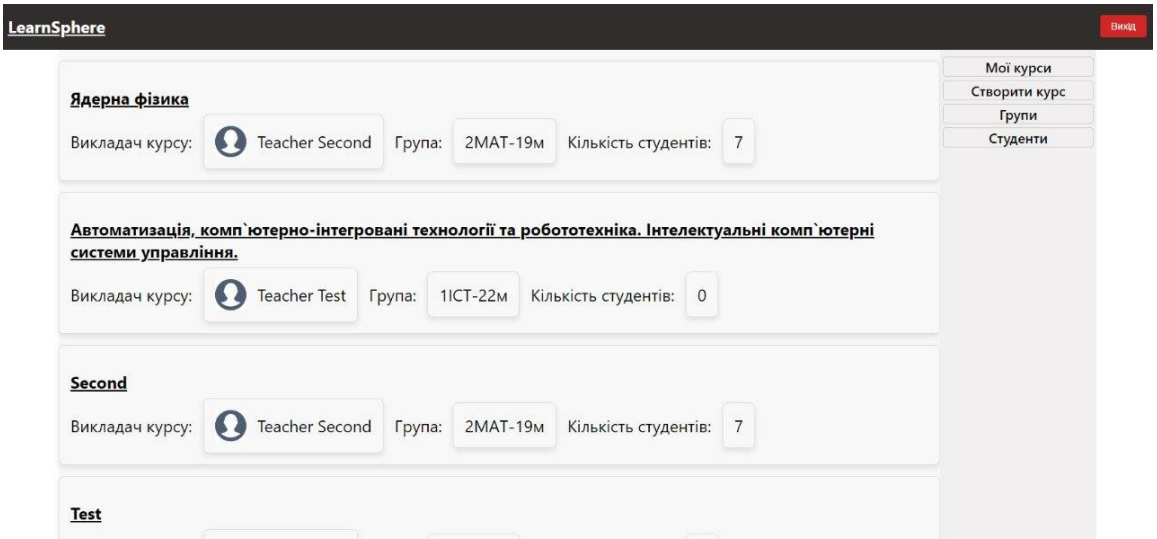

Рисунок 3.15 – Інтерфейс розділу "Мої курси" ролі адміністратора

Для ролі викладача відображаються всі курси, де він виступає викладачем, незалежно від навчальної групи курсу. Відповідно, вкладки викладача включають: "Мої курси", "Створити курс", "Студенти". Інтерфейс розділу "Мої курси" ролі викладача зображено на рисунку 3.16.

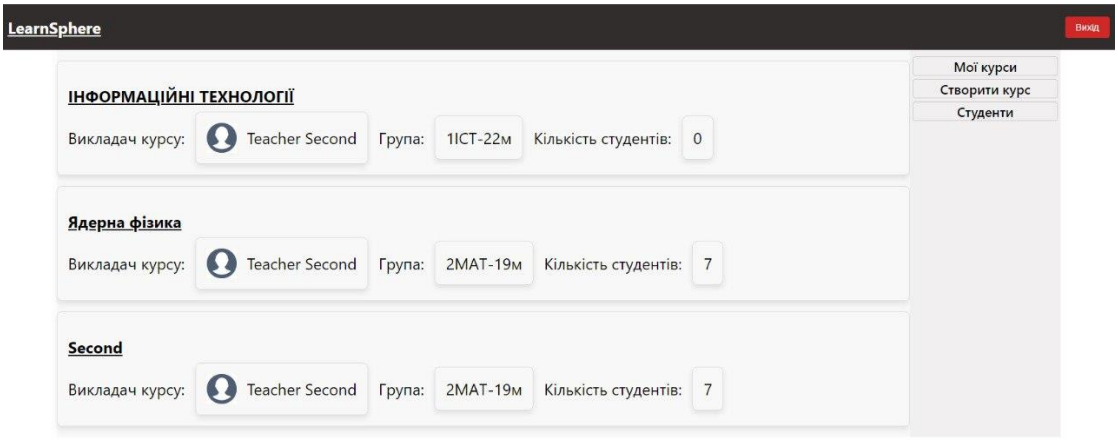

Рисунок 3.16 – Інтерфейс розділу "Мої курси" ролі викладача

У випадку авторизації в ролі студента на основній панелі відображаються курси, в яких навчальна група відповідає групі цього студента. На бічній панелі у студента лише одна вкладка "Мої курси". Інтерфейс розділу "Мої курси" ролі студента зображено на рисунку 3.17.

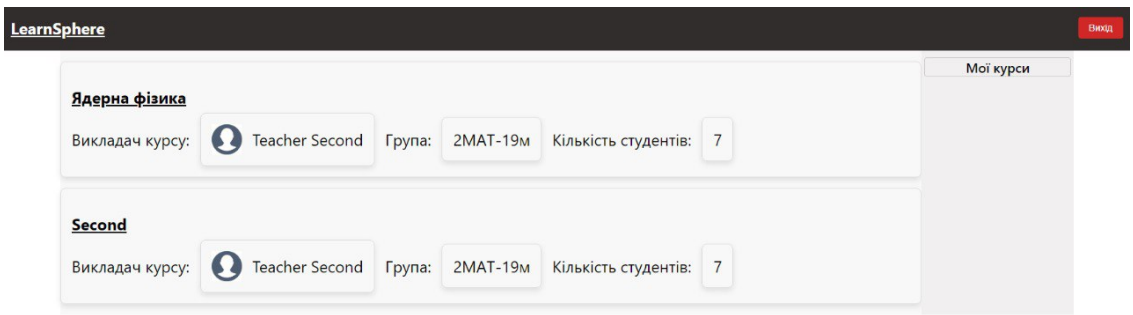

Рисунок 3.17 – Інтерфейс розділу "Мої курси" ролі студента

У ролі адміністратора та викладача у бічній панелі доступна вкладка "Створити курс". Перехід на цю вкладку призводить до динамічної зміни контенту домашньої сторінки без перезавантаження сторінки, відповідно до концепції SPA. На головній панелі з'являється форма для створення нового курсу (рис. 3.18).

Ця форма містить три поля для заповнення. Поле назви курсу вимагає ручного введення, тоді як поля викладача та групи курсу можна вибрати з випадаючих списків, де перераховані доступні викладачі та групи на платформі. Після введення всіх необхідних даних, користувач натискає відповідну кнопку для створення курсу.

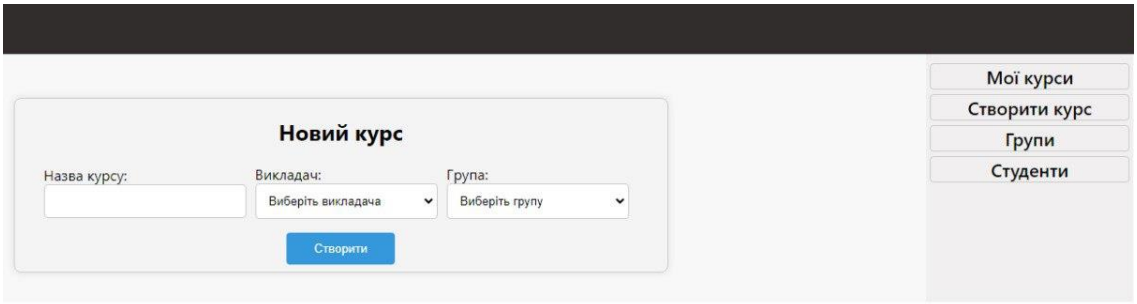

Рисунок 3.18 – Інтерфейс розділу "Створити курс "

Після цього запит відправляється на серверну частину. Після успішної валідації запиту та створення відповідного запису в базі даних, на клієнтську частину надходить відповідь. При поверненні до вкладки "Мої курси" новий курс також автоматично відображається серед доступних курсів.

У ролі адміністратора на бічній панелі наявна вкладка "Групи", перехід на яку призводить до динамічної зміни контенту домашньої сторінки без перезавантаження. На головній панелі з'являється форма для створення нової групи, яка складається лише із одного поля - назви групи (рис. 3.19).

Поле, що відповідає за кількість студентів у групі, автоматично заповнюється значенням за замовчуванням під час відправлення запиту на сервер. Коли сервер отримує запит із назвою нової навчальної групи, спочатку проводиться валідація запиту. Потім створюється новий запис в базі даних, при

цьому значення кількості студентів у групі встановлюється як 0 (значення за замовчуванням). Після успішного створення групи, на сервері формується відповідь для клієнта.

Під формою для створення групи відображаються всі існуючі групи на платформі. Кожна група представлена назвою та кількістю студентів у ній. Після створення нової групи, вона автоматично додається до списку існуючих груп.

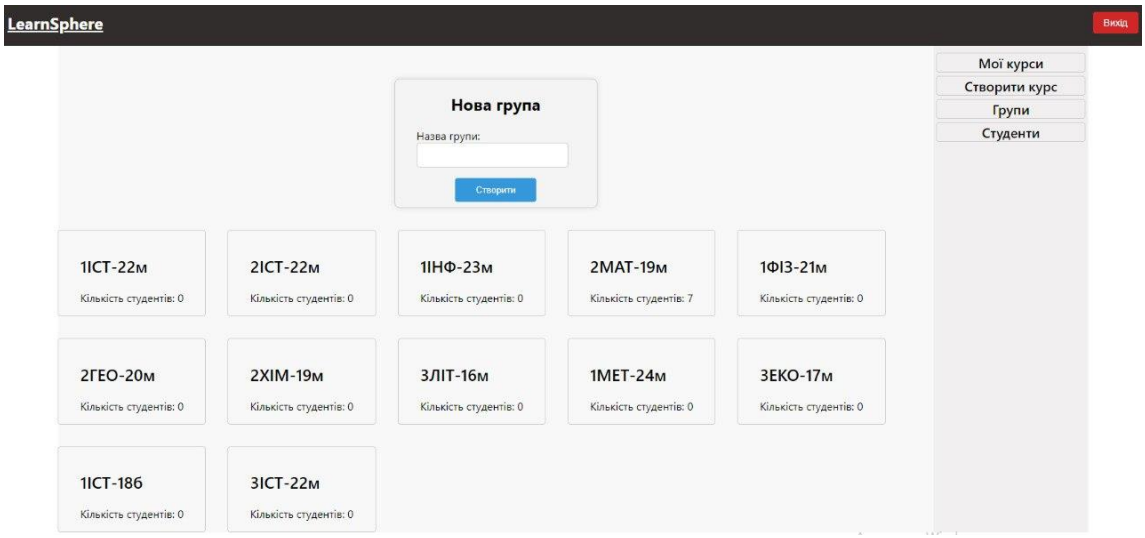

Рисунок 3.19 – Інтерфейс розділу "Групи"

У ролі адміністратора та викладача на бічній панелі існує вкладка "Студенти", перехід на яку призводить до динамічної зміни контенту домашньої сторінки. На головній панелі відображається список усіх студентів платформи (рис. 3.20).

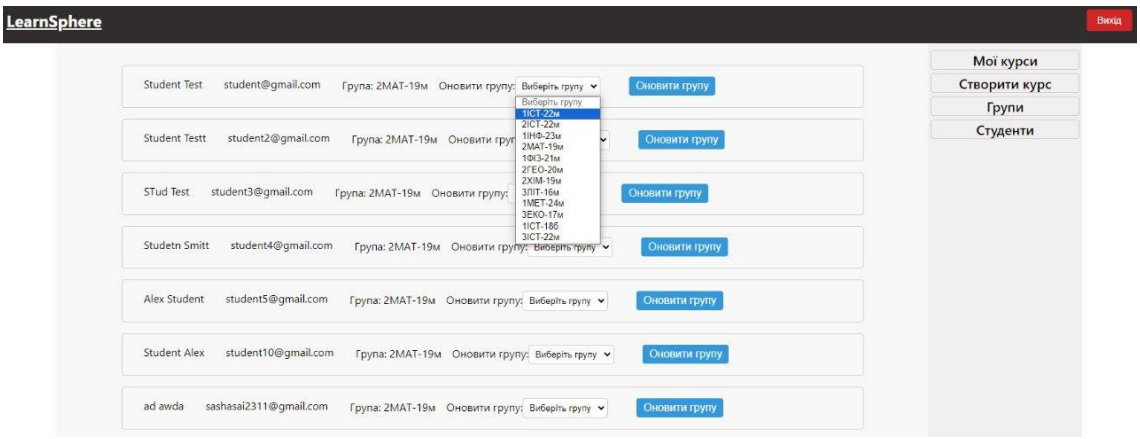

Рисунок 3.20 – Інтерфейс розділу "Студенти"

Кожний блок студента містить інформацію про ім'я, прізвище, електронну пошту та групу (якщо вона вже призначена). Крім того, доступний випадаючий список для призначення або оновлення групи студента.

Обравши групу із випадаючого списку та натиснувши відповідну кнопку, клієнтська частина надсилає запит на сервер. Після проходження валідації запиту, сервер знаходить студента за ідентифікатором і оновлює поле "навчальна група" на обрану. Після оновлення документа користувача, сервер визначає документ минулої групи, зменшує поле "кількість студентів" у цій групі на одиницю. Далі знаходиться документ нової групи, та поле "кількість студентів" збільшується на одиницю. Після цього сервер надсилає відповідь на клієнт, і відповідний користувач оновлюється.

На вкладці "Мої курси" для кожної ролі надається можливість натискати на заголовок будь-якого курсу, що відкриває головну сторінку обраного курсу. За замовчуванням для всіх ролей на головній панелі відображається сторінка матеріалів поточного курсу. У верхній частині основної панелі представлено інформаційний блок про курс, включаючи назву курсу, викладача та навчальну групу.

Для ролей адміністратора та студента відображаються всі матеріали курсу, доступні лише для перегляду (рис. 3.21). Блок матеріалу включає назву, яку можна натискати, та кількість переглядів. У ролі викладача курсу є розширені можливості (рис. 3.22): він бачить список всіх матеріалів та має форму для

створення нового матеріалу з полями "заголовок" та "посилання". Викладач також має кнопки для редагування та видалення матеріалу на кожному створеному матеріалі. При видаленні матеріалу він безповоротно видаляється з системи.

При редагуванні матеріалу назва та посилання відображаються в полях для редагування, дозволяючи внести зміни, і після натискання відповідної кнопки для оновлення, запит відправляється на сервер. Після валідації на сервері матеріал знаходиться за ідентифікатором у базі даних та оновлюється. Після оновлення сервер надсилає відповідь на клієнт.

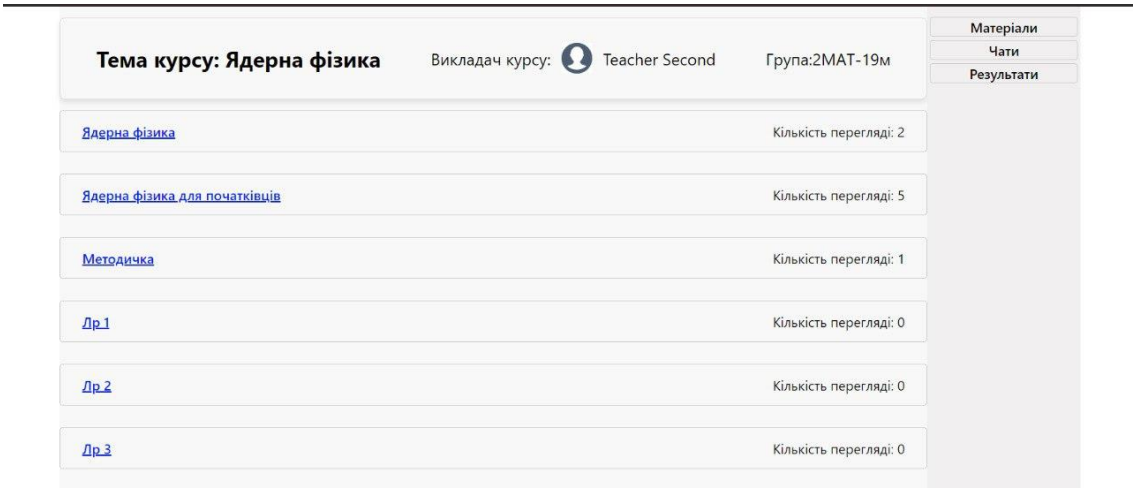

Рисунок 3.21 – Інтерфейс розділу "Матеріали" для ролей адміністратора та студента

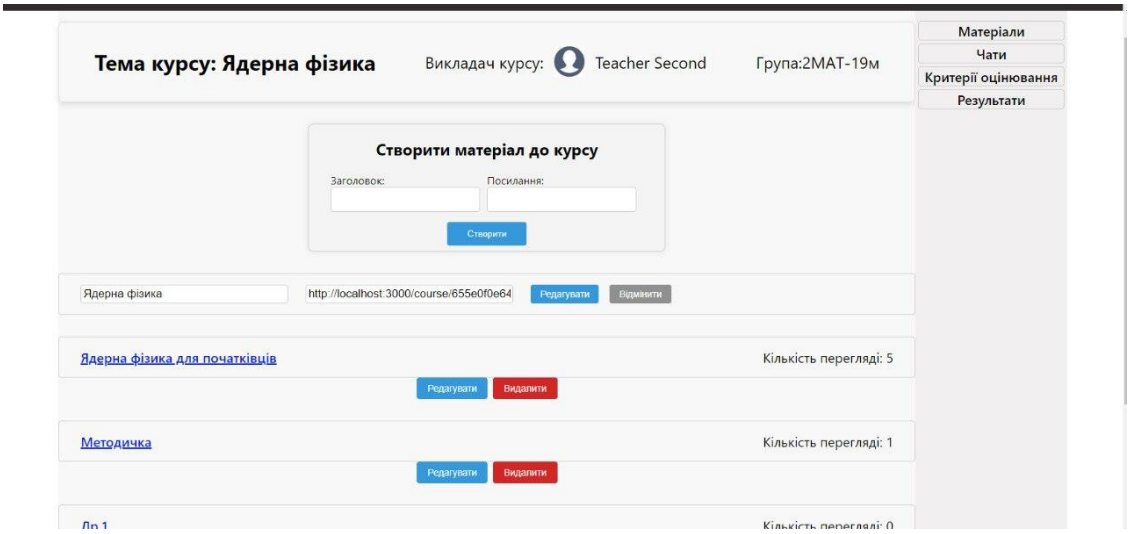

Рисунок 3.22 – Інтерфейс розділу "Матеріали" для ролі викладача

Усі три ролі мають доступ до вкладки "Чати" на сторінці курсу, розташованої в боковій панелі. Після натискання на цю вкладку контент сторінки курсу динамічно змінюється, відкриваючи чат для даного курсу. Чат включає ліву бічну панель, на якій відображаються всі доступні чати для цього курсу.

У випадку індивідуальних чатів, в лівій бічній панелі вказані ім'я та прізвище користувача, з яким пов'язаний цей чат. У випадку групових чатів, назва чату відповідає назві групи. Основна панель чату відображає всі повідомлення залежно від обраного чату.

У групових чатах відображаються всі повідомлення всієї групи, тоді як у індивідуальних чатах показуються повідомлення лише двох учасників чату. Повідомлення від користувача, який переглядає чат, розміщуються праворуч основної панелі, а повідомлення інших користувачів — ліворуч.

У нижній частині основної панелі розташоване поле для введення повідомлення, і після натискання кнопки "Відправити", повідомлення миттєво з'являється в чаті. Також є можливість відправляти посилання в чат, які автоматично розпізнаються системою та виділяються, що надає можливість натискати на них.

У ролі студента доступні індивідуальні чати з викладачем курсу та групові чати з усією групою (рис. 3.23). Викладач має доступ до індивідуальних чатів з усіма студентами курсу та групового чату (рис. 3.24). Для ролі адміністратора доступний тільки груповий чат.

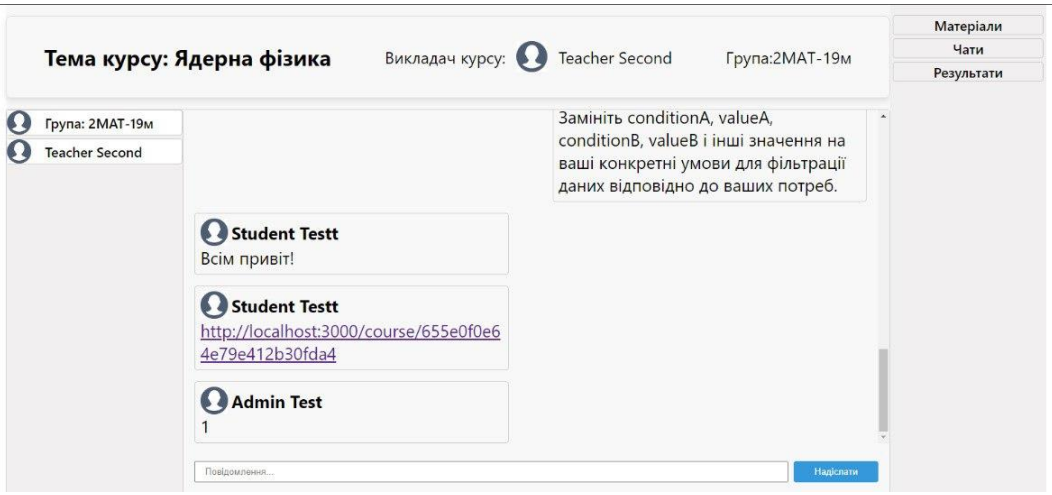

Рисунок 3.23 – Інтерфейс розділу "Чати" для ролі студента

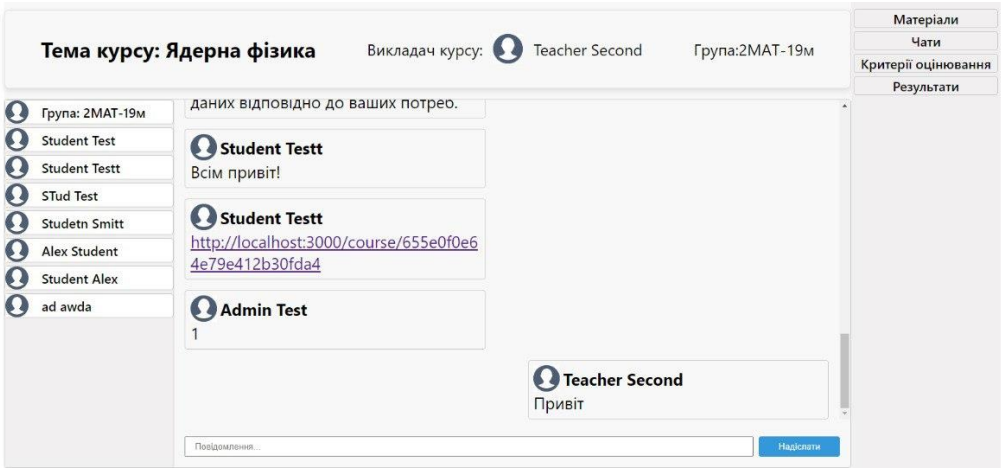

Рисунок 3.24 – Інтерфейс розділу "Чати" для ролі викладача

Для ролі адміністратора та викладача доступна вкладка "Критерії оцінювання" на бічній панелі сторінки курсу. Перейшовши на цю вкладку, контент сторінки курсу динамічно змінюється, відображаючи всі критерії оцінювання для даного курсу (рис. 3.25). Критерії оцінювання представляють собою пункти, за якими будуть проставлятися оцінки кожному студенту у цьому курсі.

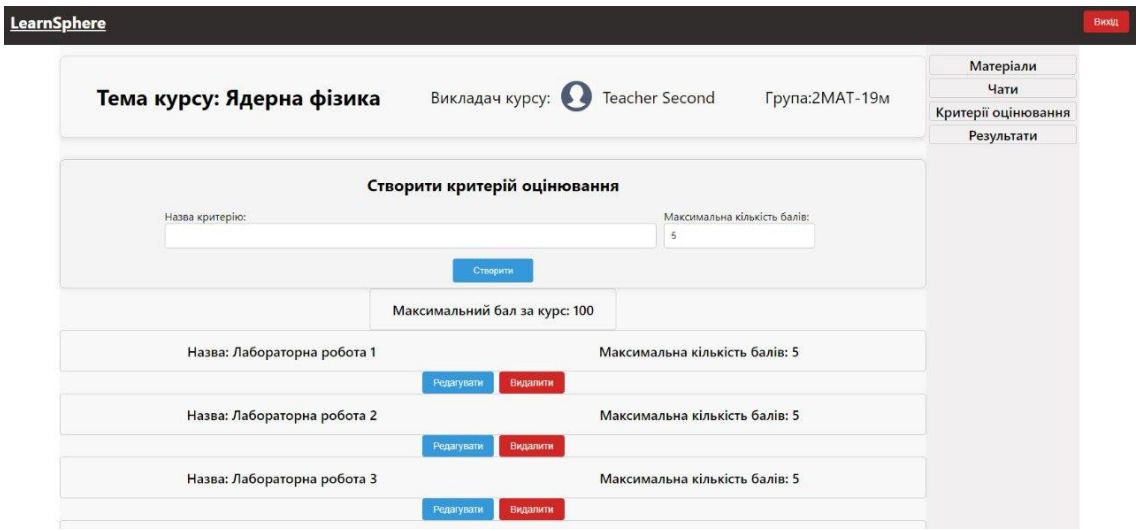

Рисунок 3.25 – Інтерфейс розділу "Критерії оцінювання" для ролі викладача

Кожен блок критерію містить назву критерію та максимальний бал, який може бути отриманий за цей критерій. Для ролі адміністратора доступний лише перегляд критеріїв курсу. У випадку ролі викладача на основній панелі відображається форма для створення нового критерію оцінювання для даного курсу. Поля для заповнення в цій формі включають назву критерію та максимальну оцінку.

Після відправки даних на сервер, проходження валідації та створення нового документу в базі даних, новий критерій з'являється у списку критеріїв. Під формою для створення критерію розташований блок, що відображає загальну максимальну кількість балів для цього курсу в залежності від усіх критеріїв. Крім того, викладач має кнопки на кожному критерії для його редагування та видалення.

При створенні нового критерію, разом із створенням документу на сервері, також генеруються оцінки за цей критерій для всіх студентів курсу, встановлюючи оцінку 0 (значення по замовчуванню).

Для всіх користувачів на бічній панелі головної сторінки курсу доступна вкладка "Результати". Перейшовши на цю вкладку, вміст динамічно змінюється, і відображаються результати студентів для даного курсу, відповідно до визначених критеріїв оцінювання.

Кожен блок результату містить інформацію про студента, включаючи його зображення, ім'я, прізвище, електронну пошту та назву групи. Під цим блоком інформації розташований блок з результатами для даного студента. В цьому блоці відображаються всі критерії оцінювання, а поруч із кожним критерієм вказана оцінка студента та максимальна оцінка, яку він може отримати за цей критерій.

У нижній частині блоку результатів розташовано загальний результат для даного студента, який представляє собою суму зароблених балів та суму балів за всі критерії, за якими він оцінюється. Для ролі студента доступні тільки його власні результати, інші студенти не відображаються (рис. 3.26).

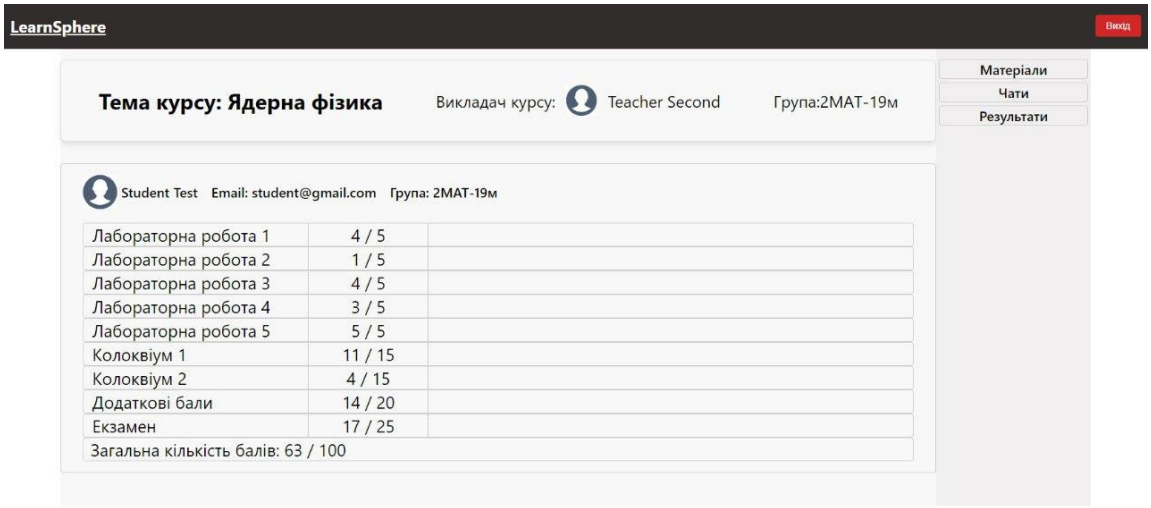

Рисунок 3.26 – Інтерфейс розділу "Результати" для ролі студента

Адміністратор має доступ до результатів всіх студентів, проте лише для перегляду. Викладач може переглядати результати всіх студентів курсу, а також має можливість оновити оцінку, використовуючи відповідну кнопку, після чого оцінка негайно оновлюється (рис. 3.27).

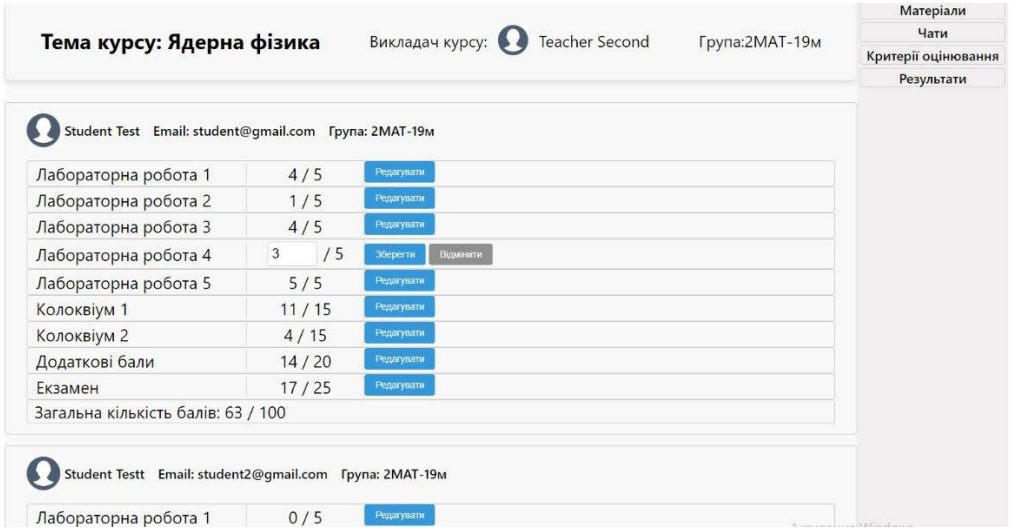

Рисунок 3.27 – Інтерфейс розділу "Результати" для ролі викладача

3.5 Висновки до розділу

У даному розділі було проведено детальний аналіз та опис основних етапів розробки клієнт-серверної системи, спрямованої на забезпечення ефективної комунікації в електронному освітньому середовищі. Робота включала в себе розгляд процесів та ролей учасників системи, представлених у вигляді діаграм прецедентів.

Основними ролями у системі були визначені студент, викладач та адміністратор, а їх взаємодія та процеси були ілюстровані на діаграмах прецедентів. Алгоритми всіх процесів системи були детально описані та продемонстровані. Особлива увага приділялась архітектурі системи, яка була реалізована з використанням архітектурного стилю REST для взаємодії між клієнтською та серверною частинами системи.

Використано бібліотеку axios для відправки запитів із клієнтської частини, фреймворк Express для обробки запитів на сервері та модуль mongoose для взаємодії серверної частини з базою даних MongoDB. Детально розглянуто структуру бази даних системи, включаючи колекції "users", "messages", "materials", "marks", "groups", "criterias" та "courses", а також зв'язки між ними.

Для реалізації серверної частини системи використано Node.js та фреймворка Express, що дозволило створити всі необхідні компоненти для ефективної взаємодії між клієнтом та сервером. Структура серверної частини була представлена компонентами, такими як головний індекс-файл, контролери, моделі, утиліти та JSON-файл, що містить налаштування сервера.

У контексті реалізації клієнтської частини використовувалась JavaScript-бібліотека React. Структура клієнтської частини системи включає ключові елементи, такі як файл запитів до серверної частини системи (fetch.js), файл axios, файл App, сторінки та компоненти. Були описані три ключові ролі системи (студент, викладач, адміністратор) разом з інтерфейсами та функціоналом кожної ролі.

Результатом цієї роботи стало створення ефективної системи для електронного навчання, спрямованої на оптимізацію комунікації в освітньому середовищі.

4 ТЕСТУВАННЯ СИСТЕМИ ЗДІЙСНЕННЯ КОМУНІКАЦІЙ В ЕЛЕКТРОННОМУ ОСВІТНЬОМУ СЕРЕДОВИЩІ

Тестування систем — ключовий етап у процесі розробки програмного забезпечення, орієнтований на докладне вивчення програмного коду та виявлення можливих помилок у функціонуванні системи. Основною метою тестування є перевірка відповідності інформаційної системи в цілому або її окремих модулів очікуванням користувача. Цей етап визначає надійність та ефективність системи, спрямовуючи увагу на забезпечення високої якості функціонування та задоволення потреб користувачів.

4.1 Тестування процесу реєстрації та авторизації

Реєстраційний процес у системі включає в себе заповнення ряду обов'язкових полів, таких як ім'я, прізвище, електронна пошта, пароль, підтвердження паролю, а також вибір ролі із доступного переліку випадаючого списку. Ця процедура призначена для встановлення основних ідентифікаційних та авторизаційних параметрів користувача в системі.

Що зроблено, що очікується та результат:

Заповнюємо всі поля такими значеннями: ім’я «Alex», прізвище «Sai», електронна пошта «alex@gmail.com», пароль «12345678», роль «студент».

Вводимо у поле повторити пароль значення, яке не дорівнює значенню поля пароль. Очікуємо, що прийде відповідь від сервера, що поля пароля і повторення пароля не рівні. У результаті із сервера прийшло повідомлення «password !== confirmPassword» (рис. 4.1).

Заповнюємо всі поля коректними значеннями. Очікуємо успішну реєстрацію користувача, та перехід на домашню сторінку студента. У результаті новий користувач був створений у базі даних, пароль захешований (рис. 4.2).

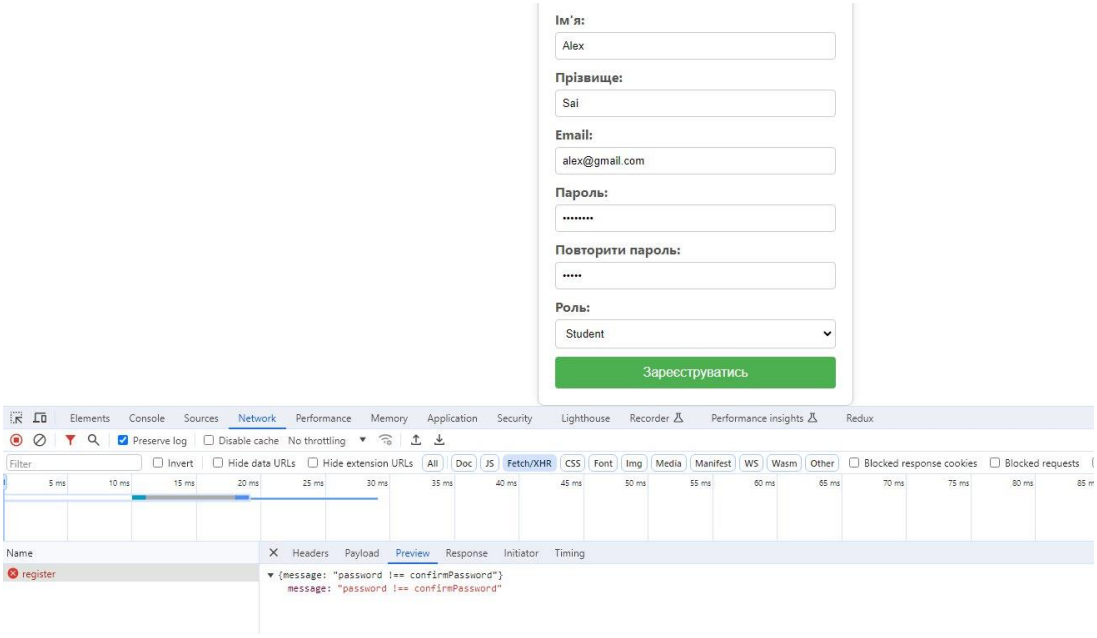

Рисунок 4.1 – Результат першого тесту процесу реєстрації

Рисунок 4.2 – Результат у базі даних після реєстрації нового користувача

Далі заходимо через адміністратора, та присвоюємо новоствореному студенту групу, із назвою «2МАТ-19м» (рис. 4.3). Очікуємо, що у даного студента з’являться на домашній сторінці всі курси назначеної групи, а в базі даних у студента з’явиться ідентифікатор групи. У результаті в базі даних у студента додалось поле «group» із ідентифікатором нової групи (рис. 4.4). На домашній сторінці студента з’являються всі курси назначеної групи (рис. 4.5).

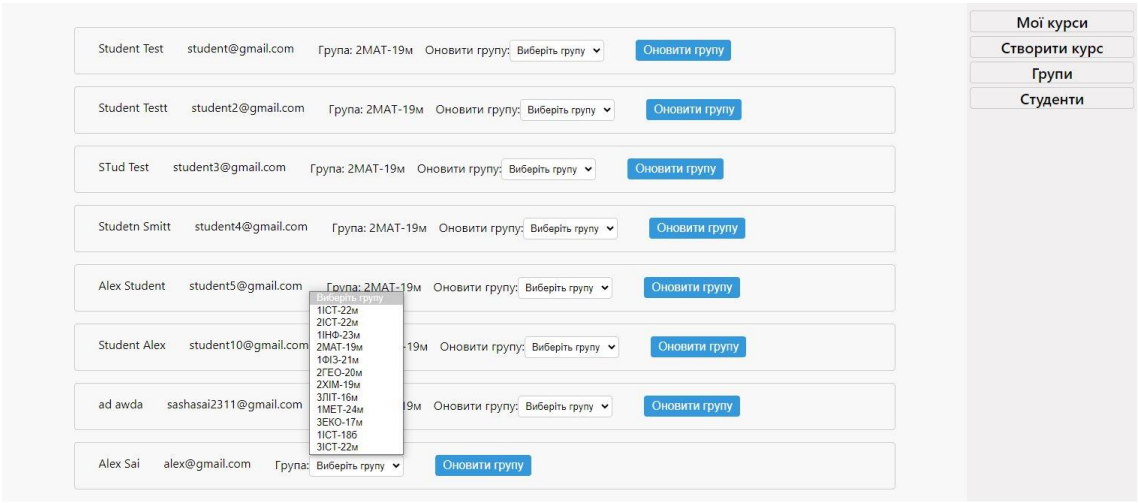

Рисунок 4.3 – Процес додавання групи студентові


```
_id: ObjectId('6576eb53a3c537b5cc0432c4')
firstName: "Alex"
lastName: "Sai"
email: "alex@gmail.com"
passwordHash: "$2b$10$42CZrec/J3UGZhCAAGMEXeHkUwVDk4SZSOKLoLUXfYyTg66D/GUXy"
role: "student"
avatarUrl: "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQRZhrZm_bEQIfEzt…"
createdAt: 2023-12-11T10:58:27.592+00:00
updatedAt: 2023-12-11T11:04:19.707+00:00
__v: 0
group: ObjectId('655b78970a5dbfa5c9ff0e46')
```


Рисунок 4.4 – Результат додавання нової групи у базі даних

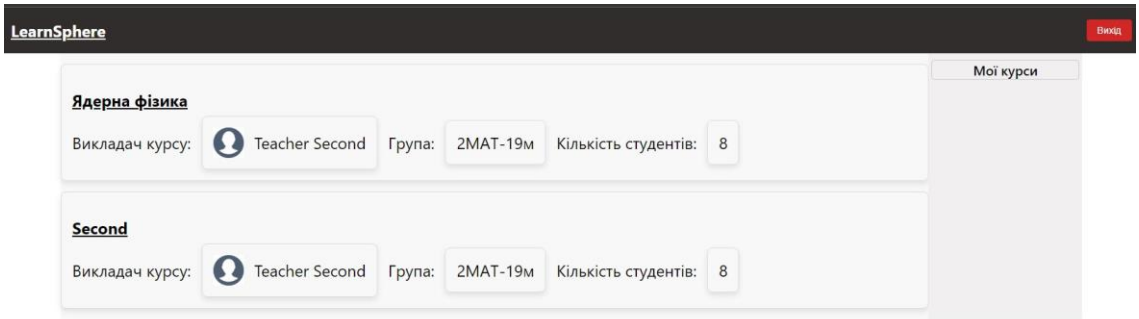

Рисунок 4.5 – Домашня сторінка користувача після присвоєння йому групи

Процес авторизації у системі здійснюється заповненням двох полів електронної пошти та пароля. Що зроблено, що очікується та результат.

Вводимо в поле електронної пошти значення пошти новоствореного користувача. Пароль вводимо, який не відповідає паролю користувача. Очікуємо, що сервер порівняє пароль із захешованим паролем в базі даних, та поверне повідомлення про те, що вони не рівні. У результаті сервер повернув відповідь про те, що або пошта або пароль не вірні (рис. 4.6).

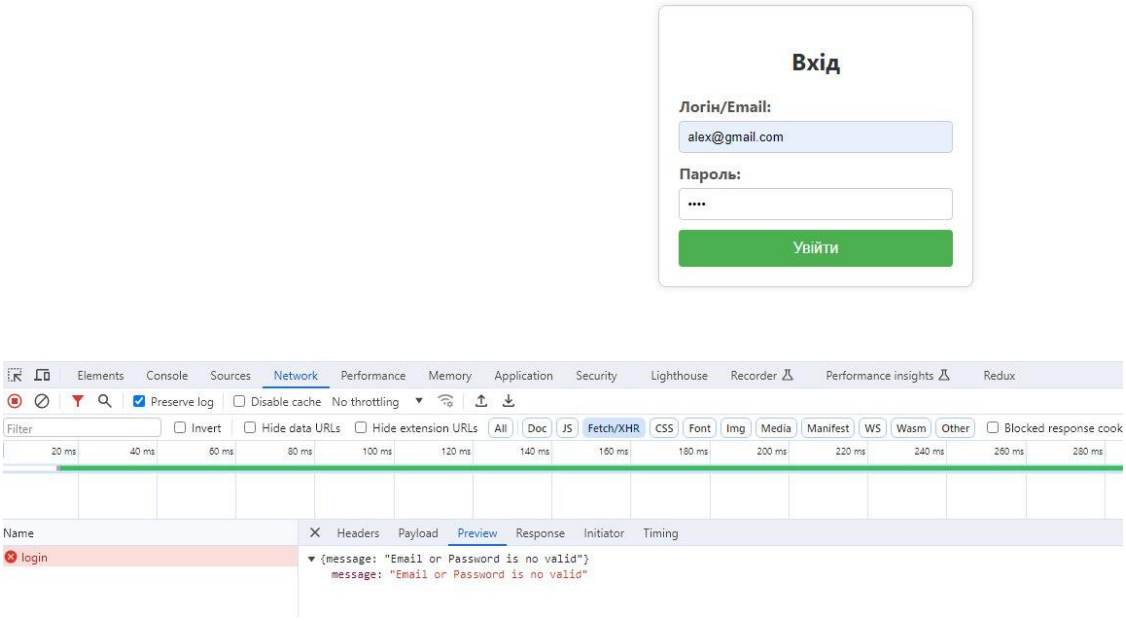

Рисунок 4.6 – Результат після не коректного паролю користувача

Після правильно заповнених полів електронної пошти та паролю, очікується, що користувач перейде на домашню сторінку. У результаті студент потрапляє на свою домашню сторінку (рис. 4.5).

4.2 Тестування всіх процесів на головній сторінці курсу

На домашній сторінці студент бачить всі свої курси, при кліку на назву курсу, очікується, що студент потрапляє на головну сторінку даного курсу. У

результаті після натискання на назву курсу, студент потрапив на головно сторінку курсу, на вкладку «Матеріали» (рис. 4.7).

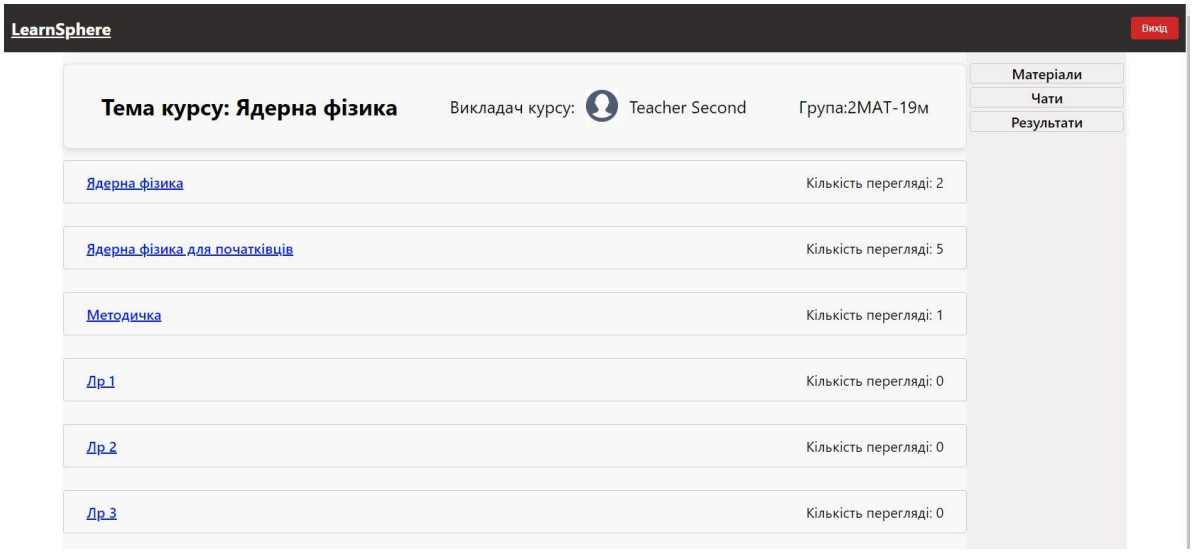

Рисунок 4.7 – Результат після натискання на назву курсу

Після переходу на вкладку «Чати», очікується що студент буде бачити загальний чат курсу, та індивідуальний чат із викладачем курсу. У результаті переходу на вкладку, студент бачить загальний чат курсу із всіма повідомленнями, та чат індивідуальний із викладачем (рис. 4.8).

Після написання повідомлення у загальний чат курсу очікується, що повідомлення моментально з’явиться із правої сторони головного блоку чату, із зображенням, іменем та прізвищем студента. У результаті повідомлення з’явилось як і очікувалось (рис. 4.9).

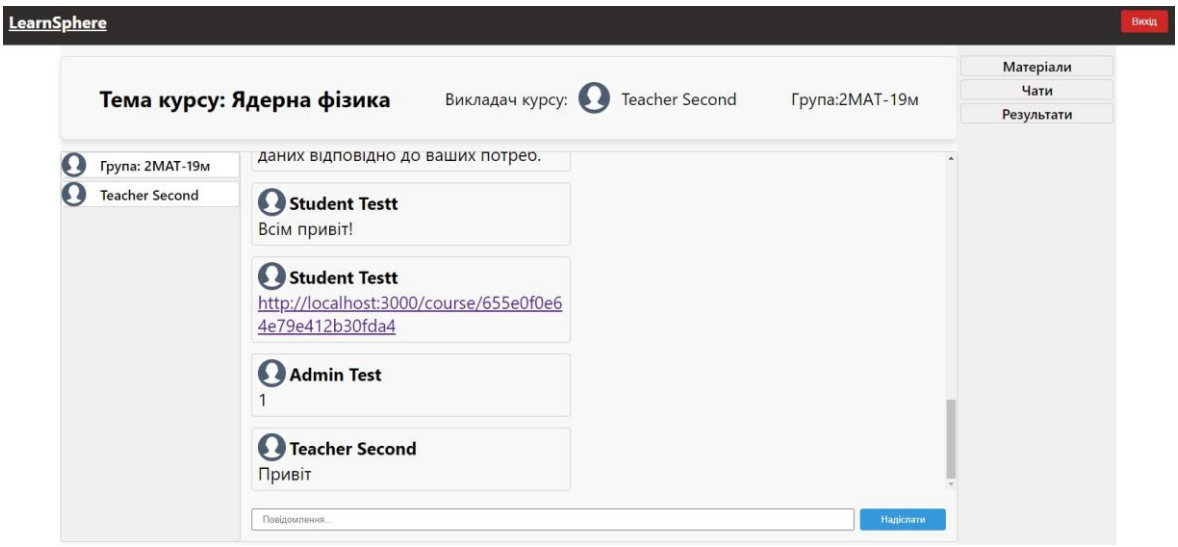

Рисунок 4.8 – Результат після натискання на вкладку «Чати»

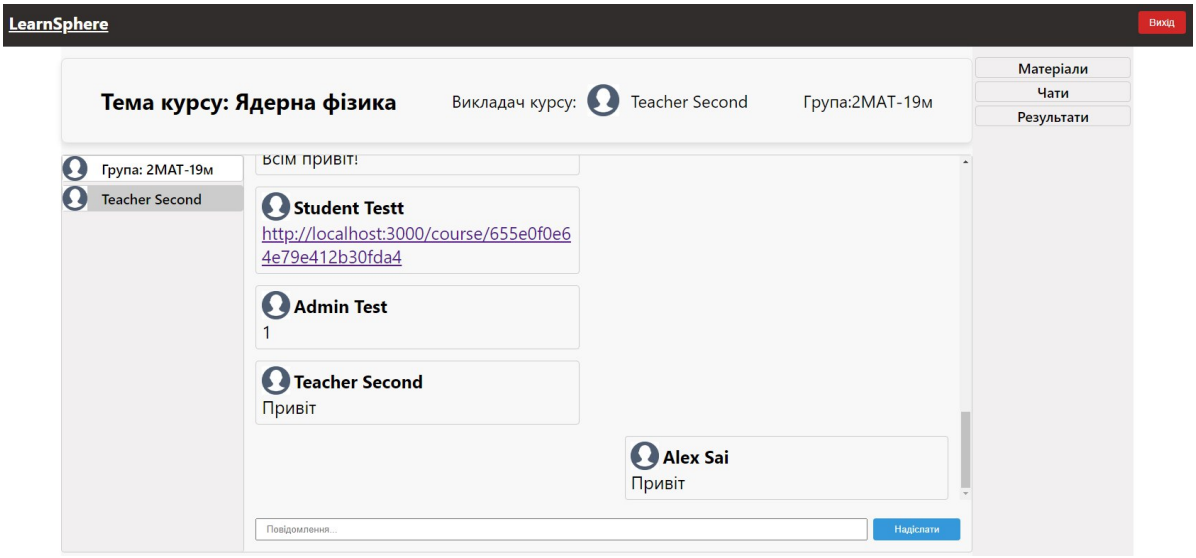

Рисунок 4.9 – Результат після відправлення повідомлення у чат

Після створення викладачем курсу всіх критеріїв оцінювання (рис. 4.10) очікується, що у результатах студента з’являться всі критерії із оцінками «0». У результаті у створеного студента з’являються результати курсу, із всіма створеними критеріями викладачем із оцінкою «0» (рис. 4.11).

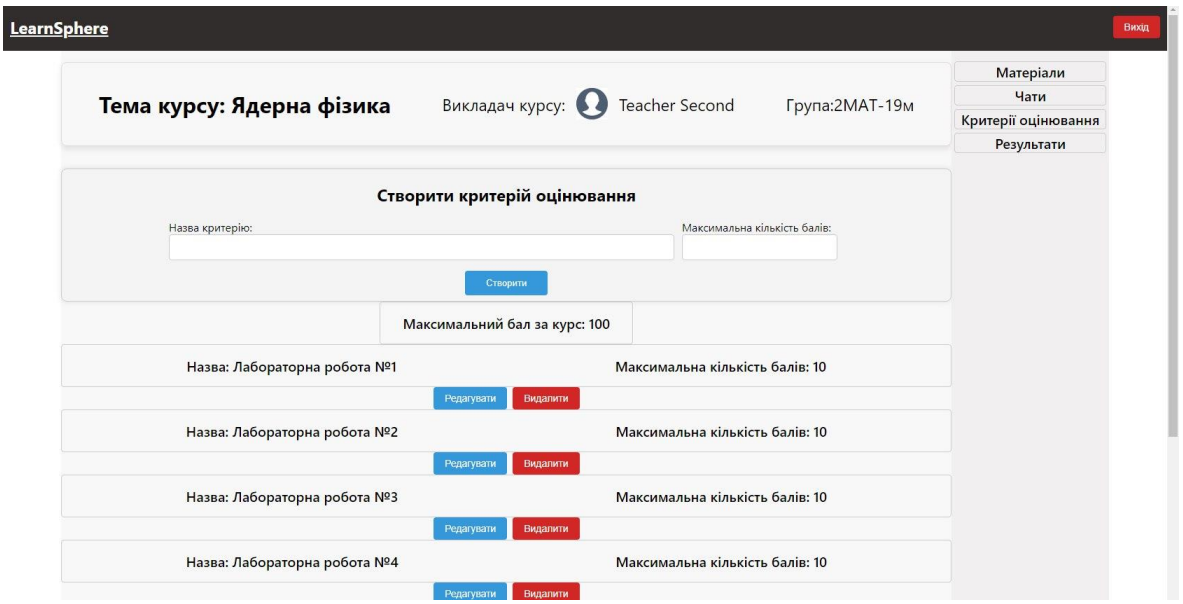

Рисунок 4.10 – Процес створення критеріїв оцінювання викладачем

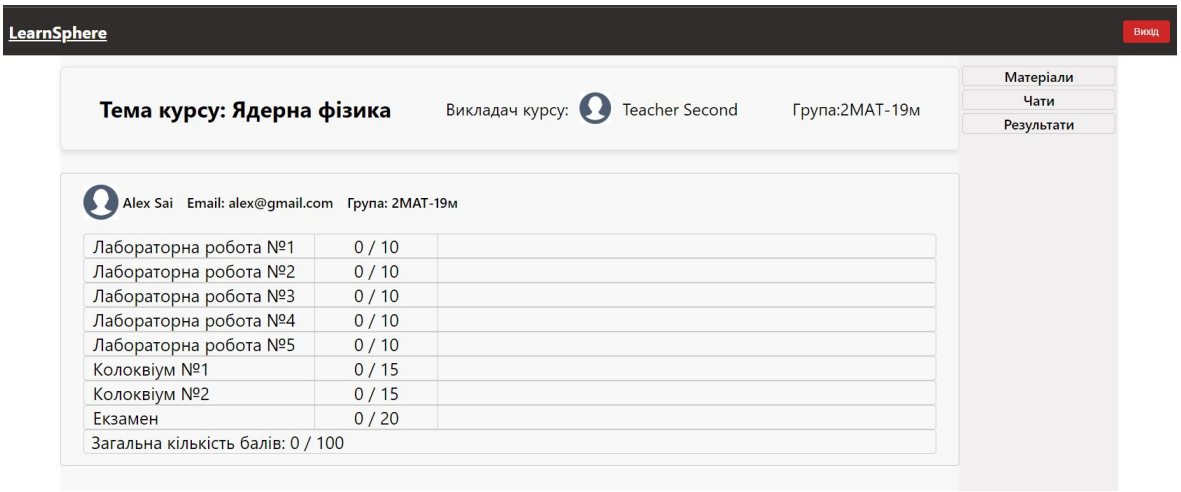

Рисунок 4.11 – Результати студента після творених критеріїв оцінювання

Заходимо в систему за викладача, та виставляємо оцінки даному студентові (рис. 4.12). Очікується, що після виставлених оцінок, студент одразу їх може переглянути у своїх результатах до курсу. У результаті всі оцінки з’явились як і очікувалось (рис. 4.13).

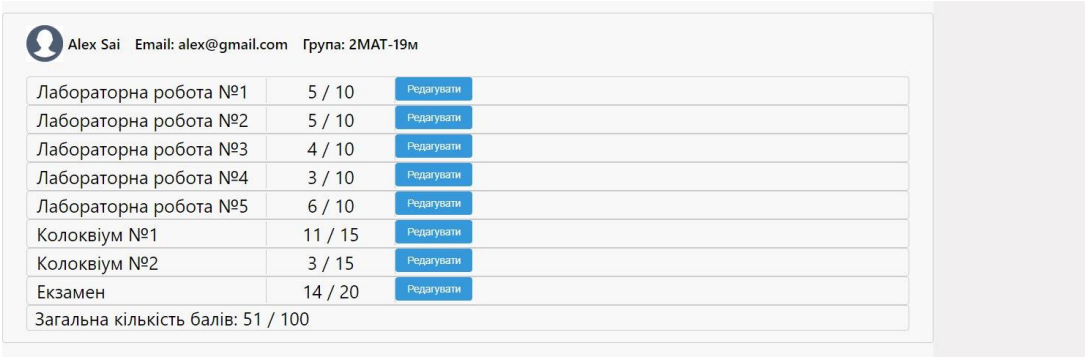

Рисунок 4.12 – Процес виставлення оцінок викладачем

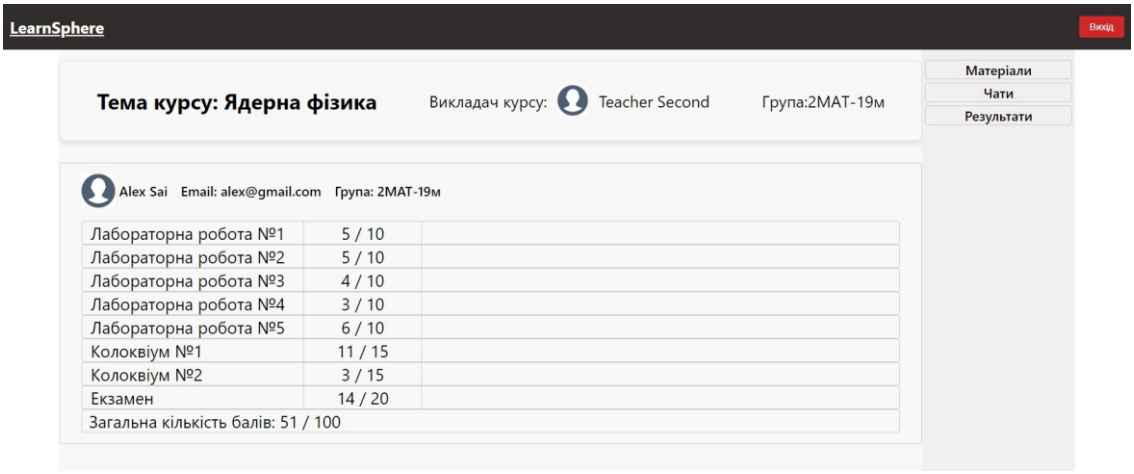

Рисунок 4.13 – Оцінки студента на вкладці «Результати»

4.3 Висновки до розділу

У даному розділі було проведено тестування ключових функціональних процесів системи, зосереджуючись на перевірці ефективності та надійності операцій реєстрації та авторизації користувачів. Також були піддані тестуванню операції, що відбуваються на головній сторінці курсу, такі як перегляд навчальних матеріалів, участь в чатах, обмін повідомленнями в чатах, створення критеріїв оцінювання, перегляд результатів, внесення оцінок викладачем та перегляд отриманих оцінок студентом.

Процедура тестування пройшла успішно, і усі отримані результати відповідають очікуванням.

5 ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Технологічний аудит розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі

Не підлягає сумніву, що комунікація в електронному освітньому середовищі сприяє активізації навчання студентів, покращенню їх взаємодії з викладачами, спілкуванню одне з одним, кращому застосуванню методичних матеріалів, поглибленню розуміння і запам'ятовуванню навчального матеріалу.

Саме електронні платформи надають можливість індивідуалізувати процес навчання, а ефективна комунікація дозволяє викладачам і студентам адаптувати підхід до конкретних потреб та можливостей кожного учасника навчального процесу, сприяє залученню студентів до дискусій, обговорень та колективних проєктів, створює зручний механізм для висловлення власних думок, задавання питань та розуміння матеріалу, відстеження успішності студентів тощо. В електронному освітньому середовищі студенти навчаються ефективно спілкуватися в онлайн, що сьогодні стає надзвичайно важливим в цифровому світі та на ринку праці.

Тому метою виконаної магістерської кваліфікаційної роботи було удосконалення методів та засобів реалізації комунікації в електронному освітньому середовищі для викладачів, студентів та інших учасників навчального процесу.

Для досягнення цієї мети було: розглянуто методи комунікації у електронному освітньому середовищі; здійснено порівняльний аналіз методів, та визначено переваги та недоліки; проаналізовано технології та підходи для програмної реалізації системи; програмно реалізовано платформу для комунікації в електронному освітньому середовищі.

В результаті було запропоновано нове програмне забезпечення, яке суттєво покращує процес комунікації викладачів, студентів та інших учасників навчального процесу в електронному освітньому середовищі.

Для встановлення комерційного потенціалу розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі було запрошено 3-х експертів: к.т.н., доцента Коваленко О.О., к.т.н., доцента Кулика Я.А. та учасника оргкомітету хакатону «Vinnytsia Linuxation» та журі конкурсу «Адміністрування Linux» (IT-Universe)» викладача кафедри КН Малініча І.П.

Встановлення комерційного потенціалу розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі було здійснено за критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання технічного рівня та комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промислово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використо

		му комплексі			вуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Експерти оцінили розроблене програмне забезпечення реалізації комунікацій в електронному освітньому середовищі так, як показано в табл. 5.2:

Таблиця 5.2 – Результати технологічного аудиту розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Коваленко О.О.	Кулик Я.А.	Малініч І.П.
	Бали, що їх виставили експерти:		
1	4	3	3
2	3	3	4
3	4	3	4

4	3	3	3
5	4	4	4
6	3	3	4
7	3	3	4
8	4	4	4
9	4	3	3
10	3	3	4
11	4	4	3
12	4	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 39$	$СБ_3 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{43+39+44}{3} = \frac{126}{3} = 42,00$		

Встановлення комерційного потенціалу розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі було зроблено на основі рекомендацій, наведених в таблиці 5.3 [30].

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 42,00 балів, то це свідчить, що розроблене програмне забезпечення реалізації комунікацій в електронному освітньому середовищі має рівень комерційного потенціалу, який вважається «високим». Це пояснюється тим, що розробка дозволяє студентам з різних частин світу отримувати якісну освіту, не виходячи з дому, суттєво зменшує географічні обмеження та забезпечує якісний доступ до ресурсів та експертів з будь-якої точки світу.

5.2 Розрахунок витрат на розроблення програмного забезпечення реалізації комунікацій в електронному освітньому середовищі

При розробленні програмного забезпечення реалізації комунікацій в електронному освітньому середовищі були зроблені певні витрати. Зокрема:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \qquad (5.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що М = (6700…25800) грн/місяць; $Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 24 дні; t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 5.4:

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	24400	1017,67	20 годин	3392,23 ≈ 3393
2. Магістрант-студент-виконавець	2000 (беремо 6700)	279,17	70	19541,9 ≈ 19542
3. Консультант з економічної частини	18300	762,50	1,5 години	190,62 ≈ 191 (при 6-годинному робочому дні)
4.Консультант навчального відділу	15000	625	2	1250
Загалом				$З_о$ = 24 376 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \qquad (5.2)$$

Приймемо, що α = 0,112. Тоді для даного випадку отримаємо:

$$З_д = 0,112 \times 24376 = 2730,11 \approx 2731 \text{ грн.}$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (24376 + 2731) \times 0,22 = 5963,54 \approx 5964 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн; $Н_а$ – річна норма амортизаційних відрахувань. Для даного випадку можна прийняти, що $Н_а$ = (2,5...25)%; Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	55500	25	3,12 (при 55% використанні	1984,12 ≈ 1985
2. Приміщення університету, кафедри	13300	2,5	3,12 при 55% використанні	47,54≈ 48
Всього				А = 2033 грн

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.}, \quad (5.5)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг; $Ц_i$ – вартість матеріалу *i*-го найменування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *i*-го найменування; $Ц_в$ – ціна відходів матеріалу *i*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн}, \quad (5.6)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.; $Ц_i$ – ціна комплектуючих *i*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих. Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 1000 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт; П – установлена потужність обладнання, кВт; П = 0,8 кВт; Ф – фактична кількість годин роботи обладнання, годин. Приймемо, що Ф = 277 годин; $К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,78. $К_д$ – коефіцієнт корисної дії, $К_д$ = 0,65.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 0,8 \cdot 277 \cdot 0,78}{0,65} = 1196,64 \approx 1197 \text{ грн.}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (5.8)$$

Для даного випадку отримаємо:

$$В_{інш} = 0,5 \times 24376 = 12188 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

$$В = 24376 + 2731 + 5964 + 2033 + 1000 + 1197 + 12188 = 49489 \text{ грн.}$$

Л). Загальні витрати на розроблення програмного забезпечення реалізації комунікацій в електронному освітньому середовищі $В_{заг}$ становлять:

$$В_{заг} = \frac{В}{\beta}, \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,9$ [30], оскільки робота практично завершена.

Тоді: $В_{заг} = \frac{49489}{0,9} = 54987,77$ грн або приблизно 55 тисяч грн.

Тобто прогнозовані загальні витрати на розроблення програмного забезпечення реалізації комунікацій в електронному освітньому середовищі становлять приблизно 55 тисяч грн.

5.3 Розрахунок економічного ефекту від можливої комерціалізації розробки

Економічний ефект від можливої комерціалізації розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі виявляється у його значно кращих функціональних можливостях та у суттєвому зростанні попиту на подібні розробки. Попит на розробку складають освітні заклади, які планують об’єднати всі навчальні процеси, навчальні матеріали, оцінювання, онлайн спілкування в одну систему, а також численні фізичні особи і репетитори, які зможуть вести свою освітню діяльність в межах однієї системи, ділитись навчальними матеріалами, оцінювати знання студентів, збирати групи, спілкуватись онлайн тощо.

Аналіз ринку також показав, що потенційна кількість користувачів аналогічних систем становить приблизно 10 шт. і буде постійно зростати. Тобто, якщо дана розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися протягом 2024-го, 2025-го та 2026-го років.

Прогноз зростання попиту на дану розробку складає по роках:

a) 2024 р. – + 5 шт. до базового року;
b) 2025 р. – + 10 шт. до базового року;
c) 2026 р. – + 15 шт. до базового року.

У 2022 році подібна за функціями розробка коштувала на ринку приблизно 50 тисяч грн. Тоді дану, значно кращу за своїми функціями розробку, можна буде реалізовувати на ринку дорожче, в середньому за 60 тисяч грн або на 10 тисяч грн дорожче.

За твердженням експертів, загальна сума прибутку для власника подібної розробки становитиме 72 тисяч грн в рік (при 5-ти користувачах). При більшій кількості користувачів величина прибутку буде відповідно вищою.

Проведемо більш точні розрахунки. Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від виведення даної розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів даної розробки у цьому році. Для даного випадку це є збільшення ціни реалізації розробки $\Delta Ц_o$ = 60 – 50 = 10 тисяч грн; N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 10 шт.; ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме по роках, відповідно: у 2024 році – + 5 шт., у 2025 році + 10 шт., та у 2026 році + 15 шт.; $Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_o$= 60 тисяч грн; n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для даного випадку $n = 3$; λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$; ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5; υ – ставка податку на прибуток. У 2023-26 роках υ = 18% (припущення).

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження і комерціалізації даної розробки (2024 р.) становитиме:

$$\Delta П_1 = [10 \cdot 10 + 60 \cdot 5] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 137 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження і комерціалізації даної розробки протягом другого (2025) року складе:

$$\Delta П_2 = [10 \cdot 10 + 60 \cdot 10] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 240 \text{ тис. грн.}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження та комерціалізації даної розробки протягом третього (2026) року складе:

$$\Delta П_3 = [10 \cdot 10 + 60 \cdot 15] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 342 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації даної розробки для інвестора становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн; т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для даного випадку т = 3 роки; τ – ставка дисконтування. Приймемо τ = 0,10 (10%); t – період часу від моменту початку розроблення програмного забезпечення реалізації комунікацій в електронному освітньому середовищі до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від впровадження та комерціалізації даної розробки, складе:

$$ПП = \frac{137}{(1+0,1)^2} + \frac{240}{(1+0,1)^3} + \frac{342}{(1+0,1)^4} \approx 113 + 181 + 234 = 528 \text{ тисяч грн,}$$

Теперішня вартість інвестицій PV, що повинні бути вкладені у реалізацію розробки: PV = (1,0…5,0) × $В_{заг}$.

Для даного випадку PV = (1,0…5,0) × 55 = 2,0 × 55 = 110 тисяч грн.

Абсолютний ефект від можливих вкладених в реалізацію даної розробки інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \qquad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для потенційного інвестора від можливої комерціалізації даної розробки, грн; PV – теперішня вартість інвестицій PV = 110 тисяч грн. Абсолютний ефект від можливого впровадження розробки складе:

$$Е_{абс} = 528 - 110 = 418 \text{ тисяч грн.}$$

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \qquad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 418 тис. грн; PV – теперішня вартість початкових інвестицій PV = 110 тис. грн; $Т_ж$ – життєвий цикл розробки, роки. $Т_ж$ = 4 років (2023-й, 2024-й, 2025-й, 2026-й роки)

Для даного випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{418}{110}} - 1 = \sqrt[4]{1 + 3,800} - 1 = \sqrt[4]{4,800} - 1 = 1,48 - 1 = 0,48 = 48,0\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією даної розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \qquad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022-2023 роках в Україні d = (0,10...0,12); f – показник, що характеризує ризикованість вкладень; f = (0,1...0,50). Приймемо f = 0,30.

Для даного випадку отримаємо:

$$\tau_{мін} = 0{,}12 + 0{,}30 = 0{,}42 \text{ або } \tau_{мін} = 42\%.$$

Оскільки величина $E_в = 48{,}0\% > \tau_{мін} = 42\%$, то потенційний інвестор може бути зацікавлений у фінансуванні та комерціалізації розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (5.15)$$

Для даного випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,48} = 2,08 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні.

Якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{137}{(1+0,2)^2} + \frac{240}{(1+0,2)^3} + \frac{342}{(1+0,2)^4} \approx 95 + 139 + 165 = 399 \text{ тисяч грн},$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$E_{абс} = 399 - 110 = 289 \text{ тисяч грн}.$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 289 тисяч грн; PV – теперішня вартість початкових інвестицій PV = 110 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{289}{110}} - 1 = \sqrt[4]{1+2,6272} - 1 = \sqrt[4]{3,6272} - 1 = 1,38 - 1 = 0,38 = 38,0\%.$$

Оскільки величина $E_в$ = 38,0% $\approx \tau_{мін} = 42\%$, то потенційний інвестор (після проведення додаткових уточнень) може бути зацікавлений у фінансуванні та комерціалізації розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі.

Якщо рівень інфляції в країні зросте до 30%, то:

$$ПП = \frac{137}{(1+0,3)^2} + \frac{240}{(1+0,3)^3} + \frac{342}{(1+0,3)^4} \approx 81 + 109 + 120 = 310 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$E_{абс} = 310 - 110 = 200 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 200 тисяч грн; PV – теперішня вартість початкових інвестицій PV = 110 тисяч грн.

Для даного випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{200}{110}} - 1 = \sqrt[4]{1+1,8182} - 1 = \sqrt[4]{2,8182} - 1 = 1,295 - 1 = 0,295 = 29,5\%.$$

Оскільки величина $E_в$ = 29,5% $< \tau_{мін} = 42\%$, то потенційний інвестор може бути і НЕ зацікавлений у фінансуванні та комерціалізації даної розробки.

Зроблені розрахунки у вигляді графіків наведено на рис. 5.1.

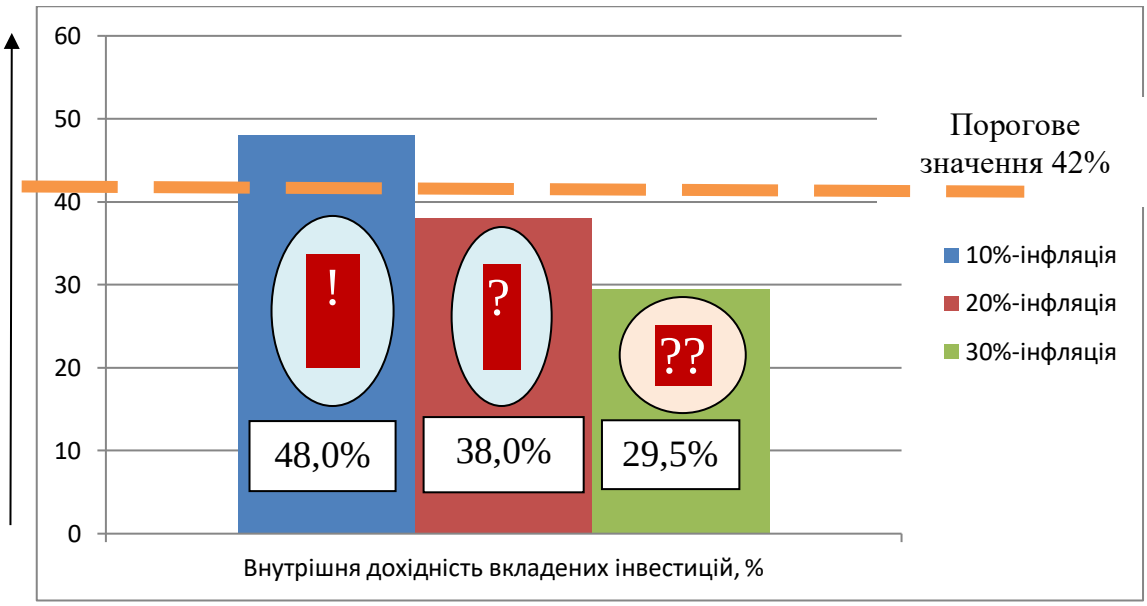

Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, вкладених у комерціалізацію розробки, від рівня інфляції в країні (10%, 20% і 30%)

Аналіз діаграм на рис 5.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності інвестицій становить $E_в$ = 48,0%, що більше порогового значення $\tau_{мін}$ = 42% і тому комерціалізація даної розробки може бути доцільною. При рівні інфляції в 20% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію розробки, становить $E_в$ = 38,0% < 42%, а при рівні інфляції в 30% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію розробки, становить $E_в$ = 29,5% < 42%.

Тому у двох останніх випадках остаточне рішення з питання комерціалізації розробки потребує проведення додаткових розрахунків (можливо – зниження рівня ризикованості вкладень, збільшення попиту на розробку, збільшення ціни реалізації розробки тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю 5.6.

Таблиця 5.6 – Результати виконаної економічної частини магістерської кваліфікаційної роботи

Показники	Задані у ТЗ	Досягнуті у магістерській	Висновок

		кваліфікаційній роботі	
1. Витрати на розробку	Не більше 60 тис. грн	55 тис. грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 400 тисяч грн	418 тисяч грн (при 10%-інфляції)	Виконано
3. Внутрішня дохідність інвестицій, %	не менше 42%	48,0% (при 10%-інфляції)	Досягнуто
4. Термін окупності інвестицій, роки	до 3-х років	2,08 років	Виконано

Таким чином, основні техніко-економічні показники (характеристики) розробленого програмного забезпечення реалізації комунікацій в електронному освітньому середовищі, визначені у технічному завданні, виконані.

ВИСНОВКИ

Проведено аналіз актуальності та значущості ефективних інструментів комунікації учасників в електронному освітньому середовищі. Розглянуті методи, такі як електронна пошта, онлайн-чати, відеоконференції та онлайн системи оцінювання. Відзначено, що необхідно розглядати їх як складові частини системи комунікації, а не повноцінні платформи для онлайн навчання.

Виконаний аналіз результатів дослідження, проведеного компанією Pearson у 2022 році, який виявив, що онлайн-форуми (90%), чати (85%), електронна пошта (80%) та відеоконференції (75%) є найпопулярнішими методами комунікації в електронному освітньому середовищі.

Додатково, в розділі проведено аналіз повноцінних навчальних платформ, таких як Google Classroom, Classtime та N-Code. Висвітлено інструменти розробки в контексті платформи N-Code, але виявлено основний недолік - неефективну систему оцінювання студентів. Ці висновки вказують на важливість подальшого вдосконалення систем комунікації та оцінювання в електронному навчальному середовищі.

У другому розділі виконано аналіз різних типів архітектур систем, таких як монолітна, клієнт-серверна, розподілена та мікросервісна архітектура. Проведено порівняльний аналіз переваг та недоліків кожного типу з метою визначення оптимального варіанту для розробки системи.

Обрано клієнт-серверну архітектуру з урахуванням розмірів додатку та обґрунтовано відмову від монолітної, розподіленої та мікросервісної архітектур через їхні недоліки у контексті даного проєкту.

Вибір клієнт-серверної архітектури спрямований на оптимізацію ефективності та надійності для успішної реалізації завдань, поставлених у дипломному проєкті.

Також розглянуті технології для розробки серверної частини системи, серед яких вибір був зроблений на користь Node.js та фреймворка Express. Зазначено переваги Node.js, такі як асинхронний підхід, що дозволяє ефективно

обробляти багато запитів паралельно, і його використання в області обміну повідомленнями в чаті.

У порівнянні з альтернативами, такими як Java та PHP, Node.js виявився більш швидким, масштабованим та ефективним для обробки багатьох одночасних підключень.

Детально проаналізовано вибір NoSQL бази даних MongoDB з урахуванням гнучкості у роботі з даними та необхідності ефективного зберігання різноструктурованих даних у системі обміну повідомленнями. Зазначено, що MongoDB є оптимальним вибором для даного проєкту, особливо враховуючи її популярність у зберіганні даних чату.

У цьому розділі також розглянуті та проаналізовані технології для розробки клієнтської частини системи. Визначено переваги JavaScript-бібліотеки React, такі як використання віртуального DOM та можливість легкої інтеграції з роутерами.

В порівнянні з альтернативами, такими як Vue та Angular, React виявився ефективнішим та зручнішим для розробки, особливо в контексті створення односторінкових додатків та обміну повідомленнями у чаті.

У розділі обґрунтовуються та систематизуються важливі технічні рішення, зроблені у процесі розробки системи.

У третьому розділі був проведений детальний аналіз та опис основних етапів розробки клієнт-серверної системи, спрямованої на забезпечення ефективної комунікації в електронному освітньому середовищі. Під час роботи було розглянуто процеси та ролі учасників системи, що були визначені як студент, викладач та адміністратор. Взаємодія та процеси були наглядно проілюстровані на діаграмах прецедентів, а алгоритми всіх процесів були детально описані та продемонстровані на схемах.

Особлива увага приділялась архітектурі системи, що була реалізована з використанням архітектурного стилю REST для ефективної взаємодії між клієнтською та серверною частинами системи. Використано бібліотеку axios для відправки запитів із клієнтської частини, фреймворк Express для обробки запитів

на сервері та модуль mongoose для взаємодії серверної частини з базою даних MongoDB.

Детально розглянуто структуру бази даних системи, включаючи колекції "users", "messages", "materials", "marks", "groups", "criterias" та "courses", а також зв'язки між ними. Реалізація серверної частини системи використовувала Node.js та фреймворк Express, створюючи всі необхідні компоненти для ефективної взаємодії між клієнтом та сервером. Структура серверної частини була представлена компонентами, такими як головний індекс-файл, контролери, моделі, утиліти та JSON-файл, що містить налаштування сервера.

У контексті реалізації клієнтської частини використовувалась JavaScript-бібліотека React. Структура клієнтської частини системи включала ключові елементи, такі як файл запитів до серверної частини (fetch.js), файл axios, файл App, сторінки та компоненти. Три ключові ролі системи (студент, викладач, адміністратор) були описані разом з інтерфейсами та функціоналом кожної ролі.

У четвертому розділі було проведено тестування ключових функціональних процесів системи, зосереджуючись на перевірці ефективності та надійності операцій реєстрації та авторизації користувачів. Також були піддані тестуванню операції, що відбуваються на головній сторінці курсу, такі як перегляд навчальних матеріалів, участь в чатах, обмін повідомленнями в чатах, створення критеріїв оцінювання, перегляд результатів, внесення оцінок викладачем та перегляд отриманих оцінок студентом. Процедура тестування пройшла успішно, і усі отримані результати відповідають очікуванням.

В економічному розділі була розглянута доцільність комерціалізації розробки системи для реалізації комунікацій в електронному освітньому середовищі. Основні техніко-економічні показники (характеристики) розробленого програмного забезпечення визначені у технічному завданні, виконані.

Результати роботи були використані в експериментальній частині електронної системи управління освітнім процесом ВНТУ JetIQ. Акт впровадження № 3/ 04.12.2023, продемонстровано в додатку Д.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Маркова Т. В. Інформаційно-комунікативне середовище як засіб підвищення майстерності педагога. Андрагогічний вісник: Наукове електронне періодичне видання. Випуск 5. 2014. С. 191-197.
2. Биков В. Ю., Спірін О. М., Пінчук О. П. Проблеми та завдання сучасного етапу інформатизації освіти. Інститут інформаційних технологій і засобів навчання НАПН України. Видавничий дім «Сам», 2017. С. 191-198.
3. Генсерук Г., Бойко М., Мартинюк С. Цифрові інструменти комунікації в освітньому процесі закладу вищої освіти. Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка, 2022. Серія: педагогіка, 1(1). С. 31–39.
4. Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації. 2023. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024 (дата звернення: 20.11.2023).
5. Михальчук С. О. Інтернет-комунікації як чинник демократизації електоральної участі громадян України: дис. канд. політ. наук: 23.00.02. Луцьк, 2017. 18 с.
6. Petar Jandric. Learning in the Age of Digital Reason. Springer, 2017. 195 с.
7. Owusu-Boampong, Angela, Holmberg, Carl. Distance education in European higher education: the potential extended. Джерело ЮНЕСКО. Німеччина, 2015.
8. Terry Anderson, Jon Dron. Three Generations of Distance Education Pedagogy.Онлайн публікація, 2020. С. 81-97.
9. Олексенко В. М. Комунікація при підготовці фахівців за дистанційною формою навчання. Вісник ВПІ, 2005, № 1. С. 94-99.
10. Компанець Н. М. Використання інформаційно-комунікаційних технологій для активізації пізнавальної діяльності студентів при

вивченні іноземних мов. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2019, № 66, Т. 1. С. 61-64.

11. Гаріфуліна В. В., Чайка І. П. Засоби комунікації для вдосконалення дистанційного навчання. Вищий навчальний заклад Укоопспілки «Полтавський університет економіки і торгівлі», 2022. 44 с.
12. Pearson. Communication: Skill Development Framework. Version 1.0 June 2022.
13. Пехник А. В. Платформи для організації дистанційного навчання. Національний університет «Одеська юридична академія», 2020. С. 220-223.
14. Ткаченко, Л. В.; Хмельницька, О. С. Особливості впровадження дистанційного навчання в освітній процес закладу вищої освіти. Університет Григорія Сковороди в Переяслав, 2021, № 75, Т. 3. С. 91-96.
15. Classtime – навчальна онлайн платформа. URL: https://www.classtime.com/uk.html (дата звернення 20.10.2023).
16. N-Code – навчальна онлайн платформа. URL: https://n-code.study/#/ (дата звернення 20.10.2023).
17. Ivan Zmerzlyi. Архітектура веб додатків. URL: https://medium.com/@IvanZmerzlyi (дата звернення 02.11.2023).
18. Клієнт-серверна архітектура. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 02.11.2023).
19. Мікросервісна архітектура для початківців. Частина II. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-two/ (дата звернення 02.11.2023).
20. PHP – скриптова мова програмування. URL: https://hyperhost.ua/uk/wiki/chto-takoe-php (дата звернення 05.11.2023).
21. Node.js – відкрита платформа для розробки веб-серверів. URL: https://nodejs.org/en/about (дата звернення 05.11.2023).

22. Java – What is Java. URL: https://www.ibm.com/topics/java (дата звернення 05.11.2023).
23. Ruby – Навіщо потрібна мова програмування Ruby. URL: https://lemon.school/blog/navishho-potribna-mova-programuvannya-ruby (дата звернення 05.11.2023).
24. Yishan Li, Sathiamoorthy Manoharan. A performance comparison of SQL and NoSQL. Department of Computer Science University of Auckland New Zealand, 2013. С 15-19.
25. PostgreSQL. URL: https://www.postgresql.org/about/ (дата звернення 07.11.2023).
26. MongoDB. URL: https://brander.ua/technologies/mongo-db (дата звернення 07.11.2023).
27. React. URL: https://uk.legacy.reactjs.org/ (дата звернення 09.11.2023).
28. Vue. URL: https://ua.vuejs.org/guide/introduction.html (дата звернення 09.11.2023).
29. Angular. URL: https://apeirondb.com/ua/blog/it-technologies-angular (дата звернення 09.11.2023).
30. Козловський В.О., Лесько О.Й., Кавецький В.В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«__» _____2023 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
УДОСКОНАЛЕННЯ МЕТОДІВ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ КОМУНІКАЦІЙ В ЕЛЕКТРОННОМУ ОСВІТНЬОМУ СЕРЕДОВИЩІ
08-31.МКР.010.02.000 ТЗ

Керівник: к.т.н., проф. каф. АІІТ

_____ Євген ПАЛАМАРЧУК

«__» _____ 2023 р.

Розробив студент гр. 1ІСТ-22м

_____Олександр САЙ

«__» _____ 2023 р.

Вінниця 2023

1. Назва та галузь застосування.

Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі. Галузь застосування: Інформаційні системи та технології.

2. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____ 2023р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри АІІТ від «__» _____ 2023р.

3. Мета та призначення роботи.

Метою дослідження є удосконалення методів та засобів реалізації комунікації в електронному освітньому середовищі для викладачів, студентів та інших учасників навчального процесу.

4. Джерела розробки:

4.1 Node.js – відкрита платформа для розробки веб-серверів. URL: https://nodejs.org/en/about (дата звернення 05.11.2023).

4.2 React. URL: https://uk.legacy.reactjs.org/ (дата звернення 09.11.2023).

4.3 MongoDB. URL: https://brander.ua/technologies/mongo-db (дата звернення 07.11.2023).

5. Показники призначення

5.1 Мінімальні вимоги до техніки:

— Процесор: Intel Core i5 3 GHz або вище;

— Об’єм оперативної пам’яті: 8 Gb або більше;

— Жорсткий диск: 120 Gb SATA2 або вище.

5.2 Технології для серверної частини системи – Node.js, Express.js.

5.3 Технології для клієнтської частини системи – React.js.

5.4 База даних – MongoDB.

5.5 Середовище розробки та запуску – Visual Studio Code.

6. Економічні показники

До економічних показників входять:

— Витрати на розробку – не більше 60 тис. грн;

— Абсолютний ефект від впровадження розробки – не менше 400 тисяч грн;

— Внутрішня дохідність інвестицій – не менше 42%;

— Термін окупності інвестицій – до 3-х років.

7. Стадії розробки:

7.1 Аналіз існуючих методів комунікації та систем, де відбувається комунікація в електронному освітньому середовищі 01.09 – 10.09

7.2 Аналіз технологій для розробки системи із впровадженням методів та засобів комунікації в електронному освітньому середовищі 11.09 – 23.09

7.3 Програмна реалізація системи 24.09 – 30.10

7.4 Тестування системи 31.10 – 02.11

7.5 Підготовка економічного розділу 03.11 – 09.11

7.6 Оформлення пояснювальної записки і графічного матеріалу 10.11 – 20.11

8. Порядок контролю та приймання

8.1 Рубіжний контроль провести до 20.11.2023.

8.2 Попередній захист МКР провести до 21.11.2023.

8.3 Захист МКР провести до 19.12.2023.

Розробив студент групи 1ІСТ-22м _____ Олександр САЙ

Додаток Б (обов'язковий)

Ілюстративна частина

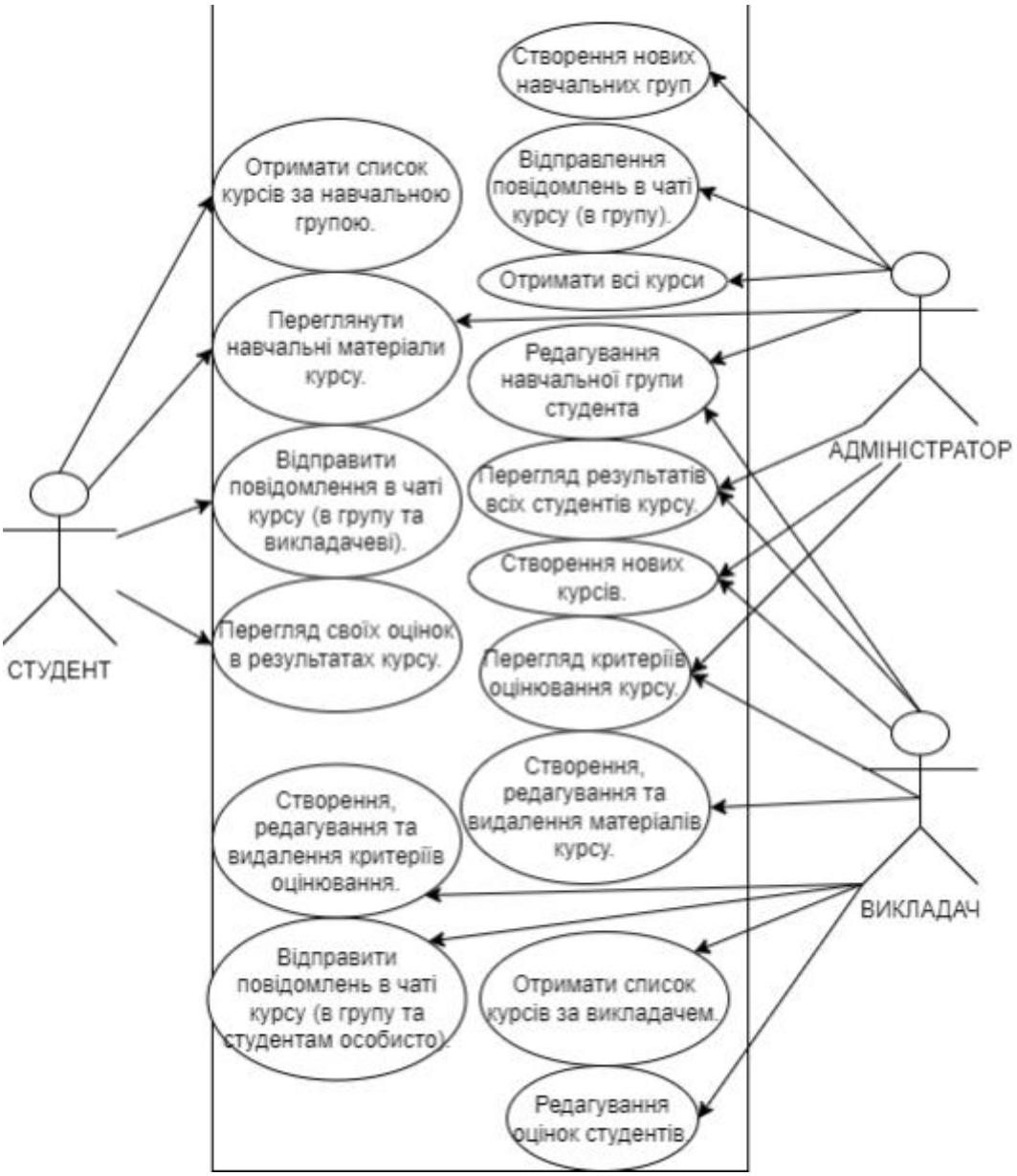

Рисунок Б.1 – Діаграма прецедентів системи

Продовження додатку Б

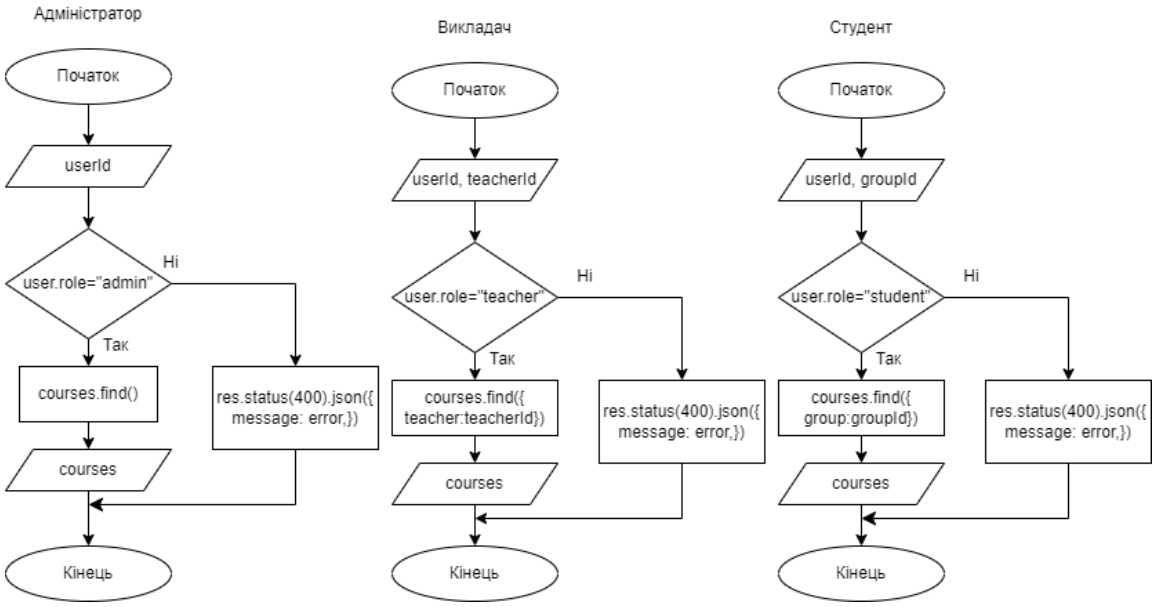

Рисунок Б.2 – Схема алгоритмів отримання курсів для всіх трьох ролей

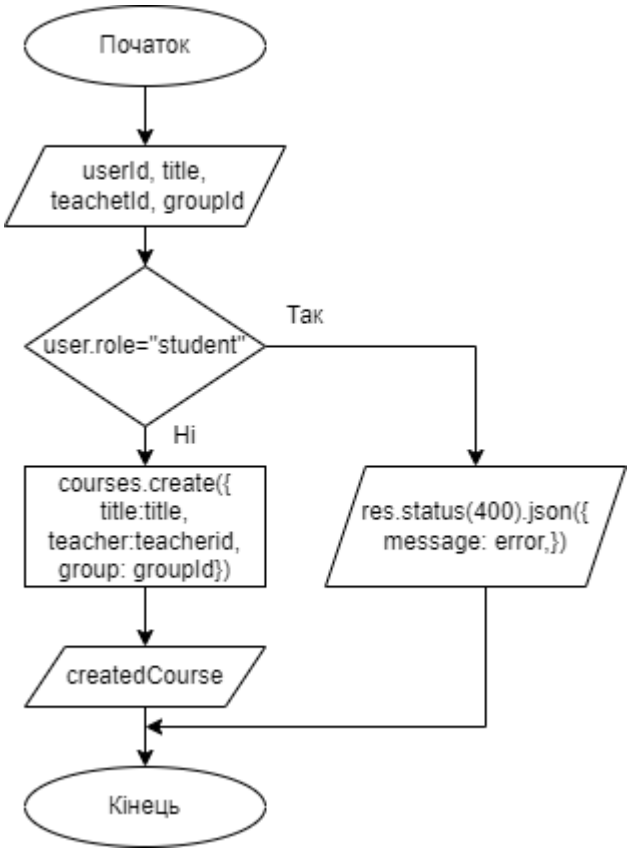

Рисунок Б.3 – Схема алгоритму процесу створення курсу

Продовження додатку Б

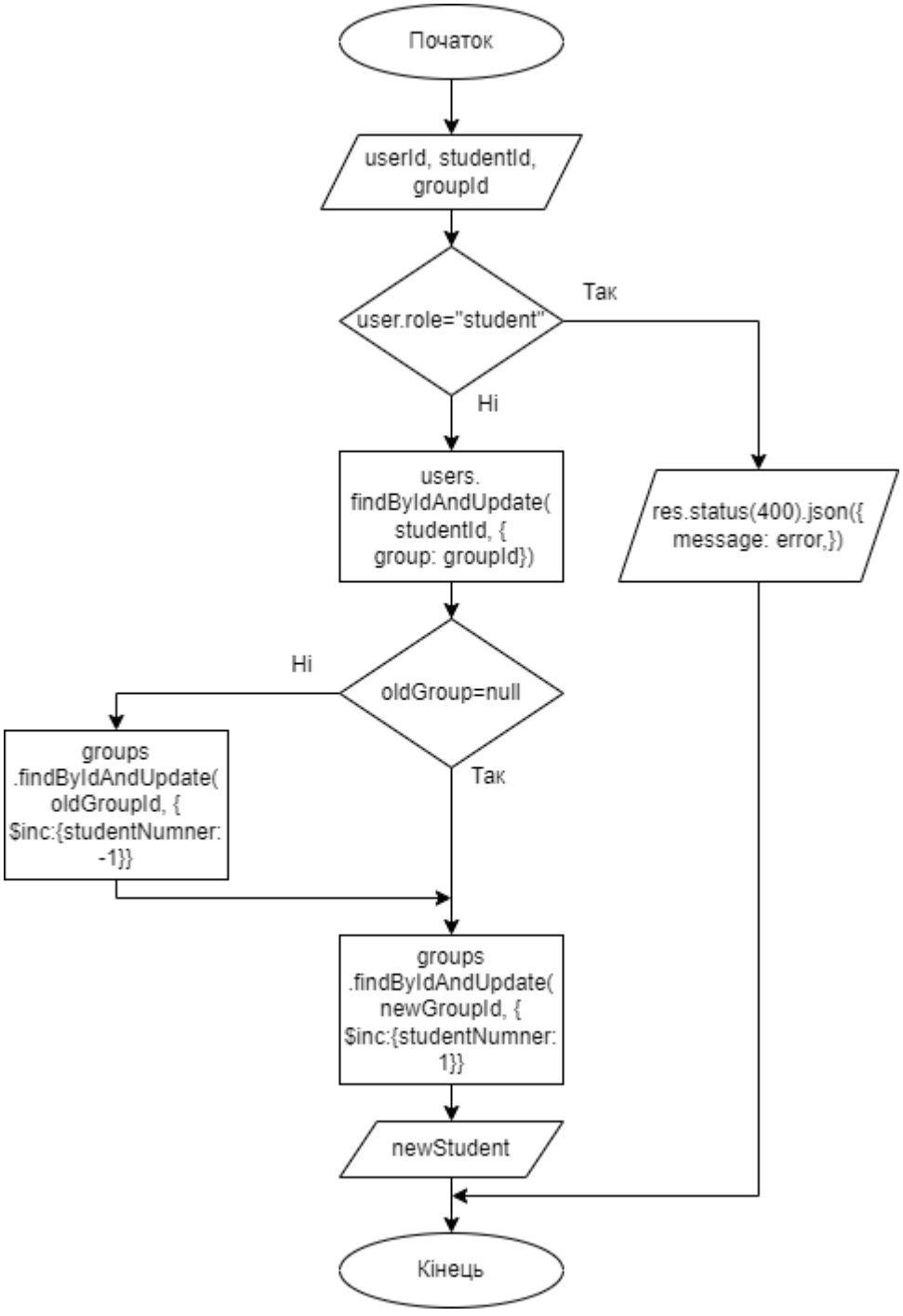

Рисунок Б.4 – Схема алгоритму оновлення навчальної групи студента

Продовження додатку Б

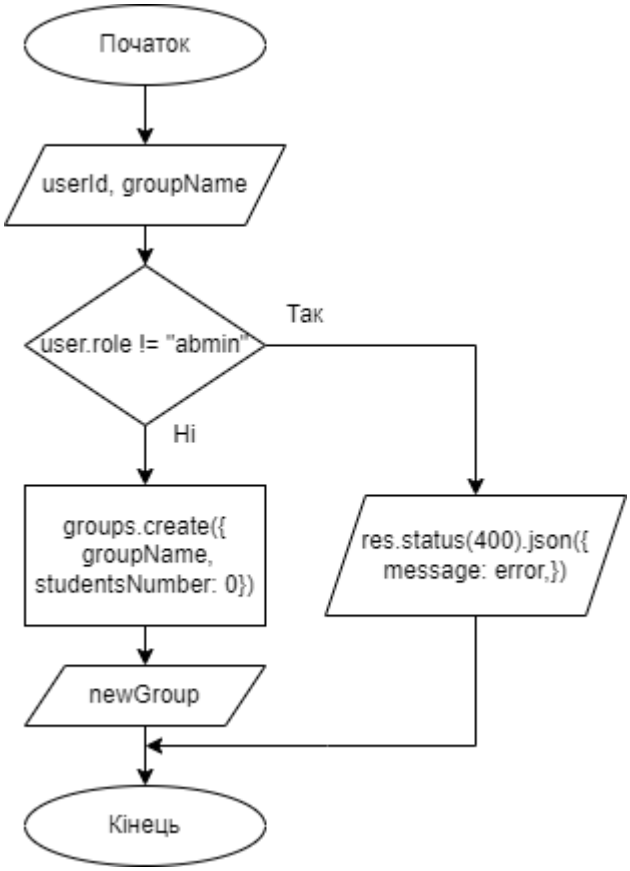

Рисунок Б.5 – Схема алгоритму створення навчальної групи

Продовження додатку Б

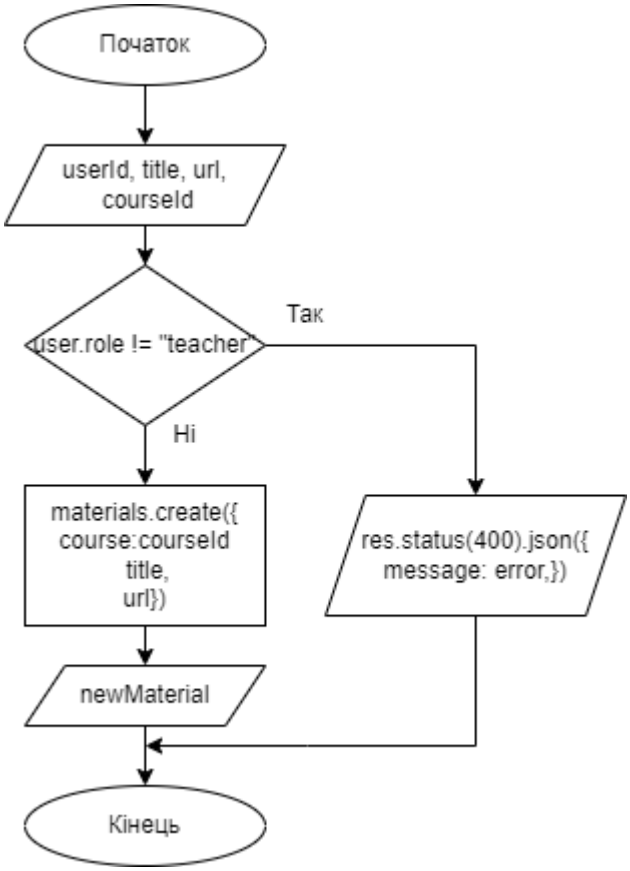

Рисунок Б.6 – Схема алгоритму створення навчального матеріалу

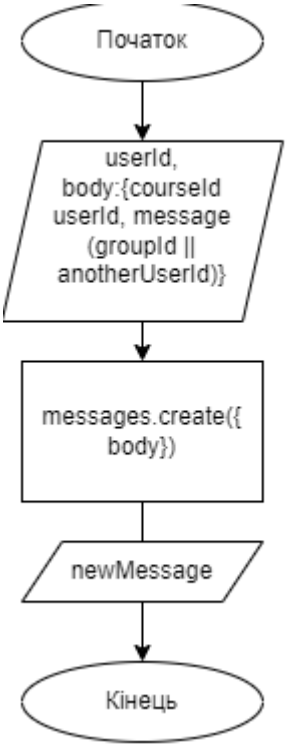

Рисунок Б.7 – Схема алгоритму створення повідомлення

Продовження додатку Б

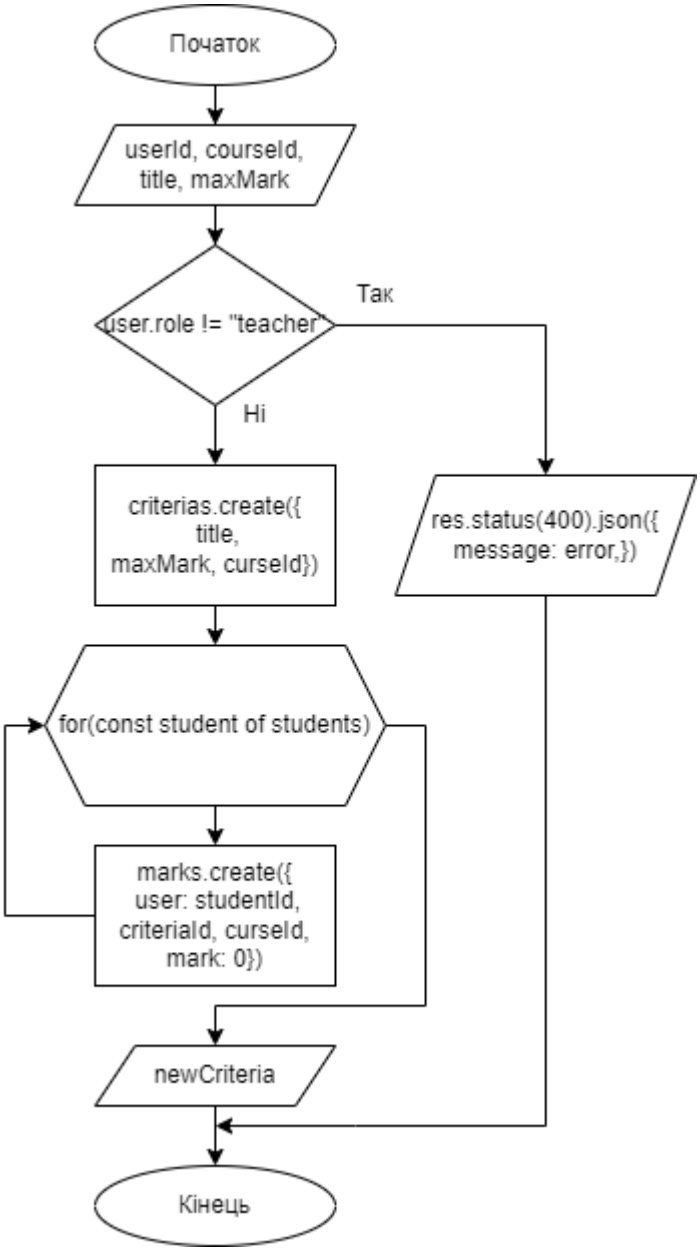

Рисунок Б.8 – Схема алгоритму створення критерію оцінювання

Продовження додатку Б

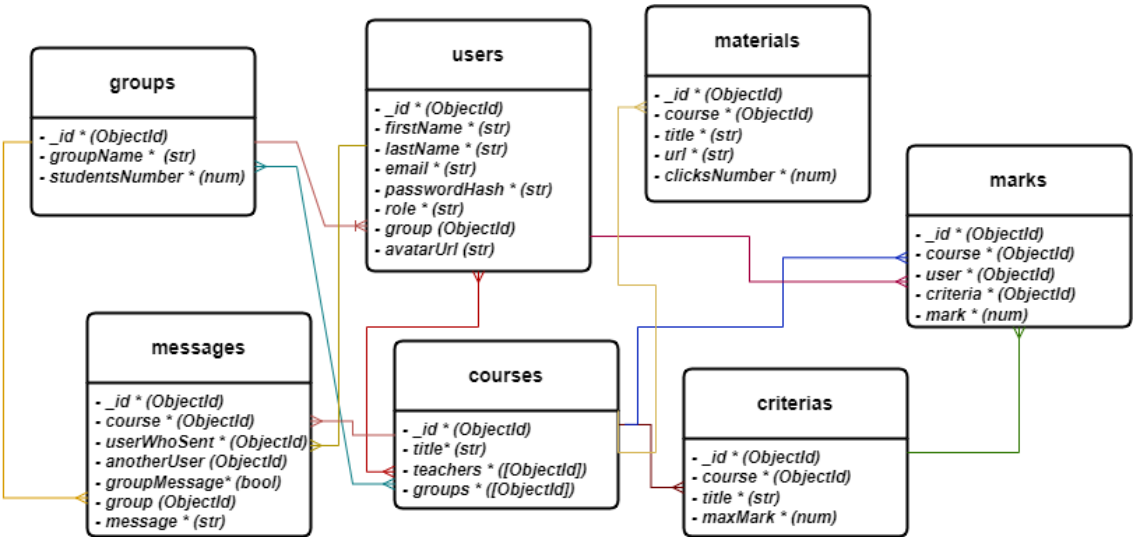

Рисунок Б.9 – Схема структури бази даних

Додаток В (обов'язковий)

Лістинг програмного забезпечення

Лістинг коду клієнтської частини сторінки курсу

```
const CoursePage = () => {
    const [me, setMe] = useState(null);
    const { id } = useParams();
    const [currentCourse, setCurrentCourse] = useState(null);
    const [sidebarValue, setSidebarValue] = useState("materials");

    useEffect(() => {
      const fetchMeData = async () => {
        const me = await fetchAuthMe();
        setMe(me);
      };

      const fetchCourseData = async () => {
        const course = await fetchGetCourseById(id);
        setCurrentCourse(course);
      };

      fetchCourseData();
      fetchMeData();
    }, []);

    const handleButtonClickMaterials = () => {
      setSidebarValue("materials");
    };

    const handleButtonClickChats = () => {
      setSidebarValue("chats");
    };

    const handleButtonClickResults = () => {
      setSidebarValue("results");
    };

    const handleButtonClickCriteria = () => {
      setSidebarValue("criteria");
    };

    if (!window.localStorage.getItem("token")) {
      return <Navigate to="/login" />;
    }

    return (
```

```
<>
  <div className={styles.coursePageWrraper}>
    <div className={styles.main}>
      <div className={styles.courseBlock}>
        <h1>Тема курсу: {currentCourse && currentCourse.title}</h1>
        <div className={styles.info}>
          Викладач курсу:
          <img
            src={currentCourse && currentCourse.teachers[0].avatarUrl}
            width={45}
            height={45}
          />
          {` ${currentCourse && currentCourse.teachers[0].firstName} ${
            currentCourse && currentCourse.teachers[0].lastName
          }`}
        </div>

        <div className={styles.info}>
          Група:
          {currentCourse && currentCourse.groups[0].groupName}
        </div>
      </div>

      {sidebarValue === "materials" && (
        <div>
          <div className={styles.createCourseBlock}>
            {me && me.role === "teacher" && currentCourse && (
              <CreateMaterial courseId={currentCourse._id} />
            )}
          </div>

          <div className={styles.groupsBlock}>
            {currentCourse && (
              <Materials courseId={currentCourse._id} me={me} />
            )}
          </div>
        </div>
      )}

      {sidebarValue === "chats" && (
        <div>
          <div className={styles.chatsBlock}>
            {me && currentCourse && (
              <Chats course={currentCourse} me={me} />
            )}
          </div>
        </div>
      )}

      {sidebarValue === "criteria" && (
        <div>
```

```
              <div className={styles.chatsBlock}>
                {me && currentCourse && (
                  <Criterias course={currentCourse} me={me} />
                )}
              </div>
            </div>
          )}

          {sidebarValue === "results" && (
            <div>
              <div className={styles.chatsBlock}>
                {me && currentCourse && (
                  <Results course={currentCourse} me={me} />
                )}
              </div>
            </div>
          )}
        </div>
        <div className={styles.sidebar}>
          <div onClick={handleButtonClickMaterials}>Матеріали</div>
          <div onClick={handleButtonClickChats}>Чати</div>
          {me && me.role === "teacher" && (
            <div onClick={handleButtonClickCriteria}>Критерії оцінювання</div>
          )}
          <div onClick={handleButtonClickResults}>Результати</div>
        </div>
      </div>
    </>
  );
};

export default CoursePage;
```

Лістинг коду серверної частини процесу оновлення навчальної групи студента

```
export const updateStudentGroup = async (req, res) => {
  try {
    const studentsId = req.params.id;

    const student = await usersModel.findOne({ _id: studentsId }).exec();

    let oldGroup = student.group ? student.group : null;

    const newStudent = await usersModel
      .findByIdAndUpdate(studentsId, {
```

```
        group: req.body.group,
      })
      .select("-passwordHash")
      .exec();

    if (oldGroup) {
      await groupsModel.findByIdAndUpdate(oldGroup, {
        $inc: { studentsNumber: -1 },
      });
    }
    await groupsModel.findByIdAndUpdate(req.body.group, {
      $inc: { studentsNumber: 1 },
    });

    res.json(newStudent);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "You do not have access",
    });
  }
};
```

Лістинг коду серверної частини процесу отримання всіх повідомлень конкретного курсу

```
export const getAllMessagesByCourseId = async (req, res) => {
  try {
    let messages;

    if (req.query.groupMessage === "true") {
      messages = await messagesModel
        .find({
          course: req.params.id,
          group: req.query.group,
        })
        .populate({
          path: "userWhoSent",
          model: "User",
        })
        .sort({ createdAt: 1 });
    } else if (req.query.groupMessage === "false") {
      messages = await messagesModel
        .find({
          course: req.params.id,
          $or: [
            {
              userWhoSent: req.query.firstUser,
```

```
            anotherUser: req.query.secondUser,
          },
          {
            userWhoSent: req.query.secondUser,
            anotherUser: req.query.firstUser,
          },
        ],
      })
      .populate({
        path: "userWhoSent",
        model: "User",
      })
      .sort({ createdAt: 1 });
  }
  res.json(messages);
 } catch (err) {
  console.log(err);
  res.status(500).json({
    message: "Fail creating",
  });
 }
};
```

Лістинг коду серверної частини процесу створення критерію оцінювання

```
export const createCriterias = async (req, res) => {
  try {
    const newCriteria = await criteriasModel.create({
      course: req.body.course,
      title: req.body.title,
      maxMark: req.body.maxMark,
    });

    const course = await coursesModel.findById(req.body.course).populate({
      path: "groups",
      model: "Group",
    });

    const students = await usersModel.find({
      role: "student",
      group: course.groups[0]._id,
    });

    for (const student of students) {
      await marksModel.create({
        user: student._id,
        course: course._id,
```

```
        criteria: newCriteria._id,
        mark: 0,
      });
    }

    res.json(newCriteria);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Fail creating",
    });
  }
};
```

Лістинг коду серверної частини процесу отримання критеріїв оцінювання конкретного курсу

```
export const getAllCriteriasByCourseId = async (req, res) => {
  try {
    const criterias = await criteriasModel
      .find({ course: req.params.id })
      .sort({ createdAt: 1 });

    res.json(criterias);
  } catch (err) {
    console.log(err);
    res.status(500).json({
      message: "Fail creating",
    });
  }
};
```

Додаток Г (обов'язковий)

Протокол перевірки навчальної (кваліфікаційної) роботи

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації

Показники звіту подібності Unicheck

Оригінальність 98,9% Схожість 1,1%

Аналіз звіту подібності (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Олександр САЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Євген ПАЛАМАРЧУК
(підпис) (ім'я, ПРІЗВИЩЕ)

Додаток Д (довідниковий)

Акт впровадження

УЗГОДЖЕНО
Директор центру дистанційної освіти
к.т.н., проф. Євген ПАЛАМАРЧУК
_____
_____
«____» _____2023 року

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«____» _____2023 року

АКТ ВПРОВАДЖЕННЯ № 3/ 04.12.2023
результатів науково-дослідних робіт

Замовник Центр дистанційної освіти Вінницького національного технічного університету *(найменування організації)*

Цим актом підтверджується, що результати роботи – «Удосконалення методів та засобів реалізації комунікацій в електронному освітньому середовищі», *(найменування теми)*

що виконав студент гр. 1ІСТ – 22м, Сай О.О. на громадських засадах *(виконавець)*

відповідно до плану роботи відділу дистанційної освіти ВНТУ на 2021-2022 рр. та впроваджено у Вінницькому національному технічному університеті *(строки виконання)* *(найменування організації, де здійснювалося впровадження)*

1. Вид впроваджених результатів: моделі та прототипи сервісів комунікацій *(експлуатація виробу, роботи, технології)*
2. Характеристика масштабу впровадження: одиничне
(унікальне, одиничне, партія, масовенал, серійне)
3. Форма впровадження: математичні та візуальні моделі; програмні компоненти
4. Новизна результатів науково-дослідної роботи розробці та впровадженні нових підходів комунікацій, які враховують сучасні вимоги та тенденції в галузі електронної освіти.
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)
5. Впроваджені: в системі JetIQ VNTU, для комунікативного контуру
6. Соціальний та науково-технічний ефект: удосконалення методів та засобів комунікацій для студентів та викладачів в електронному інформаційному середовищі.
(охорона навколишнього середовища, поліпшення й оздоровлення умов праці, удосконалення структури керування, науково-технічних напрямків, спеціальне призначення)

Від виконавця:
студент групи 1ІСТ-22м _____ Олександр САЙ