

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»

УДК 004.42.

371.3:004.4

«До захисту допущено»

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___» _____ 2025 р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення інформаційних систем»**

зі спеціальності 121 «Інженерія програмного

забезпечення» на тему: «Методи та програмне

забезпечення для проведення лабораторних занять»

Виконав (-ла):

студент (-ка) II курсу, групи ІІІ-43мп

Яремчук Денис Валентинович _____

Керівник:

доцент кафедри ІІІ, к.т.н, доц.

Фіногенов Олексій Дмитрович _____

Рецензент:

доцент кафедри ІСТ, к.т.н., доц.

Мамедова Катерина Юріївна _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів без
відповідних посилань.

Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки Кафедра інформатики та
програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення
інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«_____» _____ 2025р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Яремчук Денис Валентинович

1. Тема дисертації «Методи та програмне забезпечення для проведення лабораторних занять», науковий керівник дисертації Фіногенов Олексій Дмитрович, доцент кафедри ІІІ, к.т.н., затверджені наказом по університету від «06» листопада 2025 р. № 4841-с
2. Термін подання студентом дисертації «07» грудня 2025 р.
3. Об'єкт дослідження – Процес організації та проведення лабораторних занять з використанням сучасних інформаційних технологій.
4. Предмет дослідження – Методи, інструменти та програмне забезпечення, що реалізують підтримку лабораторних занять з використанням методу «чорної скрині».
5. Перелік завдань, які потрібно розробити – аналіз проблеми аналізу складних об'єктів та порівняльний аналіз існуючих рішень; розробка програмного забезпечення, що реалізує метод «чорної скрині» та ітеративний метод аналізу; перевірка якості програмного забезпечення.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 3 плакати
7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

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

9. Дата видачі завдання «01» вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Видача завдання	01.09.2025	
2	Визначення структури магістерської дисертації	08.09.2025	
3	Аналіз існуючих методів та програмних засобів	22.09.2025	
4	Вибір методів для реалізації в програмному забезпеченні	29.09.2025	
5	Проектування програмного забезпечення	13.10.2025	
6	Розробка програмного забезпечення	03.11.2025	
7	Перевірка якості програмного забезпечення	10.11.2025	
8	Оформлення пояснювальної записки	20.11.2025	
9	Подання дисертації на попередній захист	24.11.2025	
10	Подання дисертації на захист	07.12.2025	

Студент

Денис ЯРЕМЧУК

Науковий керівник

Олексій ФІНОГЕНОВ

РЕФЕРАТ

Розмір пояснювальної записки – 144 аркуші, містить 39 ілюстрацій, 52 таблиці, 3 додатки, 27 посилань на джерела.

Актуальність теми. Актуальність роботи зумовлена тим, що наявні засоби проведення лабораторних занять у закладах вищої освіти не повною мірою відповідають потребам викладачів і студентів. Традиційні підходи часто не враховують специфіку оптимізаційних задач і не дозволяють моделювати умови з обмеженою кількістю запитів чи їх частотою, що важливо для глибшого розуміння теми. Ситуацію ускладнює наявність великої кількості готових рішень в інтернеті, що зменшує мотивацію студентів до самостійного пошуку відповідей, а також перехід до дистанційного навчання, який обмежує безпосередній контроль викладача. Крім того, активне використання штучного інтелекту для автоматичного розв'язання завдань призводить до втрати навчальної цінності практичних робіт і потребує нових підходів до організації освітнього процесу.

Мета дослідження. Основною метою є наближення подібності умов виконання лабораторних робіт до реальних задач при аналізі складних об'єктів.

Об'єкт дослідження: Процес організації та проведення лабораторних занять з використанням сучасних інформаційних технологій.

Предмет дослідження: Методи, інструменти та програмне забезпечення, що реалізують підтримку лабораторних занять з використанням методу «чорної скрині».

Для реалізації поставленої мети **сформульовані наступні завдання:**

- провести аналіз систем автоматизованого тестування знань, видів аналізу складних об'єктів і визначити їхні переваги та недоліки;
- розробити вимоги до функціональності, архітектури та інтерфейсу вебзастосунку;
- розробити власне програмне забезпечення з використанням сучасних технологій з реалізацією взаємодії із системою у вигляді чорної скрині;
- перевірити якість розробленого рішення.

Наукова новизна результатів магістерської дисертації полягає в удосконаленні архітектури програмного забезпечення для проведення лабораторних занять, за рахунок імплементації методу «чорної скрині» та алгоритмів процесів виконання обчислювальних задач, що наближає аналіз складного об'єкту до реальних умов і вимагає від студентів приділяти увагу ефективності реалізації обчислювальних процедур та розуміння їх особливостей.

Практичне значення отриманих результатів полягає в тому, що розроблене програмне забезпечення створює умови для проведення повноцінних досліджень складного об'єкта з використанням ітераційних методів, завдяки приховуванню формалізованого опису об'єкта та реалізації механізму багаторазового запиту результатів моделювання.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Апробація. Наукові положення дисертації пройшли апробацію на IX Міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології SoftTech-2025» – м. Київ.

Публікації. Наукові положення дисертації опубліковані в:

- 1) Яремчук Д.В., Фіногенов О.Д. Методи та програмне забезпечення для проведення лабораторних занять // IX Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2025). Матеріали конференції. 26-28 листопада 2025 р. Київ.

Ключові слова: ОНЛАЙН-НАВЧАННЯ, МЕТОД «ЧОРНОЇ СКРИНІ», ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ НАВЧАННЯ, ЛАБОРАТОРНІ ЗАНЯТТЯ, ІТЕРАТИВНИЙ МЕТОД.

ABSTRACT

Explanatory note size – 144 pages, contains 39 illustrations, 52 tables, 3 applications, 27 references.

Topicality. The relevance of the work is due to the fact that the available tools for conducting laboratory classes in high education institutions do not fully meet the needs of teachers and students. Traditional approaches often do not take into account the specifics of optimization problems and do not allow modeling conditions with a limited number of requests or their frequency, which is important for a deeper understanding of the topic. The situation is complicated by the presence of a large number of ready-made solutions on the Internet, which reduces students' motivation to independently search for answers, as well as the transition to distance learning, which limits the direct control of the teacher. In addition, the active use of artificial intelligence for automatic problem solving leads to a loss of educational value of practical work and requires new approaches to organizing the educational process.

The aim of the study. The main goal is to approximate the similarity of the conditions for performing laboratory work to real tasks when analyzing complex objects.

Object of research: The process of organizing and conducting laboratory classes using modern information technologies.

Subject of research: Methods, tools and software that implement support for laboratory classes using the «black box» method.

To achieve this goal, the **following tasks** were formulated:

- to analyze automated knowledge testing systems, types of analysis of complex objects and determine their advantages and disadvantages;
- to develop requirements for the functionality, architecture and interface of the web application;
- to develop our own software using modern technologies with the implementation of interaction with the system in the form of a black box;
- to check the quality of the developed solution.

The scientific novelty of the results of the master's dissertation is improving the

architecture of the software for conducting laboratory classes, through the implementation of the «black box» method and algorithms for the processes of performing computational tasks, which brings the analysis of a complex object closer to real conditions and requires students to pay attention to the effectiveness of the implementation of computational procedures and understanding their features.

The practical value of the obtained results is that the developed software creates conditions for conducting full-fledged research of a complex object using iterative methods, due to hiding the formalized description of the object and implementing a mechanism for multiple requests for modeling results.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Computer Science and Software Engineering of the Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute».

Approbation. The scientific provisions of the dissertation were tested at the IX International Scientific and Practical Conference of Young Scientists and Students «Software Engineering and Advanced Information Technologies SoftTech-2025» – Kyiv.

Publications. The scientific provisions of the dissertation published in:

- 1) Yaremchuk D.V., Finogenov O.D. Methods and Software for Conducting Laboratory Classes. Materials of the 9th International Scientific and Practical Conference of Young Scientists and Students “Software Engineering and Advanced Information Technologies” (SoftTech-2025). November 26-28, 2025, Kyiv.

Keywords: ONLINE LEARNING, BLACK BOX METHOD, SOFTWARE FOR LEARNING, LABORATORY CLASSES, ITERATIVE METHOD.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	10
ВСТУП	11
1 АНАЛІЗ ПРОБЛЕМИ	13
1.1 Загальний огляд об'єкта дослідження	13
1.2 Опис задачі, що досліджується.....	14
1.3 Аналіз варіантів існуючих реалізацій для вирішення проблеми	16
1.4 Постановка задачі	27
Висновки до розділу	28
2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ	29
2.1 Аналіз функціональних вимог до програмного забезпечення	29
2.2 Аналіз нефункціональних вимог до програмного забезпечення	35
2.3 Опис архітектури програмного забезпечення.....	36
2.4 Опис технологій для розробки програмного забезпечення.....	43
Висновки до розділу	46
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	48
3.1 Опис схеми баз даних	48
3.2 Опис структури програмного продукту	51
3.3 Опис інтерфейсу програмного забезпечення	64
3.4 Тестування програмного забезпечення	70
Висновки до розділу	79
4 РОЗРОБКА СТАРТАП-ПРОЄКТУ	81
4.1 Опис ідеї проєкту	81
4.2 Технологічний аудит ідеї проєкту.....	82
4.3 Аналіз ринкових можливостей запуску стартап-проєкту.....	83

4.4	Розроблення ринкової стратегії проєкту	90
4.5	Розроблення маркетингової програми стартап-проєкту	93
	Висновки до розділу	95
	ВИСНОВКИ	97
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	99
	ДОДАТОК А	101
	ДОДАТОК Б	102
	ДОДАТОК В	142

ПЕРЕЛІК СКОРОЧЕНЬ

IDE	–	Integrated Development Environment, інтегроване середовище розробки
API	–	Application Programming Interface, прикладний програмний інтерфейс
IT	–	Інформаційні технології
ER	–	Entity-Relationship model
REST	–	Representational State Transfer
ОС	–	Операційна система
БД	–	База даних
СУБД	–	Система управління базами даних
ПЗ	–	Програмне забезпечення
JDBC	–	Java Database Connectivity
ORM	–	Object Relation Mapping
UI	–	User Interface

ВСТУП

Актуальність роботи зумовлена тим, що наявні засоби проведення лабораторних занять у закладах вищої освіти не повною мірою відповідають потребам викладачів і студентів. Традиційні підходи часто не враховують специфіку задач, що потребують моделювання реальних умов із обмеженою кількістю спроб або жорсткими часовими рамками, які є важливими для глибшого засвоєння матеріалу. Ситуацію додатково ускладнює перехід до дистанційного навчання, що обмежує можливість контролю виконання завдань, а також активне використання інструментів автоматичного розв'язання, що знижує навчальну цінність практичних робіт. Більшість існуючих рішень орієнтовані на тестування знань за допомогою стандартних форм завдань і не дозволяють аналізувати вхідні об'єкти без явного опису, що утруднює наближення процесу навчання до реальних практичних ситуацій. У зв'язку з цим виникає потреба у створенні спеціалізованого програмного забезпечення, що забезпечує взаємодію із завданнями на принципах «чорної скрині» та дозволяє ітеративно перевіряти гіпотези, отримуючи проміжні результати, що сприятиме підвищенню якості підготовки студентів інженерних спеціальностей.

Метою дослідження є наближення подібності умов виконання лабораторних робіт до реальних задач при аналізі складних об'єктів.

Об'єктом дослідження є процес організації та проведення лабораторних занять з використанням сучасних інформаційних технологій.

Предметом дослідження є методи, інструменти та програмне забезпечення, що реалізують підтримку лабораторних занять з використанням методу «чорної скрині».

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз існуючих систем автоматизованого тестування та визначити їхні функціональні обмеження;
- розробити вимоги до функціональності, архітектури та інтерфейсу вебзастосунку;
- створити програмне забезпечення з реалізацією взаємодії із

системою у вигляді «чорної скрині»;

– перевірити працездатність і якість розробленого рішення в умовах реального використання.

Наукова новизна результатів магістерської дисертації полягає в удосконаленні архітектури програмного забезпечення для проведення лабораторних занять, за рахунок імплементації методу «чорної скрині» та алгоритмів процесів виконання обчислювальних задач, що наближає аналіз складного об'єкту до реальних умов і вимагає від студентів приділяти увагу на ефективність реалізації обчислювальних процедур та розуміння їх особливостей.

Практичне значення отриманих результатів полягає в тому, що розроблене програмне забезпечення дозволяє зосередити увагу користувача на ефективній реалізації алгоритмів, що вивчаються, завдяки приховуванню формалізованого опису об'єкта, що наближає вирішення задачі аналізу складного об'єкту до практичних завдань. Його застосування сприяє підвищенню рівня об'єктивності оцінювання, мотивації студентів та формуванню практичних компетентностей.

1 АНАЛІЗ ПРОБЛЕМИ

1.1 Загальний огляд об'єкта дослідження

З появою навчання як такого, ще від часів перших шкіл і наставників, виникла природна потреба перевіряти, наскільки добре людина засвоїла нові знання (що і означає оцінювання). Таким чином, оцінювання було, стало і буде в подальшому невід'ємною частиною освітнього процесу, бо саме воно дозволяє визначити, чи досягнуто цілей навчання [1, с. 24–25]. Адже передача інформації має сенс лише тоді, коли її можна зрозуміти, запам'ятати й застосувати в повсякденному житті. Спочатку це робили дуже просто – учень повторював матеріал уголос або показував на практиці, чого навчився.

Однак, згодом, коли навчання стало масовим, з'явилися письмові завдання, екзамени та стандартизовані тести. Це показало недоліки традиційної, ручної перевірки знань. Коли кількість учнів та студентів зросла, а обсяг навчального матеріалу став в рази більшим, викладачі почали витрачати все більше часу на перевірку робіт. Це знижувало ефективність їхньої роботи та забиало фізичні ресурси, які можна було б присвятити навчанню чи консультаціям. Крім того, людський фактор у процесі оцінювання доволі часто призводить до суб'єктивності: оцінка може залежати від настрою, втоми або навіть особистого ставлення викладача до учня [2, с. 10-12]. До того ж, ручна перевірка сильно уповільнює отримання зворотного зв'язку – іноді результати доводиться чекати декілька дні чи навіть тижнів, що знижує мотивацію та ефективність навчання [3].

У сучасних умовах масовізації освіти проблема оперативного та об'єктивного оцінювання лише загострюється. Традиційні підходи до перевірки знань не справляються зі швидкістю розширення навчального контенту та збільшенням кількості студентів, що вимагає переходу до адаптивних цифрових інструментів оцінювання [4, с. 41-44].

З часом ці труднощі підштовхнули працівників освіти до пошуку нових рішень. Так, з'явилися автоматизовані системи перевірки знань – спочатку у вигляді простих комп'ютерних тестів, а потім і повноцінних застосунків, у тому

числі і веб-сайтів. Вони дозволяють перевіряти тести, завдання з вибором відповіді, короткі пояснення, розрахунки та інші завдання значно швидше, ніж це міг би зробити викладач вручну. Таким чином, автоматизація оцінювання не лише економить час, а й підвищує об'єктивність, адже усуває вплив людських упереджень [5, с. 45].

Сьогодні такі системи широко використовуються як у школах і вищих навчальних закладах, так і в онлайн-курсах та корпоративному навчанні. Вони дають змогу доволі швидко виявити прогалини у знаннях, надавати миттєвий зворотний зв'язок і допомагати студентам самостійно контролювати власний прогрес у навчанні. Саме автоматизація процесів контролю знань сприяє підвищенню ефективності навчання та прозорості оцінювання [6, с. 73].

1.2 Опис задачі, що досліджується

Всі існуючі застосунки для перевірки правильності відповідей введених учнем чи студентом обов'язково повинні надавати зручний графічний інтерфейс для виконання наступних операцій:

- створення запитання. Необхідно мати можливість вводити текст запитання, додавати зображення, таблиці, графіки, списки тощо;
- додавати правильні відповіді до запитань. Це можуть бути відповіді у вигляді так/ні, текстові відповіді. Також у запитання можуть бути декілька правильних варіантів відповідей. Або, наприклад, в запитанні може стояти задача співставити різні варіанти відповідей між собою;
- налаштування вартості питання серед всіх запитань тесту;
- налаштування певного обмеження на виконання завдання – кількість спроб чи дедлайн виконання завдання;
- видача завдання вчителем для учня;
- надання відповіді на запитання студентом;
- виставлення викладачем оцінки та коментаря для виконаного учнем завдання;
- перегляд оцінки учнем.

Більшість таких систем надають користувачам гарний, зручний, інтерактивний інтерфейс для виконання вказаних вище функцій. Однак, сучасні освітні платформи переважно розраховані на тестові формати і не пристосовані до задач, які потребують аналізу моделей або багатокрокових обчислень [7, с. 112-115].

По-перше, не у всіх додатках для виконання завдань існують одночасно декілька типів обмеження завдань. В абсолютній більшості додатків існує обмеження щодо дедлайну виконання завдання, трохи рідше – кількість спроб, ще рідше – кількість часу на одну спробу виконання завдання. Надання можливості зробити всі ці типи обмежень доступними для вибору одночасно, а також додатково зробити можливість міняти тип завдання прямо по ходу завдання виглядає логічним розвитком існуючих систем. До того ж в деяких дослідженнях вказується, що можливість гнучко змінювати параметри обмежень моделює реальні сценарії інженерної діяльності й повинна бути частиною сучасних навчальних платформ [8, с. 210-214]. Наприклад, зміна обмеження вже під час виконання завдання є доречною у випадку, коли викладач за активну роботу на парі додає студенту додаткові спроби для завдання. Ще краще було б зробити систему такою, щоб вона автоматично визначала тип та параметри обмеження в залежності від предмету завдання, або конкретного завдання, або завдання, що належить до предмету, який вивчається в певному університеті. Це автоматичне визначення буде відбуватися при наданні завдання студентам.

Необхідно розглянути ще одне обмеження. В більшості систем тестування механізм виконання завдань виглядає наступним чином: користувач вводить відповідь на завдання, надсилає її та отримує відповідь чи є ця відповідь правильною чи ні (в деяких випадках також показується правильна відповідь). І для великої кількості завдань такий алгоритм є прийнятним. Але в певних випадках необхідний зовсім інший механізм. Це ті випадки, коли необхідно зрозуміти поведінку певного об'єкта, що має формальний опис (зазвичай таким описом є функція одної або декілької змінних). У цьому випадку звичайний алгоритм описаний вище не буде надавати можливість саме аналізувати

поведінку об'єкта. Щоб надати користувачу таку можливість необхідно:

- приховати формальний опис (функцію) від користувача;
- надати користувачу можливість надавати відповіді та отримувати

результат функції ітеративно, тобто декілька разів. Це дозволить користувачу (якщо він правильно намагається аналізувати функцію) з кожною новою відповіддю (наданням значення змінних функції) розуміти вигляд, поведінку функції та знайти результат (зазвичай це пошук мінімуму чи максимуму функції). Саме багаторазова взаємодія за принципом «запит – відповідь» дозволяє студентам формувати аналітичне мислення та розуміння поведінки складних об'єктів [9, с. 98-102].

Такий підхід, разом із можливістю використовувати декілька типів обмеження завдання, дозволяє максимально приблизити завдання до реальних умов, адже в реальному світі далеко не завжди буде надаватися формальний опис об'єкта. Додатковим плюсом такого алгоритму є неможливість вирішення такого завдання пошуком відповіді в інтернеті використовуючи пошукові системи та штучний інтелект, такий як ChatGPT [10], Gemini [11], Grok [12] тощо, або, у випадку простої функції, простим перебором значень.

Для вирішення завдань описаних вище, також треба буде розробити систему прав доступу, щоб надавати обмеження для завдання лише для студентів з твого університету, предмету, групи, напрямку тощо.

Всі покращення описані в цьому розділі дозволять створити унікальне середовище для проведення різного роду лабораторних та контрольних заходів з предметів певного типу, наприклад «Методи оптимізацій функцій», «Чисельні методи», «Дискретна математика», «Методи машинного навчання» тощо. Ці вдосконалення дозволять студентам дійсно покращувати якість засвоєння знань та розуміння алгоритмів.

1.3 Аналіз варіантів існуючих реалізацій для вирішення проблеми

Існує дуже велика кількість платформ для виконання завдань та тестів:

- освітні: Moodle [13], «На Урок» [14], «Всеосвіта» (для дітей меншого віку) [15] і т. д.;

- платформи для проходження онлайн-курсів: Coursera [16], Udemу [17], Udacity [18] тощо;
- платформи для проходження олімпіад;
- внутрішньо-корпоративні платформи для проходження співбесід;
- веб-сайти для проходження вікторин та тестів на будь-які тематики: Google Forms [19], Kahoot! [20], Quizlet [21], Quizizz [22] тощо.

Для аналізу було обрано платформи та сайти із кожної категорії (окрім тих, що не мають публічного доступу): Moodle, Google Forms, Coursera.

Moodle (Modular Object-Oriented Dynamic Learning Environment) – це потужна, безкоштовна та відкрита (Open Source) система управління навчанням (LMS), яка дозволяє створювати персоналізовані навчальні середовища та організовувати дистанційне навчання.

Варто зазначити, що Moodle – це програмне забезпечення з відкритим вихідним кодом, а відтак будь-яка людина, установа чи компанія може завантажити його безкоштовно, встановити його на власному сервері і надати йому унікальний домен.

Основні функції, особливості та можливості Moodle детально описані в таблиці 1.1.

Таблиця 1.1 – Основні функції та особливості Moodle

Назва	Опис
Створення та організація курсів	Викладачі можуть легко створювати курси, наповнюючи їх різноманітними навчальними матеріалами (текст, відео, презентації, PDF, посилання)
Управління контентом	Підтримка завантаження файлів, використання мультимедіа та інтеграція контенту
Гнучке зарахування	Різні методи зарахування користувачів на курси, включаючи самостійну реєстрацію, ручне додавання та використання ключів доступу

Продовження таблиці 1.1

Назва	Опис
Персоналізований інтерфейс	Кожен користувач має власну сторінку, де відображаються його поточні, майбутні та минулі курси, а також стрічка новин/дедлайнів (рисунок 1.1)

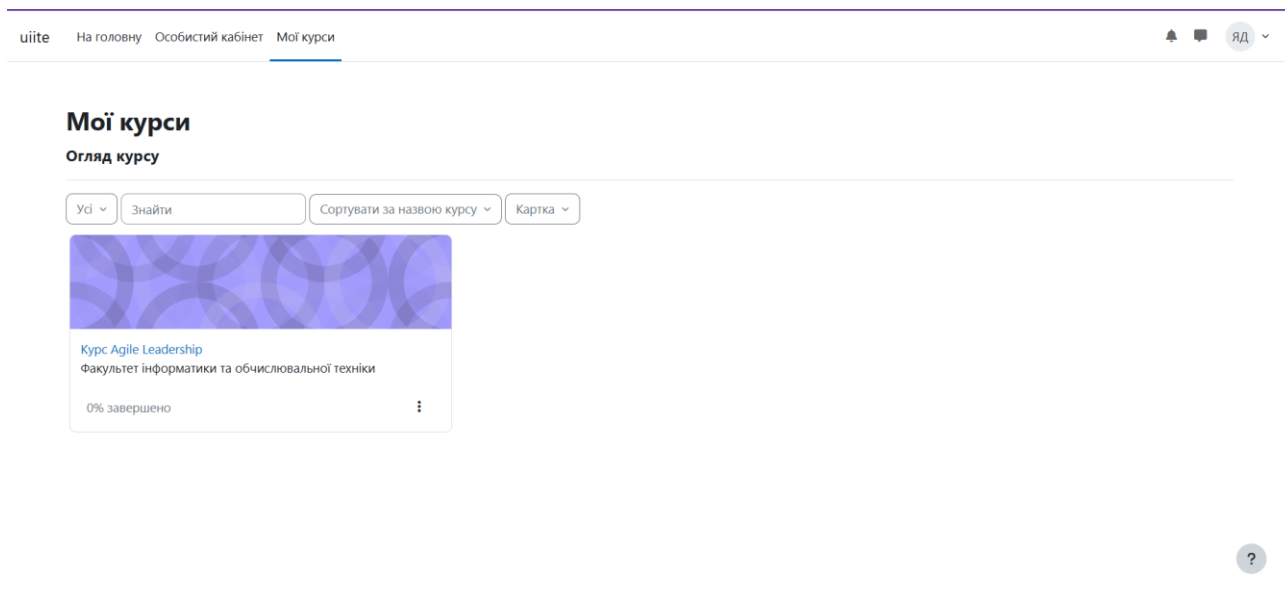


Рисунок 1.1 – Мої курси на Moodle

В контексті роботи, більший інтерес мають можливості пов'язані із процесом надання відповідей на завдання та отримання оцінок, які описані в таблиці 1.2.

Таблиця 1.2 – Можливості пов'язані із процесом надання відповідей на завдання та отримання оцінок

Назва	Опис
Види завдань	Широкий вибір типів тестових питань, які можна створювати (множинний вибір, коротка відповідь, есе, числова відповідь, перетягування) (рисунок 1.2)

Продовження таблиці 1.2

Назва	Опис
Оцінювання	Автоматичне оцінювання тестів, ручне оцінювання тестів викладачем, коментування та ведення обліку зданих робіт
Журнал оцінок	Централізована система для запису, відстеження та управління оцінками за всі елементи курсу (рисунки 1.3)
Аналітика навчання	Інструменти відстеження прогресу та активності студентів (Progress tracking), які допомагають викладачам визначити прогалини в навчанні

Питання 9
Відповіді ще не було
Макс. оцінка до 1,00
Відмітити питання
Редагувати питання

Улюблена пора року?
Відповідь:

Питання 10
Відповіді ще не було
Макс. оцінка до 1,00
Відмітити питання
Редагувати питання

Це хибне твердження
Виберіть одну відповідь:
☐ Правильно
☐ Неправильно

Питання 11
Відповіді ще не було
Макс. оцінка до 1,00

Це питання з пропущеним

Рисунок 1.2 – Надання відповідей на завдання

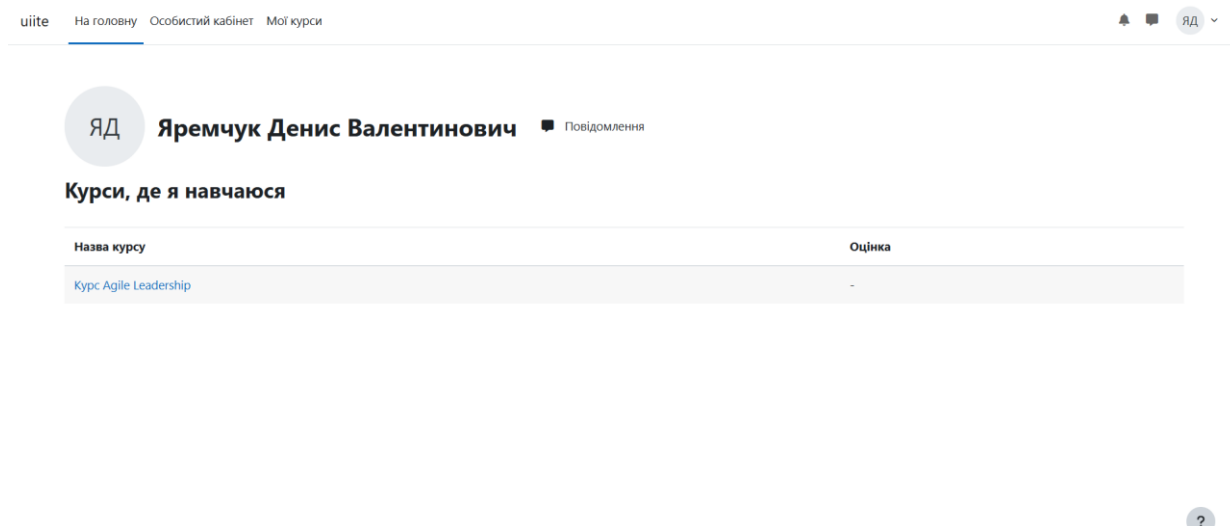


Рисунок 1.3 – Сторінка із оцінками студента

Coursera — це провідна світова платформа онлайн-навчання, яка пропонує навчальний контент від провідних університетів (наприклад, Стенфорд, Єль) та топових компаній (Google, IBM).

Coursera пропонує структуровану «навчальну екосистему» з різними рівнями складності та тривалості (рисунок 1.4), які описані в таблиці 1.3.

Таблиця 1.3 – Структуровану «навчальну екосистему» з різними рівнями складності та тривалості

Назва	Опис
Курси (Courses)	Одноразові навчальні модулі, що включають відеолекції, вправи та обговорення на форумах. Більшість із них можна пройти безкоштовно (отримати доступ до матеріалів без сертифіката);
Спеціалізації (Specializations)	Набори з 3-10 пов'язаних курсів, які дозволяють змістовно освоїти конкретну сферу знань або навичок

Продовження таблиці 1.3

Назва	Опис
Професійні сертифікати (Professional Certificates):	Комплексні програми, розроблені провідними компаніями (наприклад, Google, Meta, IBM) для підготовки до конкретних високозатребуваних професій (наприклад, Data Analyst, IT Support Specialist)

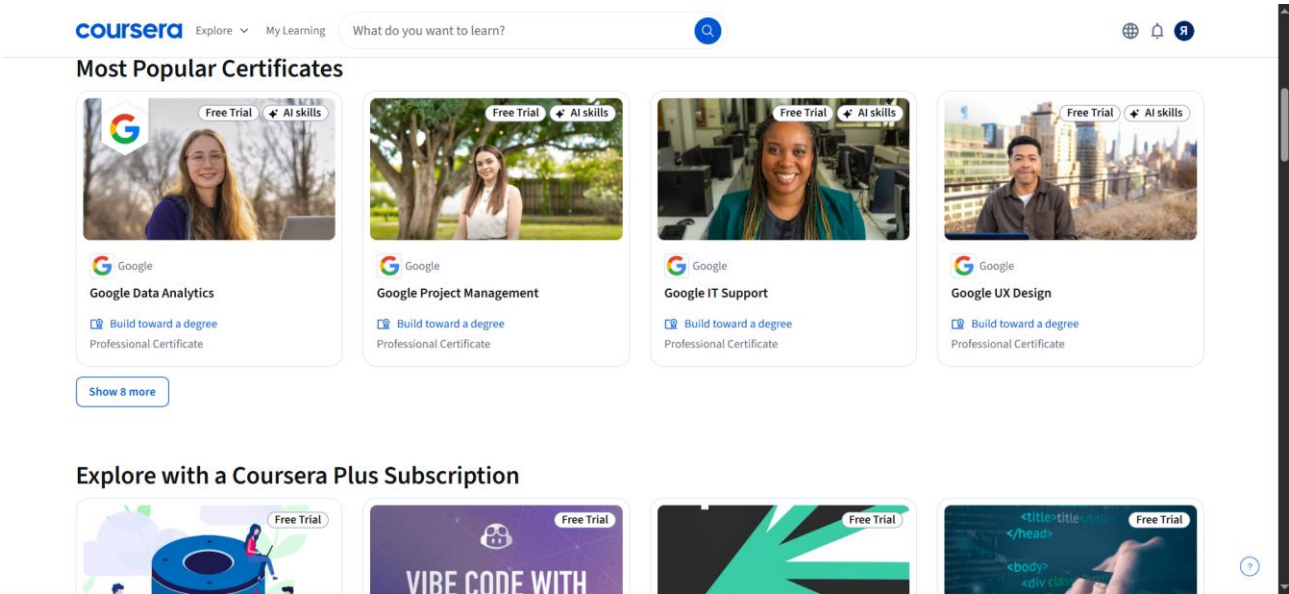


Рисунок 1.4 – Сторінка з курсами

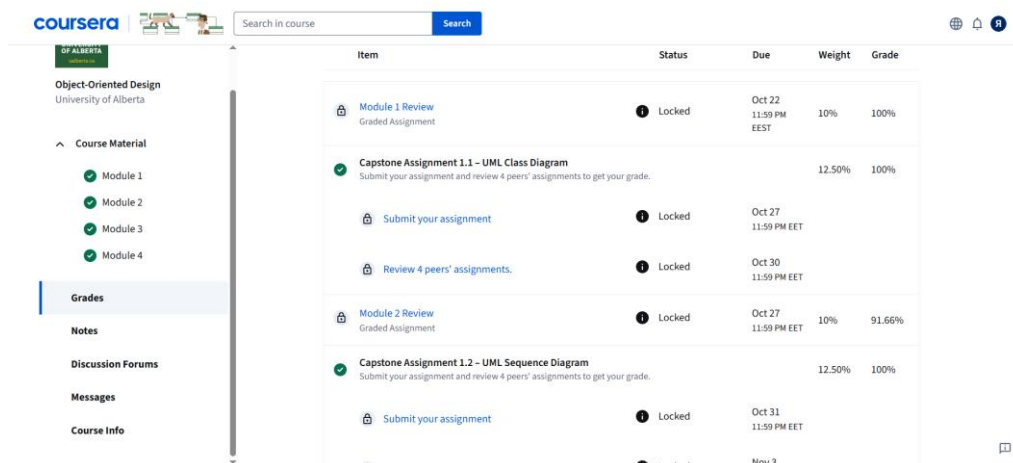
Існують також і інші рівні складності, однак немає необхідності загострювати на них увагу зараз, оскільки вони напряму не мають відношення до тематики роботи. Більш релевантні до роботи функції та особливості описані в таблиці 1.4.

Таблиця 1.4 – Функції Coursera, що мають стосунок до теми роботи

Назва	Опис
Гнучке навчання (Asynchronous Learning)	Студенти можуть переглядати лекції, виконувати завдання та спілкуватися на форумах у власному темпі, інтегруючи навчання у свій графік

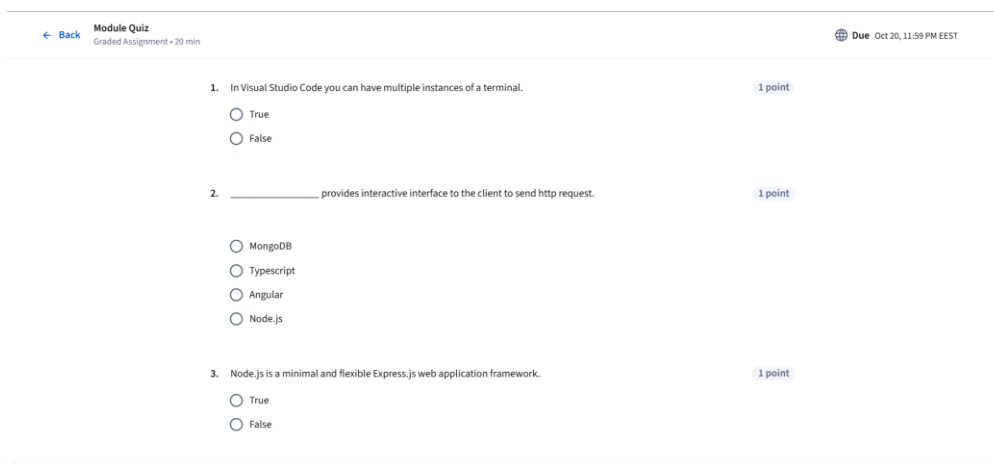
Продовження таблиці 1.4

Назва	Опис
Перевірка та оцінювання (рисунок 1.5)	Використовуються як автоматичні тести (Quizzes) (рисунок 1.6), так і взаємна перевірка робіт (Peer Review), що є поширеним механізмом для оцінювання відкритих та есеподібних завдань
Сертифікація та кредити	За умови успішного завершення програми та сплати, користувачі отримують верифіковані сертифікати (рисунок 1.7), які можуть бути використані для підвищення кваліфікації та, в деяких випадках, для зарахування університетських кредитів



Item	Status	Due	Weight	Grade
Module 1 Review Graded Assignment	Locked	Oct 22 11:59 PM EEST	10%	100%
Capstone Assignment 1.1 – UML Class Diagram Submit your assignment and review 4 peers' assignments to get your grade.			12.50%	100%
Submit your assignment	Locked	Oct 27 11:59 PM EET		
Review 4 peers' assignments.	Locked	Oct 30 11:59 PM EET		
Module 2 Review Graded Assignment	Locked	Oct 27 11:59 PM EET	10%	91.66%
Capstone Assignment 1.2 – UML Sequence Diagram Submit your assignment and review 4 peers' assignments to get your grade.			12.50%	100%
Submit your assignment	Locked	Oct 31 11:59 PM EET		
Review 4 peers' assignments.	Locked	Nov 3		

Рисунок 1.5 – Сторінка з оцінками



← Back Module Quiz
Graded Assignment • 20 min Due Oct 20, 11:59 PM EEST

- In Visual Studio Code you can have multiple instances of a terminal. 1 point

☐ True

☐ False
- _____ provides interactive interface to the client to send http request. 1 point

☐ MongoDB

☐ Typescript

☐ Angular

☐ Node.js
- Node.js is a minimal and flexible Express.js web application framework. 1 point

☐ True

☐ False

Рисунок 1.6 – Сторінка із наданням відповіді на завдання

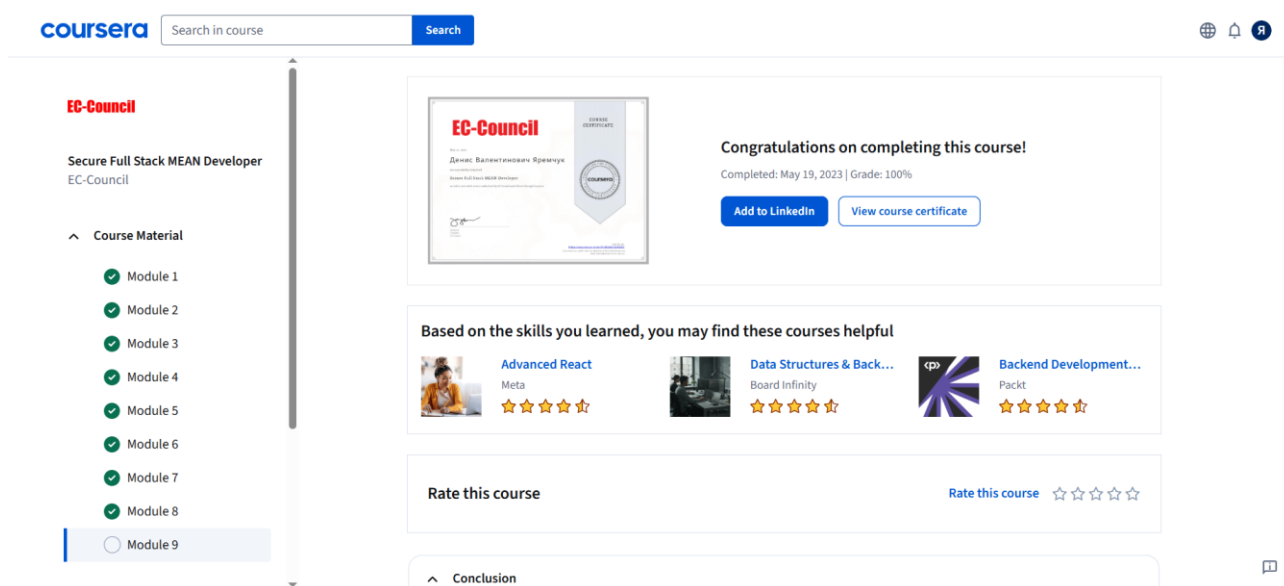


Рисунок 1.7 – Сторінка із сертифікатом

Google Форми (Google Forms) – це безкоштовний, хмарний інструмент, що є частиною пакету Google Workspace. Його основне призначення – створення онлайн-форм для збору інформації, проведення опитувань, тестування та реєстрації. Основні функції Google Forms:

- створення форм та запитань (рисунок 1.8): всі можливості описані в таблиці 1.5.

Таблиця 1.5 – Можливості створення форм та запитань в Google Forms

Назва	Опис
Простий конструктор	Інтуїтивно зрозумілий інтерфейс «перетягування» (drag-and-drop) дозволяє швидко створювати форми без навичок програмування
Різноманітність типів запитань	Коротка відповідь, абзац (довга відповідь), множинний вибір, прапорці (вибір кількох варіантів), спадний список, лінійна шкала (рейтинг), сітки, дата та час

Продовження таблиці 1.5

Назва	Опис
Додавання медіа	Можливість легко додавати зображення та відео до питань чи розділів для візуалізації або надання додаткового контексту
Розділи та логіка переходів	Форму можна розділити на окремі сторінки (розділи). Можна налаштувати умовну логіку, щоб респондент переходив до різних розділів залежно від своєї відповіді на попереднє запитання
Завантаження файлів	Можливість дозволити респондентам завантажувати файли (документи, фото) безпосередньо у форму (потрібен вхід в обліковий запис Google)

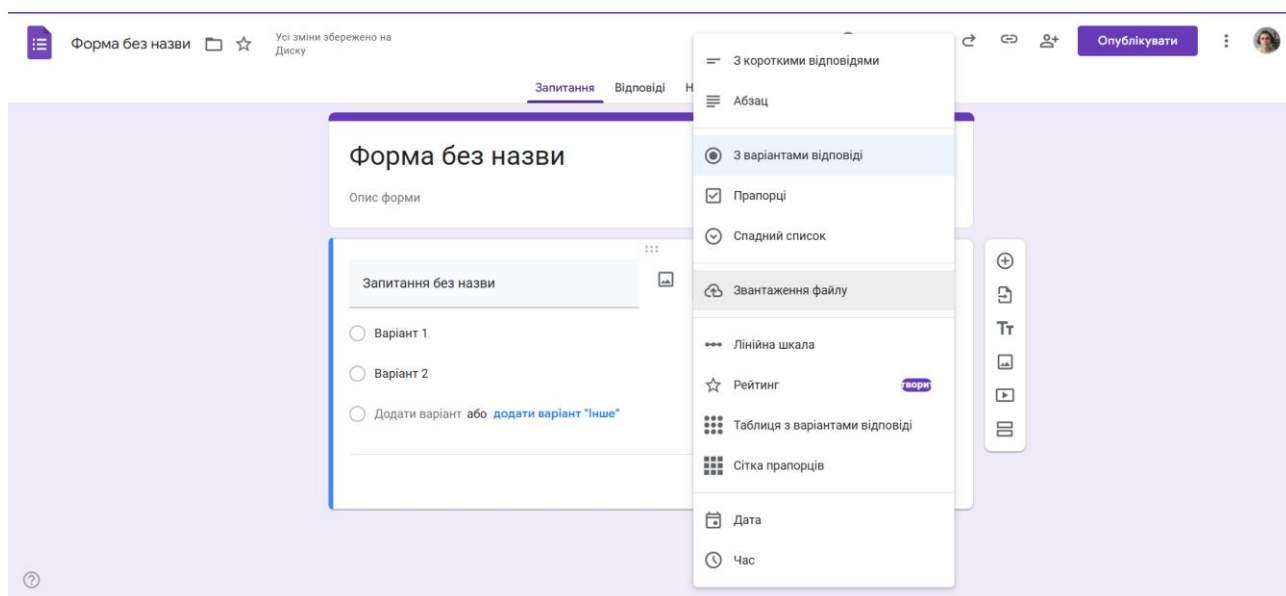


Рисунок 1.8 – Створення форми

– функції тестування (рисунок 1.9): всі можливості описані в таблиці 1.6;

Таблиця 1.6 – Можливості тестування в Google Forms

Назва	Опис
Перетворення на тест	У налаштуваннях можна перетворити будь-яку форму на тест (вікторину)
Ключі відповідей та бали	Можливість задати правильні відповіді та призначити бали за кожне запитання
Автоматичне оцінювання	Для питань із закритими відповідями (множинний вибір, прапорці) оцінка відбувається автоматично
Відгук	Можна додати автоматичний зворотний зв'язок (коментар) для правильних і неправильних відповідей

Рисунок 1.9 – Проходження форми

— збір та аналіз відповідей (рисунок 1.10): всі можливості описані в таблиці 1.7.

Таблиця 1.7 – Можливості збору та аналізу відповідей в Google Forms

Назва	Опис
Автоматичний збір даних	Усі відповіді збираються автоматично в режимі реального часу

Продовження таблиці 1.7

Назва	Опис
Візуалізація результатів	Автоматично генерація діаграм та графіків для швидкого візуального аналізу агрегованих даних (відповідей користувачів)
Інтеграція з Google Таблицями (Google Sheets)	Одним натисканням усі необроблені дані можна експортувати та синхронізувати з новою або наявною Google Таблицею для глибшого аналізу, фільтрації та використання формул

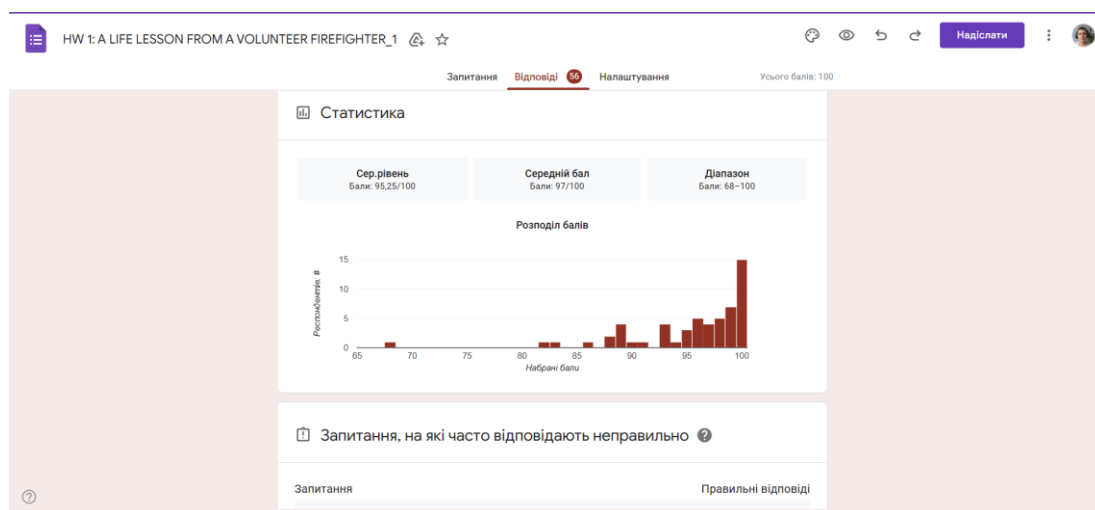


Рисунок 1.10 – Відповіді на тест

Після огляду та аналізу існуючих варіантів сформуємо порівняльну таблицю, щоб дізнатися які функції виконують існуючі аналоги, а які можливості буде мати розроблюваний застосунок (таблиця 1.8).

Таблиця 1.8 – Порівняльна таблиця існуючих аналогів та розроблюваного застосунку

Функція	Moodle	Coursera	Google Forms	Застосунок
Надання відповідей на питання	+	+	+	+

Продовження таблиці 1.8

Функція	Moodle	Coursera	Google Forms	Застосунок
Оцінювання	+	+	+	+
Наявність великої кількості типів питань	+	+	+	-
Продвинутий графічний інтерфейс для створення тестів	+	-	+	-
Можливість встановлення декількох типів обмежень на завдання одночасно	-	-	-	+
Взаємодія з системою по типу «чорної скриньки»	-	-	-	+

Розроблюваний застосунок не буде реалізовувати функціонал, який наявний у аналогах, але він не потрібен у продукті, який буде, в основному, спеціалізуватися на тестуванні завдань, де необхідно аналізувати вхідний об'єкт згідно результатів отриманих користувачем після введення ним відповідей.

1.4 Постановка задачі

На основі проведеного детального аналізу предметної області та існуючих рішень визначено, що наявні системи володіють недостатнім функціоналом для проведення лабораторних та контрольних заходів з предметів, для яких необхідно надати користувачу можливість аналізувати вхідний об'єкт без деталізованого опису завдання. Це дозволить приблизити вирішення завдань до реальних умов, що надасть практичного значення розробленому додатку.

Отже, для досягнення мети роботи, повинен бути розроблений програмний продукт, що матиме базові функції, що існують в усіх схожих системах:

- створення завдань;

- можливість студента надавати відповіді на завдання;
- можливість викладача оцінювати виконані завдання та надання коментаря щодо виконаної роботи;
- можливість студента переглядати отримані оцінки та коментарі.

До існуючого базового функціоналу необхідно додати спеціалізовані вдосконалення:

- можливість обмежити надання відповідей декількома способами причому, щоб тип обмеження можна було задати для конкретних питань перед початком їх виконання або вже під час їх виконання;
- завдання, які не матимуть текстового чи графічного опису;
- ітеративне надання відповідей та отримання результатів;
- систему прав різних рівнів для викладачів та адміністраторів системи.

Висновки до розділу

В першому підрозділі – «Загальний огляд об'єкта дослідження» – було описано яким чином спосіб оцінювання студентів еволюціонував від ручної перевірки до автоматизованих систем.

В другому підрозділі – «Опис задачі, що досліджується» – було розписано, які задачі виконують існуючі системи, а також піднято декілька питань щодо того як ці системи можна покращити.

В третьому підрозділі – «Аналіз варіантів існуючих реалізацій для вирішення проблеми» – було проаналізовано існуючі реалізації для вирішення проблеми та порівняно ці застосунки до продукту, що буде реалізований в межах написання цієї роботи

В четвертому підрозділі – «Постановка задачі» – було зроблено підсумок попередніх розділів і написано завдання, що мають бути вирішені в цьому дослідженні для вирішення описаних проблем та досягнення мети роботи.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ

2.1 Аналіз функціональних вимог до програмного забезпечення

Головними функціями програмного забезпечення є призначення завдання студенту, виконання завдання студентом, виставлення оцінки викладачем та можливість виконання всіх доступних операцій без обмеження графічного інтерфейсу. Загалом всі функції застосунку детально описані в таблиці 1.

Таблиця 2.1 – Перелік функціональних вимог

Модуль	Назва та опис функціональних вимог
Реєстрація та авторизація	Реєстрація користувача. Неавторизований користувач може зареєструватися як студент, викладач чи адміністратор
	Відновлення паролю. Зареєстрований користувач може змінити пароль до свого облікового запису
	Авторизація з використанням електронної пошти та паролю. Зареєстрований користувач може увійти до свого облікового запису за допомогою електронної пошти та паролю
	Авторизація з використанням ключа. Зареєстрований користувач може увійти до свого облікового запису за допомогою унікального для цього користувача ключа
	Вихід із облікового запису. Авторизований користувач може вийти із свого облікового запису
Керування університетами	Створення університету. Авторизований користувач може створити університет
	Редагування університету. Авторизований користувач може відредагувати університет

Продовження таблиці 2.1

Модуль	Назва та опис функціональних вимог
Керування університетами	Видалення університету. Авторизований користувач може видалити університет
	Перегляд університетів. Авторизований користувач може переглядати університети
	Пошук університетів. Авторизований користувач може шукати університети за назвою
Керування предметами	Створення предмету. Авторизований користувач може створити предмет для конкретного університету
	Редагування предмету. Авторизований користувач може відредагувати предмет конкретного університету
	Видалення предмету. Авторизований користувач може видалити предмет конкретного університету
	Перегляд предметів. Авторизований користувач може переглядати предмети
	Пошук предметів. Авторизований користувач може шукати предмети за назвою
Керування функціями	Створення функції. Авторизований користувач може створити функцію для певного предмету
	Редагування функції. Авторизований користувач може відредагувати функцію певного предмета

Продовження таблиці 2.1

Модуль	Назва та опис функціональних вимог
Керування функціями	Видалення функції. Авторизований користувач може видалити функцію певного предмета
	Перегляд функцій. Авторизований користувач може переглядати функції
	Пошук функцій. Авторизований користувач може шукати функції за назвою
Виконання завдання	Призначення випадкового завдання. Авторизований користувач може призначити собі випадкове завдання
	Призначення завдання для групи студентів. Авторизований користувач може призначити завдання студентам, що входять до групи, яка вивчає конкретні предмети
	Надання відповіді на завдання. Авторизований користувач може надати відповідь на завдання причому надання відповіді має бути обмежене 1 із 3 способів: обмеження кількості спроб для надання відповіді, налаштування дедлайну для надання відповідей, обмеження часу для надання 1 відповіді
	Завершення виконання завдання. Авторизований користувач може завершити виконання завдання
	Перегляд відповідей завдання. Авторизований користувач може переглядати відповіді на конкретне завдання

Продовження таблиці 2.1

Модуль	Назва та опис функціональних вимог
Виконання завдання	Перегляд завдань. Авторизований користувач може переглядати завдання
Керування оцінками	Виставлення оцінки. Авторизований користувач може виставляти оцінки за завдання
	Перегляд оцінок. Авторизований користувач може переглядати оцінки за завдання
Керування групами	Створення групи. Авторизований користувач може створити групу, вказавши предмети, що будуть вивчатися в групі та студентів, що будуть входити до групи
	Редагування групи. Авторизований користувач може відредагувати групу, вказавши предмети, що будуть вивчатися в групі та студентів, що будуть входити до групи
	Видалення групи. Авторизований користувач може видалити групу
	Перегляд груп. Авторизований користувач може переглядати групи
	Пошук груп. Авторизований користувач може шукати групи за назвою
Керування користувачами	Підтвердження користувачів. Авторизований користувач може підтвердити користувачів
	Відхилення користувачів. Авторизований користувач може відхилити користувачів

Продовження таблиці 2.1

Модуль	Назва та опис функціональних вимог
Керування користувачами	Перегляд користувачів. Авторизований користувач може переглядати користувачі системи
	Пошук користувачів. Авторизований користувач може шукати користувачів за іменем та прізвищем
Керування правами користувача	Надання прав користувачу. Авторизований користувач може надати іншому користувачу права на виконання дій щодо університетів, функцій, предметів, завдань
	Позбавлення прав користувача. Авторизований користувач може забрати в іншого користувача права на виконання дій щодо університетів, функцій, предметів, завдань
Керування обмеженням виконання завдань	Зміна обмеження виконання завдання по замовчуванню. Авторизований користувач може змінити обмеження виконання завдання (кількість спроб, дедлайн надання відповідей, час для надання 1 спроби) по замовчуванню
	Зміна обмеження виконання поточних завдань. Авторизований користувач може змінити обмеження виконання поточних завдань (кількість спроб, дедлайн надання відповідей, час для надання 1 спроби)
Надсилання запитів	Перегляд всіх доступних запитів. Авторизований користувач може переглядати всі доступні запити

Продовження таблиці 2.1

Модуль	Назва та опис функціональних вимог
Надсилання запитів	Пошук запитів. Авторизований користувач може шукати запити за текстом запиту
	Пошук полів запиту. Авторизований користувач може шукати поля певного запиту
	Надсилання запитів. Авторизований користувач може надсилати запити на виконання будь-яких дій (окрім авторизації чи реєстрації)
	Отримання відповіді на запит. Авторизований користувач може отримувати відповіді на надісланий запит
Надсилання електронних листів	Надсилання електронних листів. При виконанні певних дій, користувач може отримати відповідний електронний лист

Для правильного розподілу функцій між користувачами, у додатку повинні бути реалізовані 4 ролі користувачів: неавторизований користувач, студент, викладач університету, адміністратор системи. Розподіл функцій між ролями користувачів зображений у вигляді Use Case діаграми на рисунках 2.1 – 2.2.

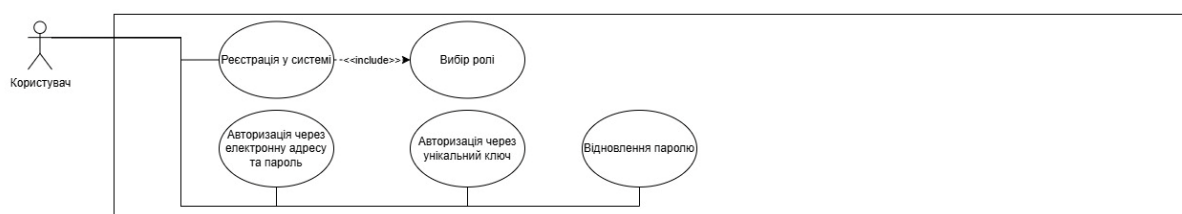


Рисунок 2.1 – Use Case діаграма неавторизованого користувача

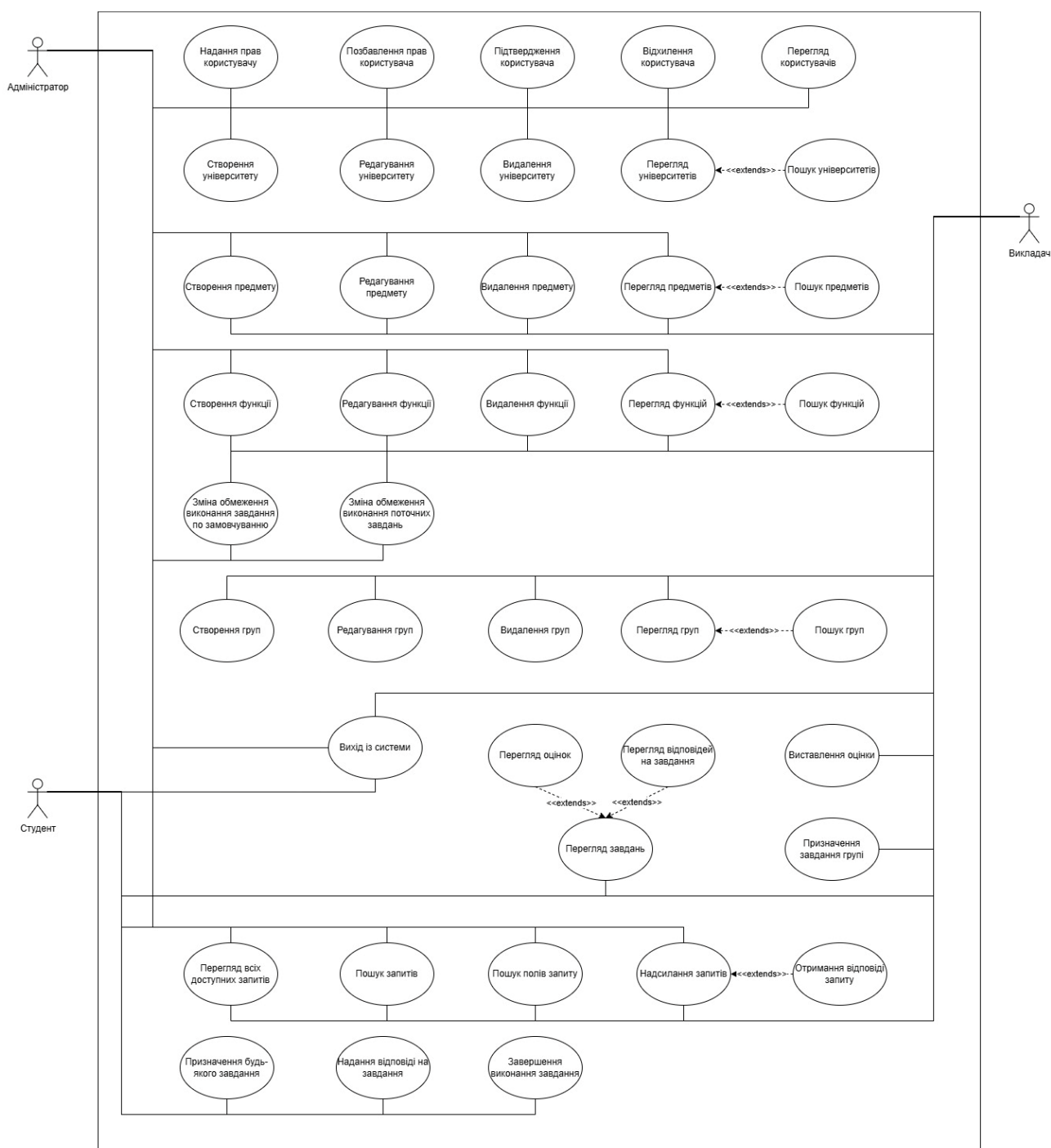


Рисунок 2.2 – Use Case діаграма викладача, адміністратора та студента

Згідно рисунку 2.2, можна зробити висновок, що користувач із відповідною роллю буде мати доступ лише до необхідних йому функцій.

2.2 Аналіз нефункціональних вимог до програмного забезпечення

В розроблюваному веб-додатку повинні бути дотримані такі нефункціональні вимоги:

- безпека (security): система повинна бути захищена від наступних вразливостей: CSRF, SQL Injection, XSS. Роль користувача та його права повинні враховуватися при виконанні дій в системі;
- продуктивність (performance): система повинна працювати без збоїв, а також підтримувати одночасну діяльність 250 користувачів. При такому навантаженні, час відповіді сервера не повинен бути більше 4 секунд для кожної операції користувача. Це необхідно, оскільки при виконанні завдань, одночасно на сервері може знаходитися багато студентів і для них критично, щоб система швидко реагувала на їхні дії;
- доступність (availability): система повинна бути доступна 99.99% часу;
- сумісність (compatibility): система повинна працювати в різних браузерах на різних ОС, а очікуваний результат виконання операцій повинен бути однаковим в незалежності від ОС та браузера;
- зручність у використанні (usability): система не повинна мати двозначний інтерфейс. Це означає, що користувач повинен розуміти, що буде відбуватись після виконання операції в додатку. Якщо користувач неправильно взаємодіяв із системою, то система повинна повідомляти користувача про помилки;
- модифікованість (modifiability): система має бути зроблена так, щоб пізніше, при реалізації нових функцій, інші функції працювали згідно функціональних вимог.

2.3 Опис архітектури програмного забезпечення

Вибір архітектури розпочинається із визначення частин екосистеми додатків. Повинен існувати бекенд-сервер, який буде отримувати запити на виконання дій від користувача. На ньому будуть проводитися всі складні операції отримання даних та збереження даних введених користувачем. Також, користувачу має мати можливість робити операції через зручний графічний інтерфейс, тобто має бути розроблений фронтенд-додаток.

Однак, надання користувачу одного інтерактивного не застосунку буде достатнім для досягнення мети роботи, а саме максимальне наближення виконання завдань до реальних умов. Саме тому необхідно розробити окремий додатковий застосунок для надсилання будь-яких запитів напряму на сервер, без використання графічного додатку, який буде виконувати наперед визначені операції.

Тепер, коли визначено, що буде 2 фронтенд-додатки та 1 сервер, необхідно визначитися з архітектурою. Для цього, спочатку, потрібно ознайомитися з існуючими підходами до розробки архітектури веб-додатків:

- моноліт: всі функції програми (клієнтський інтерфейс та сервер, де оброблюються запити користувача) об'єднані в один повноцінний додаток та виконуються як один процес. Це підхід, в якому весь функціонал знаходиться в одному місці [23];
- мікросервіси: програма складається з декількох невеликих повністю незалежних компонентів (сервісів), які взаємодіють між собою або напряму або через спеціалізований компонент – gateway;
- трирівнева архітектура: дуже поширена в багатьох існуючих застосунках, оскільки розділяє весь додаток на 3 рівні:
 - а) Презентаційний (клієнт): користувацький інтерфейс;
 - б) Бізнес-логіка (сервер): обробка запитів та виконання завдань;
 - в) Рівень даних: зберігання та управління даними в базі даних [24].

Всі архітектури зазначені вище мають свої переваги та недоліки. Наприклад, монолітну архітектуру зручно використовувати, якщо кількість функцій в додатку є невеликою, в такому випадку не потрібно розділяти додаток на декілька частин. Але при цьому, при розширенні застосунку, додання кожної нової функції буде вимагати все більше і більше часу, бо при зміні чи доданні нової функції, необхідно перевіряти чи існуючі операції виконуються як очікується.

Мікросервісну архітектуру зручно використовувати, коли необхідно

реалізувати великий додаток, який можна розділити на невеликі незалежні частини. Однак, в такому випадку необхідно налаштувати правильний зв'язок між всіма компонентами системи, що інколи може бути складно реалізувати.

Монолітну та мікросервісну архітектуру доволі часто використовують в різноманітних додатках, але використовують їх для специфічних задач. Для розроблюваного додатку достатньо буде використовувати 3-рівневу архітектуру, оскільки вона містить всі необхідні елементи:

- рівень представлення: на ньому будуть знаходитися 2 фронтенд-додатки, що нададуть користувачу можливість по-різному взаємодіяти із сервером;
- рівень бізнес-логіки: на ньому знаходиться сервер, на якому виконуються бізнес-операції;
- рівень даних: на ньому буде знаходитися сервер бази даних, де буде знаходитися інформація необхідна для функціонування системи.

Єдина відмінність від класичної 3-рівневої архітектури, це наявність 2 фронтенд додатків на рівні представлення.

Схему функціонування додатку, що використовує графічний інтерфейс, як можливість надіслати наперед заданий запит, можна представити таким чином:

- а) Користувач виконує певну операцію: натискає кнопку, переходить на сторінку, вводить інформацію в полі для вводу тощо.
- б) Виконання цієї дії призводить до надсилання запиту на сервер;
- в) У відповідь сервер надсилає або не надсилає (в залежності від конкретного запиту) відповідь, яка певним чином оброблюється в додатку.

Схему функціонування додатку, що дозволяє надсилати будь-які запити на сервер можна описати так:

- а) Користувач вводить адресу запиту, вводить інформацію необхідну для запиту;
- б) Користувач надсилає запит на сервер;
- в) У відповідь сервер надсилає або не надсилає (в залежності від конкретного запиту) відповідь, яка показується в додатку в необробленому

вигляді.

Для рівня бізнес-логіки, а саме серверної частини, процес обробки запитів краще охарактеризувати у контексті компонентів бекенд-додатку, до яких входить:

- REST-контролер: у моменті, коли з фронтенд-додатку надсилається будь-який запит на сервер, його приймає саме контролер. Він визначає метод запиту, ідентифікатор запиту (/api/users/1 тощо), тіло запиту, та параметри запиту. Сама інформація, що приходить на сервер, приходить у вигляді JSON-об'єкту, але, для зручності, сервер автоматично перетворює її у необхідний формат (у вигляд класів);
- сервіс: коли REST-контролер приймає запит він не виконує всі операції сам, для цього він використовує сервіси. По факту, сервіси оброблюють отриманий запит певним чином та зберігають його в базу даних. Сервіси можуть використовувати інші сервіси, якщо це необхідно;
- репозиторій: щоб зберегти інформацію оброблену в сервісі або отримати інформацію з бази даних та передати її користувачу, використовуються репозиторії. Репозиторії напряду взаємодіють із сервером бази даних.

Всі компоненти серверу розділені згідно сутності, яка є об'єктом операції. Наприклад, якщо є такі сутності, як користувач чи профіль, то створюються контролери, сервіси, репозиторії окремо для виконання операцій з користувачем та профілем.

Останній рівень архітектури – це рівень даних. Щоб забезпечити збереження даних використовуються сервери баз даних. Взаємодія між сервером баз даних та репозиторієм на рівні бізнес-логіки відбувається через надсилання спеціалізованих запитів із репозиторію на сервер баз-даних та отримання чи збереження інформації.

Всі компоненти, що взаємодіють в додатку доволі зручно зобразити використовуючи C4 нотацію, а саме 1 її рівень – рівень програмних систем (рисунок 2.3), та 2 рівень – рівень програмних контейнерів (рисунок 2.4).

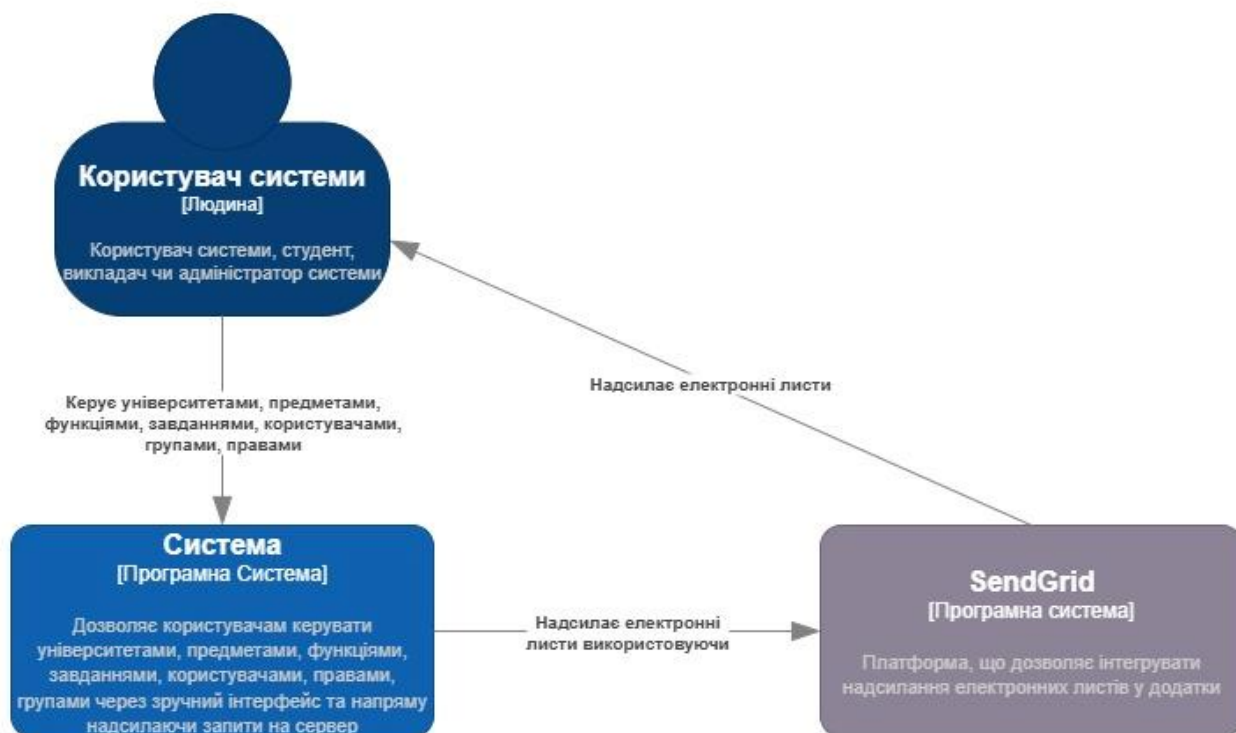


Рисунок 2.3 - Діаграма 1-ого рівня C4

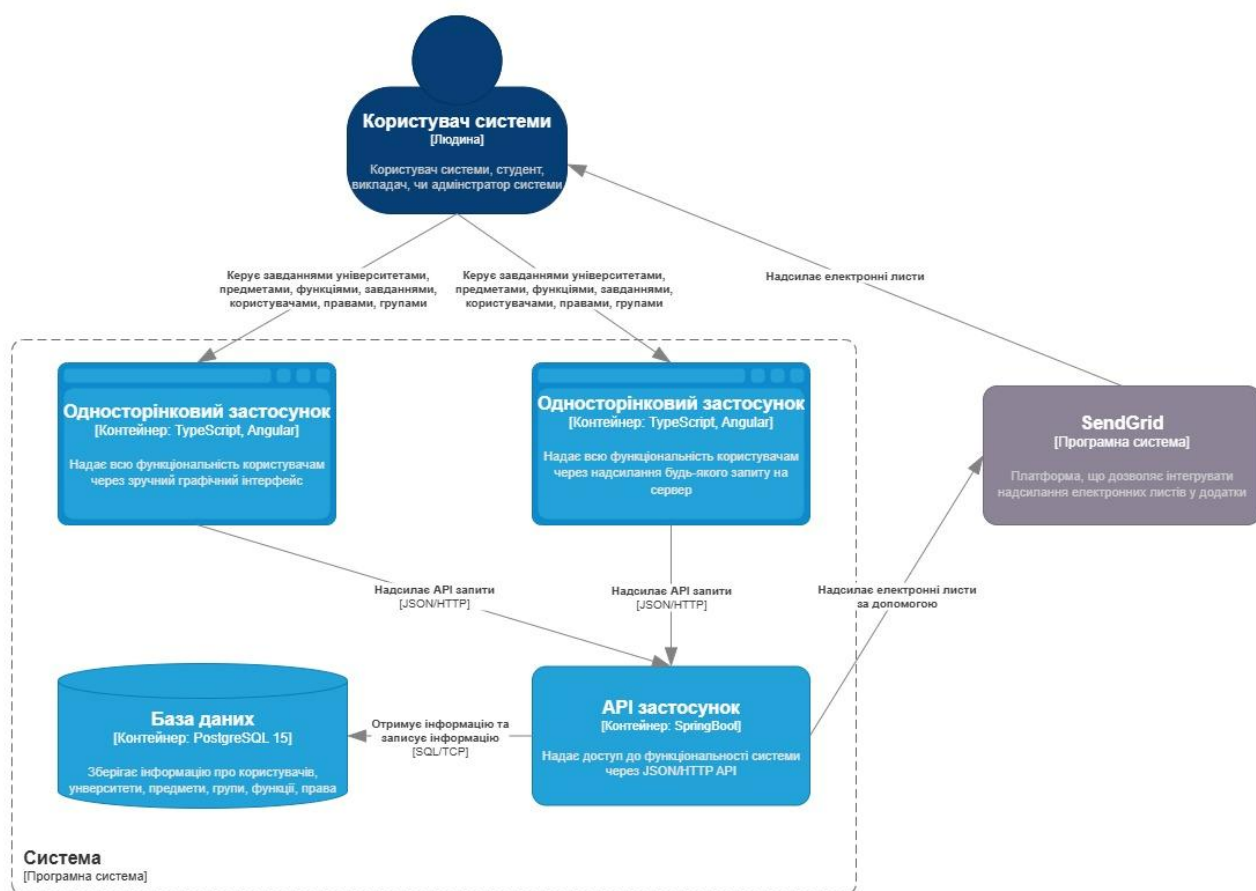


Рисунок 2.4 – Діаграма 2-ого рівня C4

Ще не було згадано SendGrid – платформу для надсилання листів. SendGrid є зовнішньою (вже існуючою) системою і не входить до 3-рівневої архітектури напряду. Звернення до SendGrid відбувається на рівні сервісів на сервері.

Також, варто розглянути систему авторизації, аутентифікації та реєстрації в розроблюваному додатку.

При реєстрації вибирається роль користувача – студент, викладач чи адміністратор, а також вводиться персональна інформація така як ім'я, прізвище та адреса електронної пошти. Звісно ж також користувач вводить пароль для подальшого входу в систему. При реєстрації, в базі даних зберігається користувач, його роль та персональна інформація. Пароль, введений користувачем, в базі даних зберігається у хешованому вигляді, для більшої безпеки.

В системі передбачено 2 види аутентифікації – через адресу електронної пошти, пароль та унікальний ключ користувача. Вхід у систему через пошту та пароль використовується для додатку, що надає графічний інтерфейс для взаємодії із системою. Вхід у систему через унікальний ключ використовується в додатку для надсилання запитів напряду на сервер. Унікальний ключ в свою чергу користувач може отримати лише увійшовши в обліковий запис в додатку з графічним інтерфейсом. Причому, кожного разу користувач буде отримувати новий унікальний ключ, для вищої надійності. Унікальний ключ зберігається у базі даних у хешованому вигляді.

Після аутентифікації, у випадку, якщо електронна пошта чи пароль неправильні, виводиться відповідне повідомлення користувачу. У разі успішної аутентифікації, користувачу відправляється 2 JWT-токени: 1-ий для можливості надсилання запитів на сервер, 2-ий – для оновлення 1-ого токена у випадку, якщо термін його дії закінчиться. JWT токен містить електронну пошту (для ідентифікації користувача), а також роль користувача. Цей токен необхідний для авторизації в додатку.

Авторизація в додатку, як було згадано вище, відбувається через використання JWT-токена. Ці токени повертаються як відповідь запиту на

аутентифікацію. Після їх отримання, вони зберігаються в фронтенд-додатках для подальшого використання при надсиланні запитів. При надсиланні запитів, токен, що використовується для авторизації, вставляється в заголовок запиту. На сервері цей токен перевіряється: чи існує електронна пошта, зашифрована в токені, в базі даних, а також, чи підписаний цей токен тим же ключом, який був наперед заданий на сервері. Це дозволяє перевірити чи є ключ тим же, що був виданий при аутентифікації і не був змінений. Якщо токен має неправильну сигнатуру, то користувача автоматично викидує із системи.

JWT-токен має термін дії. Якщо термін токена для авторизації запитів закінчився, то автоматично надсилається запит для його оновлення. В цьому процесі використовується токен оновлення, термін дії якого зазвичай більший ніж в токену для авторизації запитів. Токен для оновлення так само перевіряється на сервері, і у випадку, якщо він правильний, а також, якщо його термін дії не минув, то у відповідь повертається нова пара токенів, що зберігається у фронтенд-додатках і користувач продовжує працювати. Якщо термін токена для оновлення теж закінчився, то користувача автоматично викидує із системи.

Система авторизації, що використовує токени використовується в обидвох фронтенд-додатках, оскільки на сервері неможливо виконати запит без JWT-токена.

Варто розглянути архітектуру додатку у контексті нефункціональних вимог. Щоб забезпечити нефункціональні вимоги, було реалізовано наступні обмеження:

- всі складні операції будуть виконуватися на сервері. Це дозволить уникнути навантаження на фронтенд-додаток, який по свої суті не призначений для таких операцій;
- надсилати на клієнт лише дані, що там дійсно необхідні, без лишньої інформації. Це дозволить збільшити швидкість отримання відповіді на запит користувача;
- використовувати права доступу, ролі, хешування паролів, унікального ключа, а також системи авторизації через JWT-токени;

- використання асинхронних функцій на клієнтській стороні. Це дозволить уникнути незручностей для користувача при роботі із системою;
- використовувати сучасні бібліотеки стилів для створення user-friendly веб-сторінок.

2.4 Опис технологій для розробки програмного забезпечення

Технології, які були використані для розробки програмного забезпечення можна розділити на 4 частини – ті які використовувалися для написання серверу, клієнтських додатків, ті, що використовувалися для доступу до баз даних, та інші.

Для написання серверу була вибрана мова програмування Java, фреймворк для написання саме веб-додатку на Java – Spring. Для забезпечення доступу до даних використовувалася бібліотека Spring Data, а також ORM фреймворк Hibernate. Для забезпечення безпеки серверу використовувалася бібліотека Spring Security. Для спрощення написання бойлер-плейт коду (той що пишеться завжди, у всіх додатках, код «по замовчуванню») використовувалася бібліотека Lombok. Для роботи із JWT-токенами використовувалася бібліотека io.jsonwebtoken. Для забезпечення міграцій баз даних використовувався бібліотека Flyway. Для написання серверу з використанням Spring фреймворку використовувався IDE – IntelliJ IDEA Ultimate.

Серед популярних мов програмування можна виокремити 3, які найбільш пристосовані та найбільш використовуються для веб-розробки: C#, Python, Java. Всі вони однаково підходили б для розробки серверу, але саме мова Java була обрана, оскільки в мене є досвід розробки вже декількох додатків саме мовою Java.

Сама по собі мова Java не пристосована для написання веб-сервера. Для цього використовуються фреймворки. Найбільш відомі з них, написані саме для мови Java – це Spring, Java EE, Netty. Вибір на користь Spring був зроблений з ряду причин:

- гнучкість: забезпечення більшої гнучкості і модульності, що дозволяє розробникам обирати та використовувати лише необхідні компоненти;

- простота: багато розробників вважають Spring простішим у використанні, особливо для створення нових додатків;
- велика кількість бібліотек: існує дуже багато додаткових бібліотек, що спрощують розробку (Spring Security, Spring Data тощо) [25].

Для доступу до баз даних в фреймворці Spring зазвичай використовують бібліотеку Spring Data, так як вона є стандартом для цього в Spring. В цій бібліотеці є 2 підходи для доступу до баз даних: використання JDBC API (написання SQL-запитів вручну) або ORM фреймворку Hibernate (дозволяє взаємодіяти з базою даних у вигляді класів, методів, об'єктів). Оскільки в розроблюваній системі існує дуже багато сутностей, то було вирішено використовувати Hibernate. В додатку не буде жодних складних операцій з базою даних, тому JDBC API не дасть жодних переваг у швидкості через виконання SQL-запитів напряму в базі даних (Hibernate генерує SQL-запити автоматично на основі методів, що використовуються для об'єктів, що в деяких випадках могло зменшувати швидкодію виконання запитів).

Стандартом для забезпечення безпеки в Spring є бібліотека Spring Security. Вона дозволяє реалізувати перевірку JWT-токенів, а також перевірку чи має користувач необхідну роль для виконання запиту.

Для зменшення написання додаткового коду, який виконує стандартні операції (отримання значення поля класу, збереження значення поля класу, методи для порівняння об'єктів одного класу), було вирішено використати бібліотеку Lombok. Це також дає можливість залишати класи невеликого розміру, а також такими, що містять лише важливі поля та методи.

Бібліотека `io.jsonwebtoken` дозволяє генерувати JWT-токени та підписувати їх унікальним ключем для забезпечення безпеки.

Контроль міграцій необхідний для того, щоб контролювати, які запити будуть виконуватися при кожному запуску серверу. Це необхідно, щоб додаток стабільно працював при кожному запуску. Також, це дозволяє уникати ручного додавання даних в базу даних, що неможливо при розгортанні нової версії додатку на основному сервері. Для контролю міграції, в фреймворку Spring

використовуються в основному 2 бібліотеки: Flyway, Liquibase. Flyway була обрана через великий досвід її використання, а також відсутності необхідності в перевагах (відмінностях) Liquibase (можливість визначення структури бази через xml, json та інші формати, а також системи логування міграцій при паралельному програмуванні) [26].

В якості IDE, був використаний IntelliJ IDEA Ultimate. Вона має зручний графічний інтерфейс, інструменти для відладки серверу (дебаг), вбудований інструмент для перегляду бази даних, що дозволяє не використовувати спеціалізований додаток для доступу до БД. До того ж, ця IDE використовується багатьма розробниками для додатків на мові Java та фреймворку Spring.

Для розробки клієнтських додатків був використаний один стек технологій. В якості фронтенд-фреймворку був використаний Angular, і відповідно мова TypeScript. Для реалізації гарного інтерфейсу були використані бібліотеки Bootstrap та FontAwesome. IDE для написання фронтенд-додатків – Visual Studio Code.

В додатках необхідне асинхронне виконання методів. 2 найпопулярнішими фреймворками, що повноцінно дозволяють це реалізувати є React та Angular. Angular, та відповідно мова TypeScript, була вибрана виключно через досвід написання додатків використовуючи ці технології.

Для стилізації додатку були використані бібліотеки Bootstrap та FontAwesome. Bootstrap дозволяє реалізувати привабливі таблиці, поля для вводу та інші елементи сторінки. FontAwesome дозволяють додати маленькі зображення (іконки), що будуть підказувати користувачу функціонал інтерфейсу.

Щодо IDE, то Visual Studio Code була використана через її хороший інтерфейс, багатофункціональність, наявність спеціалізованих додаткових плагінів для Angular та TypeScript, а також, знову, через великий досвід використання цієї IDE.

Сервером баз даних є PostgreSQL. Вибір на користь саме цієї СУБД, а не MySQL, Microsoft SQL, SQLite чи навіть NoSQL баз даних був зроблений з таких

причин:

- надійність і стабільність: має довгу історію активної розробки та чудову репутацію за відмовостійкість і продуктивність, відповідаючи стандарту ACID;
- відкритий код і безкоштовність: повністю безкоштовна, доступна за ліцензією з відкритим кодом, що дозволяє вільно використовувати, змінювати та розповсюджувати її без витрат на ліцензування;
- масштабованість: добре підходить як для невеликих стартапів, так і для великих корпоративних систем, які вимагають високої продуктивності та роботи з великими обсягами даних [27].

Окремий застосунок для отримання доступу до баз даних не використовувався, оскільки цей функціонал вже існує в IDE IntelliJ IDEA Ultimate.

Технологією, що входить до категорії «інші» можна вважати Postman, що використовувався для перевірки виконання API-запитів. Цей додаток дозволяє ефективно перевіряти правильність роботи серверу завдяки потужному функціоналу та зручному інтерфейсу.

Висновки до розділу

В першому підрозділі – «Аналіз функціональних вимог до програмного забезпечення» – були сформовані та описані функціональні вимоги до системи, які дозволять повноцінно реалізувати завдання, що були поставлені в рамках роботи.

В другому підрозділі – «Аналіз нефункціональних вимог до програмного забезпечення» – були написані нефункціональні вимоги до системи, що дадуть змогу зручно користуватися системою.

В третьому підрозділі – «Опис архітектури програмного забезпечення» – була розписана як загальна архітектура додатку, так і архітектура кожного з його елементів.

В четвертому підрозділі – «Опис технологій для розробки програмного забезпечення» – були показані технології, що дозволять створити додаток, що

реалізовує поставлені функціональні та нефункціональні вимоги.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Опис схеми баз даних

В межах розробки програмного забезпечення передбачені сутності, які описані в таблиці 3.1.

Таблиця 3.1 – Сутності бази даних

Назва	Опис
Університет	Має назву та містить в собі багато предметів
Предмет	Має назву та містить в собі багато функцій
Функція	Містить текст функції, кількість змінних, мінімальні та максимальні значення
Завдання користувача	Містить завдання та функцію, які буде виконувати користувач
Завдання	Містить підказку до завдання та тип завдання: знайти мінімальне чи максимальне значення
Користувач	Має ім'я, прізвище, електронну адресу та унікальний ключ
Права користувача	Відноситься до університету, предмету, функції, на дії з якими у користувача буде доступ
Роль користувача	Є 3 ролі користувачів: студент, викладач та адміністратор
Група	Має назву та відноситься до предметів, що вивчаються в групі та студентів, що входять до групи
Запит	Містить текст, пояснення, метод запиту, що може виконувати користувач
Поле запиту	Містить тип, назву та пояснення поля, що відповідає певному запиту

Продовження таблиці 3.1

Назва	Опис
Оцінка	Містить оцінку та коментар щодо виконання завдання
Відповідь на завдання	Містить відповідь, що надав користувач, а також її порядок
Обмеження на завдання	Містить інформацію про обмеження завдання: кількість спроб, дедлайн чи обмеження часу для надання 1-єї відповіді

Однак в базі даних сутності зберігаються по-різному або в різних таблицях в залежності від необхідних зв'язків між сутностями. Перелік та опис всіх таблиць в базі даних наведені в таблиці 3.2.

Таблиця 3.2 – Перелік таблиць бази даних

Назва	Опис
answers	Містить інформацію про відповіді надані для завдання студента
assignments	Містить інформацію про завдання, що можуть бути присвоєні студенту
default_assignment_restrictions	Містить інформацію про обмеження завдань по замовчуванню
fields	Містить інформацію про поля, що будуть використовуватися при виконанні запитів
flyway_schema_history	Містить інформацію про виконані міграції
functions	Містить інформацію про функції та до яких предметів вони відносяться
groups	Містить інформацію про групи

Продовження таблиці 3.2

Назва	Опис
group_students	Містить інформацію про те, які студенти входять до якої групи
group_subjects	Містить інформацію про те, які предмети вивчаються в групі
marks	Містить інформацію про оцінки виставлені викладачем за завдання студента
roles	Містить інформацію про те, які ролі існують в системі
subjects	Містить інформацію про предмети та університет, до якого предмети відносяться
universities	Містить інформацію про університет
url_fields	Містить інформацію про те які поля відносяться до яких запитів
urls	Містить інформацію про запити, які може виконувати користувач
user_assignments	Містить інформацію про те, які функції та завдання, має користувач, а також обмеження на конкретне завдання
user_info	Містить персональну інформацію про користувача: ім'я, прізвище та унікальний ключ
user_permissions	Містить інформацію про права користувача на виконання дій в межах університету, предмету чи функції
user_roles	Містить інформацію про те, які ролі має який користувач
users	Містить інформацію про користувачів: електронну адресу, пароль, токен для оновлення

Всі таблиці містять первинний ключ, який дозволяє ідентифікувати записи. Всі зв'язки між полями різних таблиць реалізовані у вигляді зовнішніх ключів. Обмежень на поля таблиць немає, окрім необхідності наявності первинного ключа в кожному записі таблиці. Більш детально, поля та зв'язки між полями деяких таблиць бази даних зображені на рисунку 3.1 у вигляді ER-діаграми.

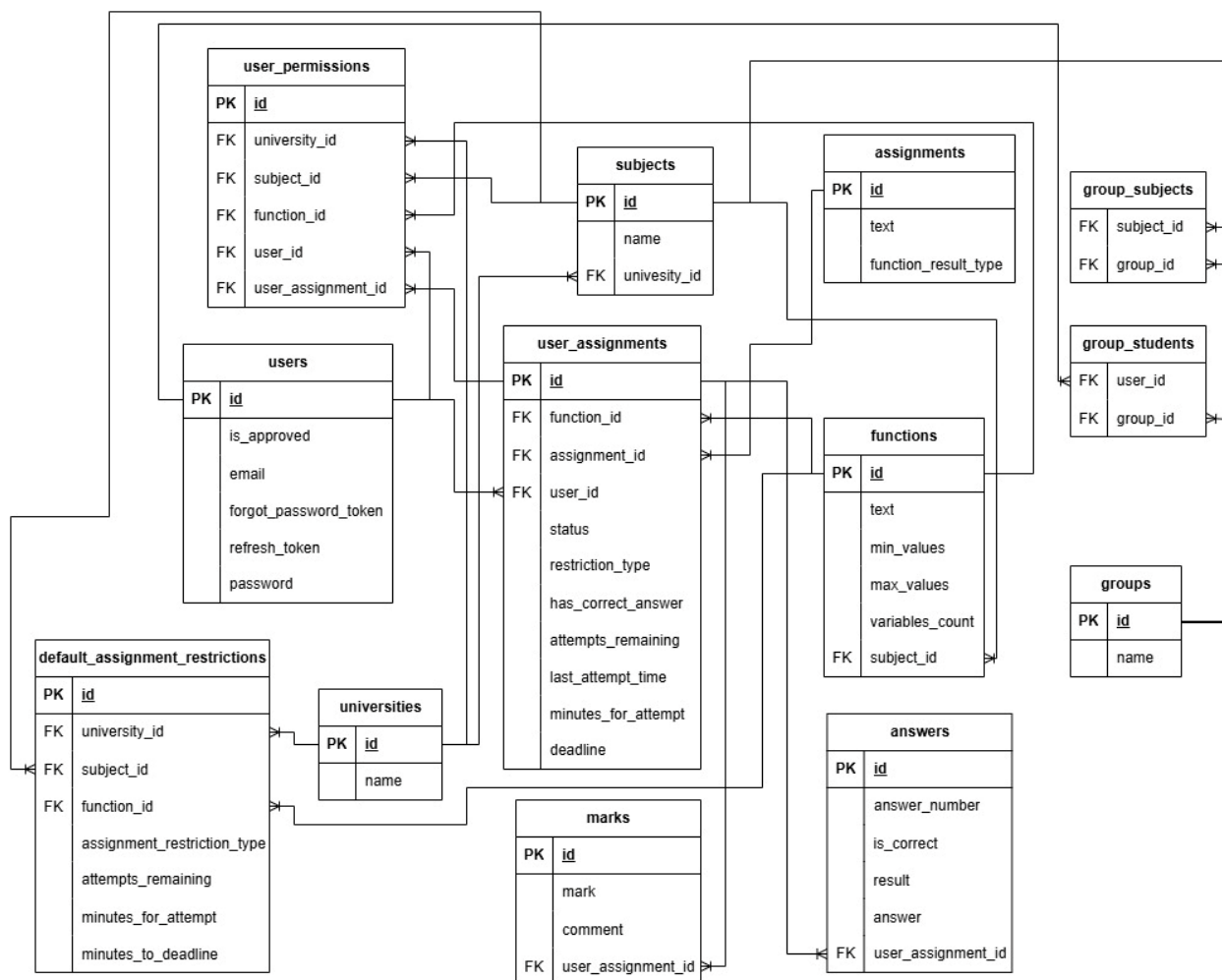


Рисунок 3.1 – ER-діаграма основних сутностей бази даних

ER-діаграма всіх сутностей бази даних застосунку знаходиться в додатку

А.

3.2 Опис структури програмного продукту

Для повного опису структури додатку необхідно: Спочатку

– описати всі сутності верхнього рівня на стороні сервера: контролери,

сервіси та репозиторії;

- розписати основну структуру клієнтських застосунків;
- продемонструвати максимально детально як проходить процес присвоєння завдання – одної із основних функцій застосунку.

Опис структури програмного продукту необхідно розпочати з контролерів (таблиця 3.3) на сервері.

Таблиця 3.3 – Контролери на стороні сервера

Назва	Призначення
AuthController	Приймає запити на реєстрацію, вхід через електронну пошту, через ключ, підтвердження користувача, відхилення користувача, зміна паролю, оновлення токenu
Assignment Controller	Приймає запити на отримання завдань певного користувача, а також конкретного завдання, присвоєння завдання випадковим чином певному студенту, а також певній групі студентів, завершення завдання, надання відповіді і отримання відповідей на певне завдання
Assignment RestrictionController	Приймає запити на отримання обмежень завдань по замовчуванню, отримання типів обмежень, задання обмежень по замовчуванню, видалення обмеження завдань по замовчуванню, задання обмеження завдань для поточних завдань
FieldController	Приймає запит на отримання полів для конкретного посилання
FunctionController	Приймає запити на отримання функцій, створення, редагування, видалення функцій
GroupController	Приймає запити на отримання груп, створення, редагування, видалення груп;

Продовження таблиці 3.3

Назва	Призначення
MarkController	Приймає запити на отримання оцінок для конкретного завдання, виставлення оцінки
PermissionController	Отримання дозволів користувача, задання прав користувачу, позбавлення прав користувача
SubjectController	Приймає запити на отримання предметів, створення, редагування, видалення предметів
UniversityController	Приймає запити на отримання університетів, створення, редагування, видалення університетів
UrlController	Приймає запити на отримання запитів, отримання методів запитів
UserController	Приймає запит на отримання інформації про користувача, отримання ключа користувача

Існуючі сервіси мають ті ж назви та реалізують методи, які надають функціонал, для яких контролери отримують запит: AuthService, FieldService, FunctionService, MarkService, PermissionService, SubjectService, UniversityService, UrlService, AssignmentService, AssignmentRestrictionService, UserService. Єдині сервіси, які мають додаткові методи описані в таблиці 3.4.

Таблиця 3.4 – Опис сервісів на стороні серверу, що містять додаткові функції

Назва сервісу	Опис
AssignmentRestrictionService	Містить додаткові методи для визначення обмежень завдань для різних сутностей (університет, предмет, функція)

Продовження таблиці 3.4

Назва сервісу	Опис
PermissionService	Містить додаткові методи для визначення прав користувачів на виконання певних дій для різних сутностей (університет, предмет, функція, завдання користувачів)
SessionService	містяться методи для отримання користувача, що зараз надсилає запит.

Репозиторії називаються відповідно до сутностей, до яких надається доступ через ці репозиторії: AnswerRepository, AssignmentRepository, DefaultAssignmentRestrictionRepository, FieldRepository, FunctionRepository, GroupRepository, MarkRepository, RoleRepository, SubjectRepository, UniversityRepository, UrlFieldRepository, UrlRepository, UserAssignmentRepository, UserInfoRepository, UserPermissionRepository, UserRepository.

Клієнтський застосунок розділений на 3 окремі найбільші частини: модулі, компоненти та сервіси. Модулі – це частина проекту, що об’єднана якоюсь сутністю, концепцією тощо. Також в модулі задаються обмеження для входу на сторінку та шляхи до компонентів модуля. Компонент – це частина модуля, яка реалізовує частину функціоналу модуля. Сервіс – це елемент структури, який надсилає запит на контролер на сервері.

Модулі та компоненти в клієнтському застосунку, що реалізовує зручний графічний інтерфейс для взаємодії із основними сутностями описані в таблиці 3.5.

Таблиця 3.5 – Модулі графічного додатку

Назва	Призначення
AuthModule	Містить компоненти, що показують сторінки реєстрації, входу через електронну пошту, зміну пароллю
AssignmentModule	Містить компонент, що показує сторінку із завданнями доступними користувачу та самого виконання завдання;
GroupModule	Містить компоненти, що показують сторінки з групами доступними користувачу, сторінки створення, редагування груп
UniversityModule	Містить компоненти, що показують сторінки з університетами доступними користувачу, сторінки створення, редагування університетів
SubjectModule	Містить компоненти, що показують сторінки з предметами доступними користувачу, сторінки створення, редагування предметів
FunctionModule	Містить компоненти, що показують сторінки з функціями доступними користувачу, сторінки створення, редагування функцій
UserModule	Містить компоненти, що показують сторінки з користувачами доступними адміністратору, сторінки редагування користувачів (їхніх прав), сторінку з отриманням ключа користувача
ErrorModule	Містить компоненти, що відображають сторінки з помилками

Сервіси, що знаходяться в цьому клієнтському застосунку схожі на сервіси реалізовані на сервері за винятком декількох сервісів, які описані в таблиці 3.6.

Таблиця 3.6 – Опис сервісів на стороні клієнту, що містять додаткові функції

Назва сервісу	Опис
AuthService	Додатково містить метод logout() який виконує вихід із облікового запису користувача
JWTTokenService	Дозволяє отримувати інформацію закодовану в токени, оновлювати поточний токен користувача тощо;
LocalStorageService	Дозволяє зберігати та отримувати токен користувача та токен для оновлення в локальному сховищі сайту

Клієнтський застосунок, що дозволяє надсилати будь-які запити на сервер містить в разі менше модулів, компонентів та сервісів, ці модулі розписані в таблиці 3.7.

Таблиця 3.7 – Модулі графічного додатку для надсилання запитів

Назва	Призначення
AuthModule	Містить компонент з сторінкою входу через ключ
ToolModule	Містить компонент з сторінкою для надсилання запитів на сервер
ErrorModule	Містить компонент з сторінкою, на якій відображається помилка для користувача

В цьому фронтенд-застосунку також містяться JWTTokenService, LocalStorageService, що виконують ті ж функції, що і в першому клієнтському застосунку. Інші сервіси, що унікальні для цього застосунку описані в таблиці 3.8.

Таблиця 3.8 – Опис сервісів на стороні другого клієнтського застосунку, що містять додаткові функції

Назва сервісу	Опис
AuthService	Містить метод для авторизації користувача за унікальним ключем
ApiService	Містить метод для надсилання запитів на вказану адресу, із відповідними параметрами запиту чи тілом запиту
UrlService	Містить запити на отримання всіх запитів, запита по тексту запита, отримання методів запитів
FieldService	Містить запит для отримання полів для введеного запиту

Тепер необхідно розглянути процес взаємодії серверу та клієнтських застосунків на прикладі присвоєння випадкового завдання користувачу. Опис процесу розпочинається із моменту натиснення кнопки на отримання завдання (рисунок 3.2).

```
<button class="btn btn-primary" (click)="assign()">
  <i class="fa-solid fa-plus me-2"></i> Отримати завдання
</button>
```

Рисунок 3.2 – Фрагмент коду, що додає кнопку для надсилання запиту на сервер на сторінку

При натисненні кнопки, виконується метод `assign()` в компоненті (рисунок 3.3), в якому з модального вікна повертаються ідентифікатори предметів (з яких буде братися завдання), які використовуються при виклику методу `assign()` сервісу `AssignmentService` (рисунок 3.4).

```

modalRef.componentInstance.saveAttempt.subscribe(
  (value: number[]) => {
    let dto = new AssignDto(value);
    this.assignmentService.assign(dto).subscribe({
      error: (error: any) => {
        errorSubject.next(error.error.message);
      },
      complete: () => {
        modalRef.close();
        this.getAssignmentsByUserId();
      }
    })
  })
);

```

Рисунок 3.3 – Фрагмент коду з методом assign() компоненту
AssignmentComponent

```

assign(assignDto: AssignDto): Observable<any> {
  return this.httpClient.post(`${this.url}/assign`, assignDto);
}

```

Рисунок 3.4 – Фрагмент коду із методом assign() сервісу AssignmentService

Цей метод надсилає запит на сервер. Але перед цим, запит перехоплюється так званим перехоплювачем, цей перехоплювач додає токен користувача в хедер вихідного запиту, а також перевіряє відповідь на запит (рисунок 3.5). Якщо цей запит повертає 401 код статусу, то відбувається запит на оновлення токenu користувача використовуючи AuthService та його метод refreshToken() (рисунок 3.6). Якщо сервер знову повертає 401 код статусу, то це означає, що термін дії токenu для оновлення завершився. В такому разі, користувача розлогінює із системи, щоб користувач знову увійшов у систему. Якщо токен успішно оновився, то він зберігається у користувача та виконуємо запит, який користувач хотів виконати спочатку.

```

const setHeader = (request: HttpRequest<any>) => {
  const token = jwtService.getToken();
  return token ? request.clone({ setHeaders: { Authorization: 'Bearer ' + token } }) : request;
};

const handle401Error = (request: HttpRequest<any>, nextFn: HttpHandlerFn): Observable<any> => {
  if (isRefreshingToken) {
    return tokenRefreshed$.pipe(
      filter(Boolean),
      take(1),
      concatMap(() => nextFn(setHeader(request)))
    );
  }

  isRefreshingToken = true;
  tokenRefreshed$.next(false);

  return authService.refreshToken(new RefreshTokenDto(jwtService.getRefreshToken())).pipe(
    concatMap((res: TokensDto) => {
      jwtService.setToken(res.accessToken);
      jwtService.setRefreshToken(res.refreshToken);
      tokenRefreshed$.next(true);
      return nextFn(setHeader(request));
    }),
    catchError(err => {
      authService.logout();
      router.navigateByUrl('/auth/signin');
      return throwError(() => err);
    }),
    finalize(() => {
      isRefreshingToken = false;
    })
  );
};

```

Рисунок 3.5 – Фрагмент коду із перехоплювачем на стороні клієнта

```

refreshToken(refreshTokenDto: RefreshTokenDto): Observable<TokensDto> {
  return this.httpClient.put<TokensDto>(`${this.url}/refreshToken`, refreshTokenDto);
}

```

Рисунок 3.6 – Фрагмент коду із запитом на оновлення токєну

На сервері, цей запит перехоплюється схожим перехоплювачем. Цей перехоплювач, по-перше, перевіряє чи правильний токен знаходиться в хедерах запиту, та задає поточного користувача в сесії (рисунок 3.7), а також перевіряє чи роль користувача збігається з тою, що була присвоєна цьому запиту (рисунок 3.8).

```

@Override
protected void doFilterInternal(HttpServletRequest request,
                                HttpServletResponse response,
                                FilterChain filterChain) throws ServletException, IOException {
    String jwt = parseJwt(request);
    if (jwt != null && jwtUtils.validateAccessToken(jwt)) {
        String email = jwtUtils.getEmailFromAccessToken(jwt);

        UserDetails userDetails = userDetailsService.loadUserByUsername(email);
        UsernamePasswordAuthenticationToken authentication =
            new UsernamePasswordAuthenticationToken(
                userDetails,
                null,
                userDetails.getAuthorities());
        authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

        SecurityContextHolder.getContext().setAuthentication(authentication);
    }

    filterChain.doFilter(request, response);
}

```

Рисунок 3.7 – Фрагмент коду, що перевіряє токен та зберігає користувача в сесії

```

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.cors(Customizer.withDefaults()).
        csrf(AbstractHttpConfigurer::disable)
        .exceptionHandling(exception ->
            exception.authenticationEntryPoint(unauthorizedHandler))
        .sessionManagement(session ->
            session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authorizeHttpRequests(auth ->
            auth.requestMatchers(studentUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(teacherUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(adminUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(studentTeacherUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(adminTeacherUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(userUrls).hasAnyRole(UserRole.ADMIN.name(),
                UserRole.TEACHER.name(), UserRole.STUDENT.name())
            .requestMatchers(publicUrls).permitAll()
            .requestMatchers(authUrl).permitAll()
            .anyRequest().authenticated()
        )
        .authenticationManager(authenticationManager(http, userDetailsService))
        .addFilterBefore(authenticationJwtTokenFilter(),
            UsernamePasswordAuthenticationFilter.class);

    return http.build();
}

```

Рисунок 3.8 – Фрагмент коду, що перевіряє роль користувача та задає перехоплювач

Якщо одна з перевірок не проходиться, то надсилається повідомлення про

помилку із статусом 401, що означає, що користувач не авторизований для виконання цієї дії.

Якщо проходяться всі перевірки на попередньому етапі, то запит отримується в методі контролера AssignmentController (рисунок 3.9).

```
@PostMapping("/assign")
public ResponseEntity<?> assign(@RequestBody AssignDto assignDto) {
    assignmentService.assign(assignDto);
    return ResponseEntity.ok().build();
}
```

Рисунок 3.9 – Фрагмент коду із методом в контролері AssignmentController

В цьому методі викликається метод assign(), що належить AssignmentService (рисунки 3.10 та 3.11).

```
for (User user : users) {
    List<UserAssignment> possibleUserAssignments = new ArrayList<>();

    for (Function function : allFunctions) {
        for (Assignment assignment : allAssignments) {
            if (assignment.getFunctionResultType().equals(FunctionResultType.MIN) &&
                !getMinMaxValuesForFunction(function, FunctionResultType.MIN).isEmpty())
                possibleUserAssignments.add(new UserAssignment(user, function, assignment));
            else if (assignment.getFunctionResultType().equals(FunctionResultType.MAX) &&
                !getMinMaxValuesForFunction(function, FunctionResultType.MAX).isEmpty()) {
                possibleUserAssignments.add(new UserAssignment(user, function, assignment));
            }
        }
    }

    if (possibleUserAssignments.isEmpty())
        throw new NoAvailableAssignmentsException();

    List<UserAssignment> userAssignments = user.getUserAssignments();
    possibleUserAssignments.removeAll(userAssignments);

    Random random = new Random();
    int possibleFunctionsCount = possibleUserAssignments.size();
    int possibleFunctionIndex = random.nextInt(possibleFunctionsCount);

    UserAssignment userAssignment = new UserAssignment();
    userAssignment.setAssignment(possibleUserAssignments.get(possibleFunctionIndex).getAssignment());
    userAssignment.setUser(user);
    userAssignment.setFunction(possibleUserAssignments.get(possibleFunctionIndex).getFunction());
    userAssignment.setHasCorrectAnswer(false);
    userAssignment.setStatus(AssignmentStatus.ASSIGNED);
}
```

Рисунок 3.10 – Фрагмент коду із методом assign()

```

DefaultAssignmentRestriction defaultAssignmentRestriction =
    assignmentRestrictionService.getDefaultRestrictionForFunction(userAssignment.getFunction());

if (defaultAssignmentRestriction == null) {
    defaultAssignmentRestriction = assignmentRestrictionService.getDefaultRestriction();
}

if (defaultAssignmentRestriction.getAssignmentRestrictionType()
    .equals(AssignmentRestrictionType.N_ATTEMPTS)) {
    userAssignment.setRestrictionType(AssignmentRestrictionType.N_ATTEMPTS);
    userAssignment.setAttemptsRemaining(defaultAssignmentRestriction.getAttemptsRemaining());
} else if (defaultAssignmentRestriction.getAssignmentRestrictionType()
    .equals(AssignmentRestrictionType.DEADLINE)) {
    userAssignment.setRestrictionType(AssignmentRestrictionType.DEADLINE);
    userAssignment.setDeadline(defaultAssignmentRestriction.getDeadline());
} else {
    userAssignment.setRestrictionType(AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES);
    userAssignment.setMinutesForAttempt(defaultAssignmentRestriction.getMinutesForAttempt());
}

allUserAssignments.add(userAssignment);

```

Рисунок 3.11 – Фрагмент коду із методом assign() (продовження)

В цьому методі визначається завдання, яку буде присвоєне користувачу, причому це завдання не може бути присвоєно вдруге. Також із іншого сервісу – AssignmentRestrictionService – отримується обмеження по замовчуванню для поточної конфігурації завдання: спочатку перевіряємо обмеження на функцію, потім на предмет, потім на університет. Те, яке обмеження буде знайдено першим, буде застосовано до завдання. Далі, присвоєне завдання користувача зберігається в БД через UserAssignmentRepository.

Варто загострити увагу на двох моментах.

Перший – це вирішення виключних ситуацій (exception). Якщо зустрічається якась виключна ситуацію, то застосовується оператор throw для створеного спеціального виключення (рисунок 3.12), а це виключення перехоплюється в спеціалізованому методі в RestControllerAdvice (рисунок 3.13).

```

public class NoAvailableAssignmentsException extends RuntimeException {
    public NoAvailableAssignmentsException() {
        super("Немає доступних завдань. Спробуйте пізніше!");
    }
}

```

Рисунок 3.12 – Фрагмент коду із спеціалізованим виключенням

```

@ExceptionHandler(value = NoAvailableAssignmentsException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
public BaseExceptionDto handleAssignmentException(NoAvailableAssignmentsException ex, WebRequest request) {
    return new BaseExceptionDto(
        HttpStatus.BAD_REQUEST.value(),
        ex.getMessage());
}

```

Рисунок 3.13 – Фрагмент коду із методом, що перехоплює виключення

Другий момент – це збереження створеної сутності в репозиторії AssignmentRepository. Цей репозиторій не містить сам по собі методи для роботи із сутностями, він лише успадковує методи із інтерфейсу JpaRepository, який отримує клас сутності та клас ідентифікатору сутності (рисунок 3.14).

```

@Repository
public interface AssignmentRepository extends JpaRepository<Assignment, Integer> {
}

```

Рисунок 3.14 – Фрагмент коду AssignmentRepository

Тепер необхідно описати цей же процес, але із використанням клієнтського застосунку для надсилання запитів. Після натискання кнопки для відправки запиту (рисунок 3.15), викликається метод sendRequest() (рисунок 3.16) в якому відбувається валідація всіх полів запиту та поля для введення тексту запиту і методу запиту, формується об'єкт з параметрами чи тілом запиту і викликається метод sendRequest() сервісу ApiService (рисунок 3.17).

```

<button class="btn btn-primary" (click)="sendRequest()">Надіслати</button>

```

Рисунок 3.15 – Фрагмент коду, що додає кнопку для відправки запиту на сторінку

```

sendRequest() {
  this.validateForm(this.urlForm);

  if (this.urlForm.invalid) {
    return;
  }

  if (this.fields.length > 0) {
    this.validateForm(this.fieldsForm);

    if (this.fieldsForm.invalid) {
      const firstInvalidIndex = this.fieldsFormControls.controls.findIndex(c => c.invalid);
      this.scrollToTarget(firstInvalidIndex);
      return;
    }
  }

  let queryOrBody: Record<string, any[]> = {};
  for (let i = 0; i < this.fields.length; i++) {
    if (this.fields[i].multiple === true) {
      Object.assign(queryOrBody, {
        [this.fields[i].name]: [...(queryOrBody[this.fields[i].name] || []),
          this.fieldsFormControls.at(i).value]
      })
    } else {
      Object.assign(queryOrBody, { [this.fields[i].name]: this.fieldsFormControls.at(i).value })
    }
  }

  this.showResponse = true;
  this.apiService.sendRequest(this.urlForm.value.url, this.urlForm.value.method, queryOrBody).subscribe({
    next: (resp: HttpResponse<any>) => {
      this.responseStatus = resp.status;
      if (resp.body)
        this.response = resp.body;
      else
        this.response = null;
    },
    error: (err) => {
      this.responseStatus = err.status;
      if (err.error)
        this.response = err.error;
    }
  })
}

```

Рисунок 3.16 – Фрагмент коду із методом компонента, що відправляє запит

```

sendRequest(url: string, method: MethodType, paramsOrBody: any): Observable<any> {
  if (method == MethodType.GET || method == MethodType.DELETE) {
    let params = new HttpParams({ fromObject: paramsOrBody });
    return this.httpClient.request(mapMethodTypeToLabel(method), `${this.url}${url}`, {
      params: params, observe: 'response' });
  } else {
    return this.httpClient.request(mapMethodTypeToLabel(method), `${this.url}${url}`, {
      body: paramsOrBody,
      observe: 'response'
    });
  }
}

```

Рисунок 3.17 – Фрагмент коду із методом, який відправляє запит на сервер

В цьому методі, визначається чи є об'єкт саме параметром запиту чи тілом (в залежності від методу) і надсилається запит по введеному запиту та методу. На стороні сервера, цей запит буде оброблюватися так же само, як і запит із клієнтського застосунку із графічним інтерфейсом.

3.3 Опис інтерфейсу програмного забезпечення

Як вже було зазначено в пункті 2.4, для створення інтерфейсу

використовувалися бібліотеки Bootstrap та FontAwesome, що забезпечували стилізацію контенту та додавання іконок.

Перш за все, для зручного переміщення між сторінками в будь-який момент, був зроблений так званий сайдбар – елемент, що завжди знаходиться зліва на сторінці та містить посилання на всі сторінки, що доступні користувачам.

На сторінці реєстрації містяться поля для вводу імені, прізвища, вибору ролі, введення адреси електронної пошти, паролю та підтвердження паролю. Якщо одне з полів не заповнене, електронна адреса чи пароль мають неправильний формат, пароль та підтвердження паролю, то виводяться відповідні повідомлення. Для реєстрації необхідно натиснути кнопку «Зареєструватися». Якщо користувач із введеною поштою вже існує, то виводиться відповідне повідомлення (рисунок 3.18).

The screenshot displays a web application interface with a dark blue sidebar on the left containing icons for a menu, home, user profile, and settings. The main content area is light blue and features a registration form titled "Регістрація" with a user icon. The form includes the following elements:

- Fields for "Ім'я" (Name), "Прізвище" (Surname), and "Роль" (Role) with a dropdown arrow.
- A red error message below the name field: "Ім'я є обов'язковим".
- An "Електронна пошта" (Email) field containing the text "amerscan@".
- A red error message below the email field: "Невірний формат електронної пошти".
- A "Пароль" (Password) field.
- A "Підтвердіть пароль" (Confirm password) field.
- A "← Назад" (Back) button at the bottom left.
- A blue "Зареєструватися" (Register) button at the bottom right with a user icon.

Рисунок 3.18 – Сторінка реєстрації

На сторінці входу, як зазвичай, знаходяться поля для вводу електронної адреси та паролю. Якщо електронну адресу чи пароль не введено, то виводиться відповідне повідомлення. Для входу потрібно натиснути кнопку «Увійти» (рисунок 3.19).

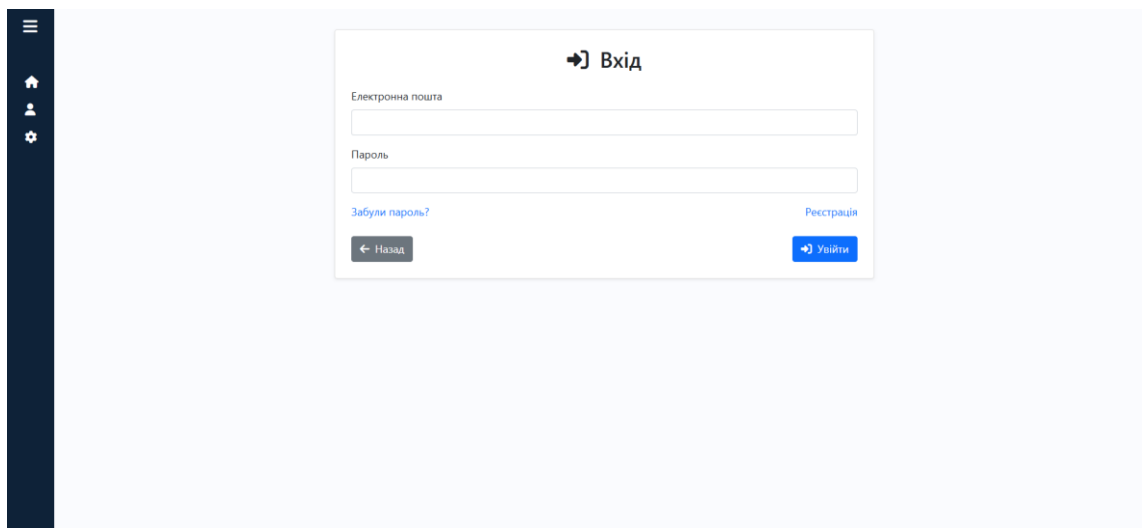


Рисунок 3.19 – Сторінка входу

Після натиснення кнопки «Увійти», користувача переносить на сторінку із завданнями, де міститься інформація про завдання, його оцінку та коментар (якщо оцінка вже була виставлена) (рисунок 3.20). Для викладача, також відображається кнопка для переходу на сторінку редагування обмеження завдання. Для студента відображається кнопка для присвоєння випадкового завдання для себе. Після натиснення кнопки отримання завдання відповідне завдання з'являється на екрані у списку активних завдань. Для початку виконання завдання натискається кнопку старту. Для перегляду лише введених відповідей натискається кнопку для перегляду.

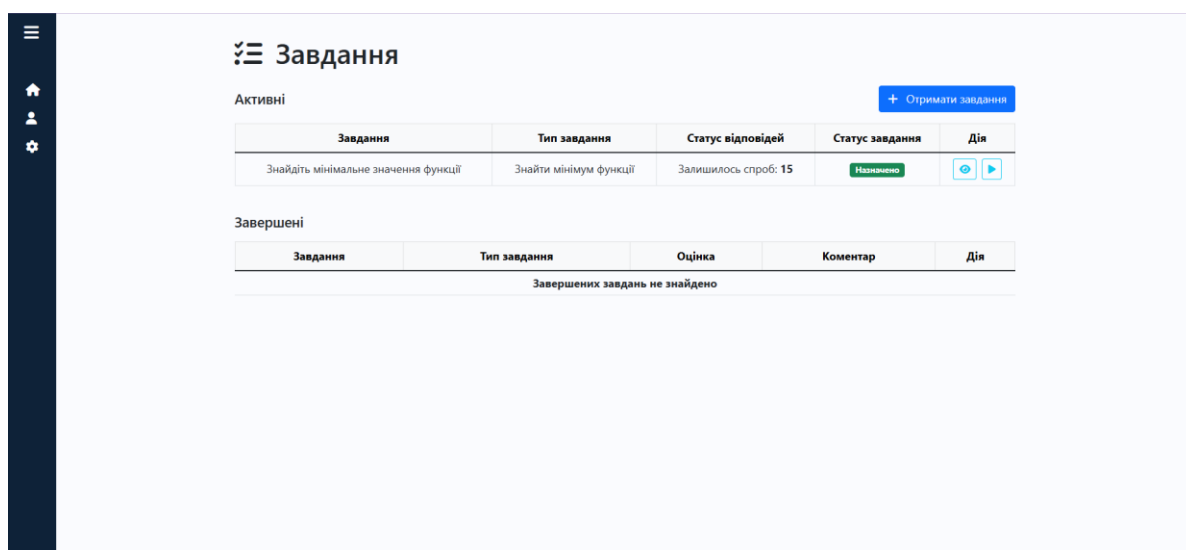


Рисунок 3.20 – Сторінка із завданнями

На сторінці із завданням показується невелика підказка до виконання завдання, інформація про обмеження завдання (якщо це кількість спроб, то показується кількість спроб, що залишилася, якщо це дедлайн, то показується час дедлайну, якщо це кількість часу для введення 1 відповіді, то показується час, коли можливе введення наступної відповіді), поля для введення значення змінної функції, кнопки для відправлення відповіді та завершення виконання завдання, а також введені відповіді (рисунки 3.21). Якщо відповідь неправильна, то показується відповідне повідомлення. Якщо значення змінної не введене, або введене значення не є числом, то виводиться відповідне повідомлення. Якщо кількість спроб завершилася, то кнопка для надання відповіді блокується, залишаючи можливість лише для завершення виконання завдання. Якщо обмеження завдання – це дедлайн, то після натиснення кнопки надання відповіді, покажеться відповідне повідомлення, кнопка для надання відповіді заблокується, а відповідь не зарахується. Якщо обмеження завдання – це кількість часу на 1 спробу, то після надання відповіді, кнопка для надсилання нової відповіді заблокується, доки користувач знову не зможе надати відповідь (рисунки 3.22).

Рисунок 3.21 – Сторінка для виконання завдання

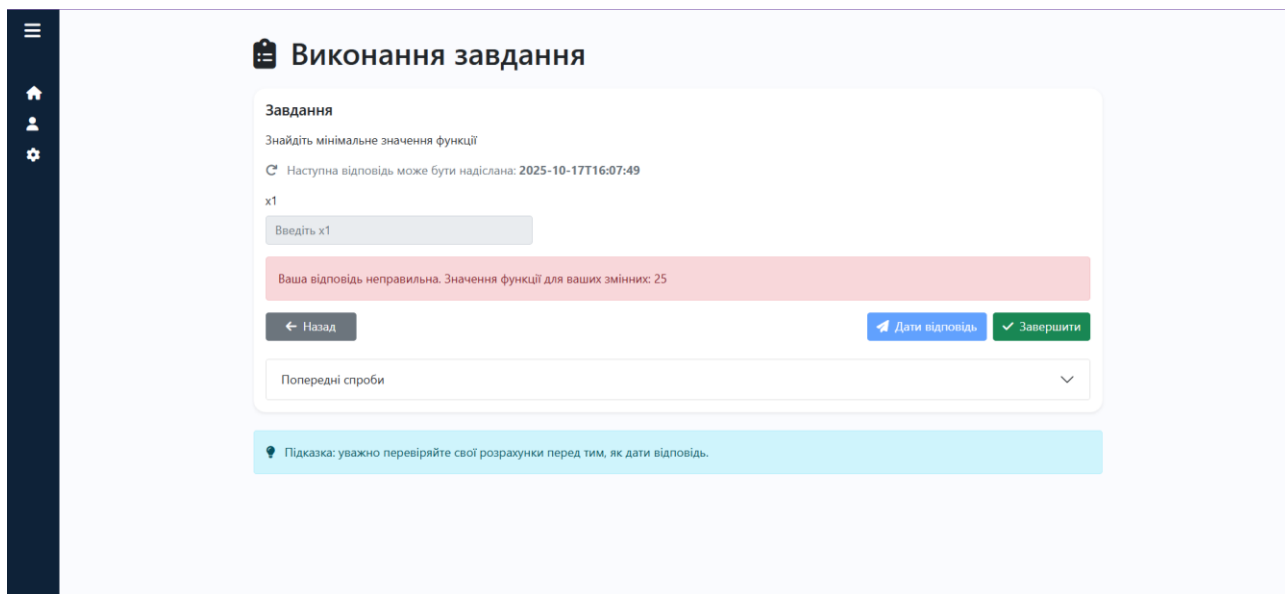


Рисунок 3.22 – Виконання завдання із обмеженням на кількість часу для 1 спроби

Необхідно також оглянути інтерфейс для надсилання запитів напряму до сервера без графічного інтерфейсу, що був показаний вище. Для доступу до цієї сторінки, необхідний ключ, який можна отримати в цьому застосунку з графічним інтерфейсом.

На цій сторінці буде відображатися лише кнопка для отримання ключа та відповідно сам ключ (рисунок 3.23).

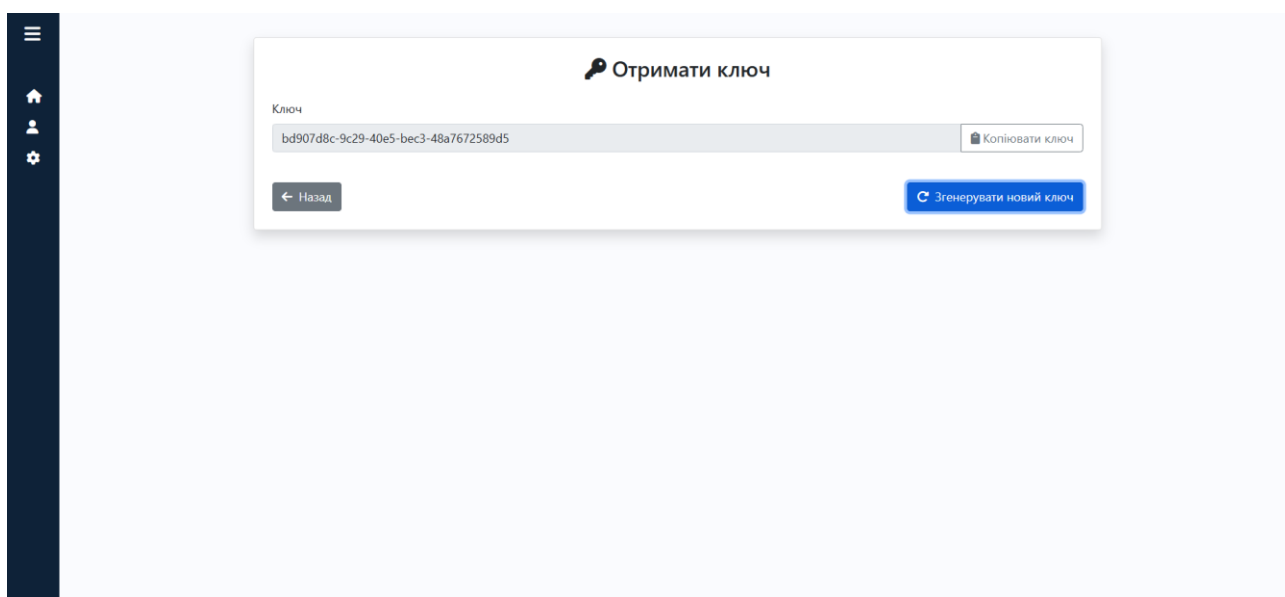


Рисунок 3.23 – Сторінка з ключем

На сторінці для входу на сторінку для надсилання запитів буде форма для входу, що містить поле для вводу ключа (рисунок 3.24). Для входу потрібно натиснути кнопку «Увійти». Якщо ключ неправильний, то виводиться відповідне повідомлення.

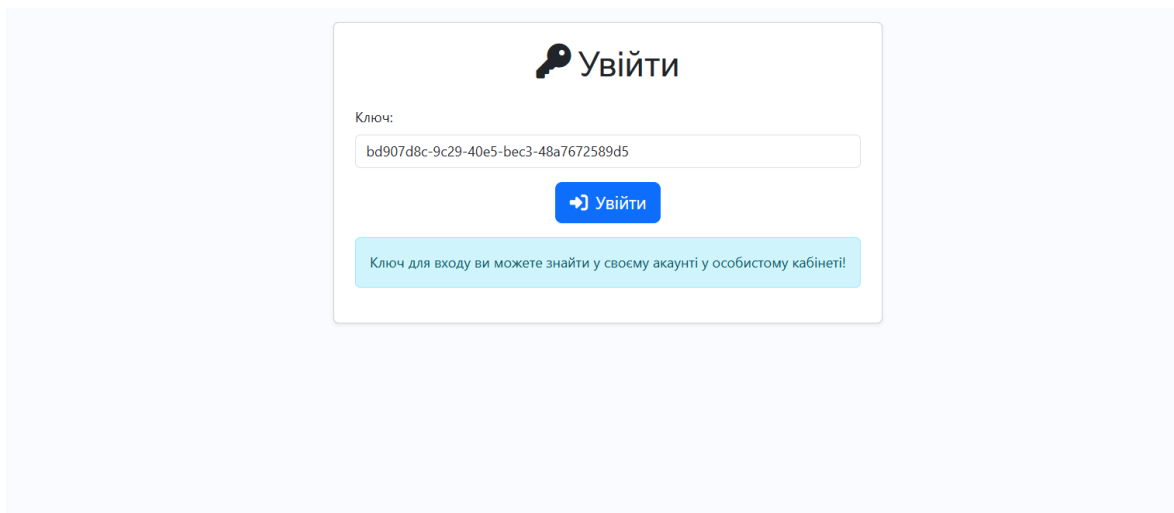


Рисунок 3.24 – Сторінка для входу за ключем

Після натиснення кнопки входу на сторінці входу, користувача переносить на сторінку для надсилання запитів (рисунок 3.25). Вона розділена на 2 частини: більшу частину займає зона для вводу відповіді, вводу значень полів та отримання відповіді, меншу частину займає список всі доступних запитів для користувача по якому можна шукати необхідний тобі запит. Якщо текст запиту не введено або метод запиту не вибрано, то виводиться відповідне повідомлення. Якщо введеного запиту не знайдено, то виводиться відповідне повідомлення. При введенні запиту, який існує в системі, появляються поля запиту. Якщо формат відповіді неправильний, то виводиться відповідне повідомлення. Після натиснення кнопки «Надіслати» надсилається запит на сервер. Під зоною для вводу значень полів знаходиться відповідь, що була отримана в результаті надсилання запиту.

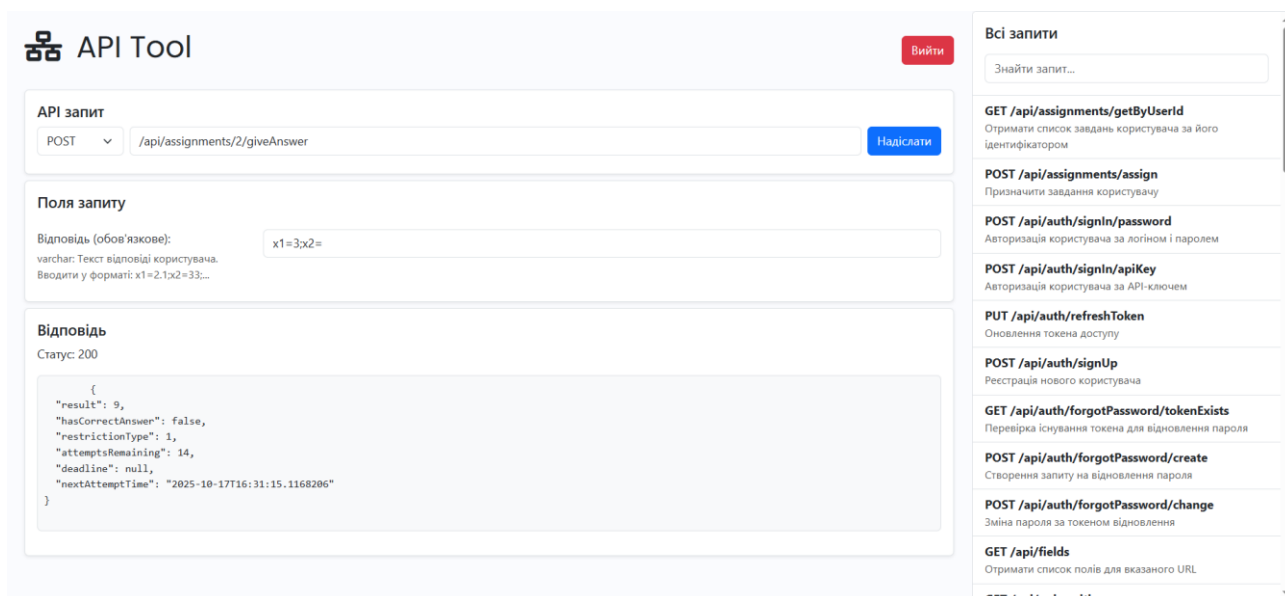


Рисунок 3.25 – Сторінка для надсилання запитів

Таким чином можна описати алгоритм взаємодії сервера та клієнтського застосунку і в контексті всіх інших операцій додатку.

3.4 Тестування програмного забезпечення

Тестування програмного забезпечення виконується для того щоб:

- перевірити правильність роботи програмного забезпечення відповідно до функціональних вимог;
- перевірити правильність роботи програмного забезпечення відповідно до нефункціональних вимог;
- перевірити сумісність веб-додатку з останніми версіями сучасних браузерів (Chrome, Opera, Firefox) на персональних комп'ютерах та ноутбуках;
- перевірити сумісність веб-додатку із браузерами на мобільних пристроях.

Перевірка функціональних вимог, коректності збереження даних тощо була проведена за допомогою мануального тестування згідно наперед визначених варіантів використання додатку, що описані у вигляді тест-кейсів. Тест-кейси, що використовувалися для перевірки правильної роботи основних функцій додатку наведено у таблицях 3.9 – 3.21.

Таблиця 3.9 – Тест 1.1

Тест	Реєстрація студента
Модуль	Реєстрація
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю, університет, роль студента
Опис проведення тесту	У відповідні поля вводяться: електронна пошта у правильному форматі, яка до цього не була зареєстрована в системі, пароль від 8 символів, підтвердження паролю, яке співпадає з раніше введеним паролем, вибирається роль студента, вибирається університет студента. Після цього натискається кнопка підтвердження реєстрації
Очікуваний результат	Реєстрація проходить успішно, зареєстрований студент додається у систему і перенаправляється на сторінку авторизації
Фактичний результат	Реєстрація проходить успішно, зареєстрований студент додається у систему і перенаправляється на сторінку авторизації

Таблиця 3.10 – Тест 1.2

Тест	Реєстрація викладача
Модуль	Реєстрація
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю, університет, роль викладача

Продовження таблиці 3.10

Опис проведення тесту	У відповідні поля вводяться: електронна пошта у правильному форматі, яка до цього не була зареєстрована в системі, пароль від 8 символів, підтвердження паролю, яке співпадає з раніше введеним паролем, вибирається роль викладача, вибирається університет викладача. Після цього натискається кнопка підтвердження реєстрації
Очікуваний результат	Реєстрація проходить успішно, зареєстрований викладач додається у систему і перенаправляється на сторінку авторизації
Фактичний результат	Реєстрація проходить успішно, зареєстрований викладач додається у систему і перенаправляється на сторінку авторизації

Таблиця 3.11 – Тест 2.1

Тест	Авторизація користувача
Модуль	Авторизація
Номер тесту	2.1
Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна адреса, пароль
Опис проведення тесту	У відповідні поля вводяться електронна адреса та пароль, що знаходяться в системі
Очікуваний результат	Користувач потрапляє на сторінку із завданнями, якщо користувач є студентом чи викладачем; користувач потрапляє на сторінку із користувачами, якщо користувач є адміністратором

Продовження таблиці 3.11

Фактичний результат	Користувач потрапляє на сторінку із завданнями, якщо користувач є студентом чи викладачем; користувач потрапляє на сторінку із користувачами, якщо користувач є адміністратором
---------------------	---

Таблиця 3.12 – Тест 3.1

Тест	Вихід із системи користувача
Модуль	Вихід із системи
Номер тесту	3.1
Початковий стан системи	Користувач авторизований
Вхідні дані	-
Опис проведення тесту	Користувач натискає кнопку виходу із системи
Очікуваний результат	Користувач потрапляє на сторінку авторизації
Фактичний результат	Користувач потрапляє на сторінку авторизації

Таблиця 3.13 – Тест 4.1

Тест	Отримання завдання студентом
Модуль	Виконання завдання
Номер тесту	4.1
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданнями
Вхідні дані	Теми

Продовження таблиці 3.13

Опис проведення тесту	Користувач натискає кнопку отримання завдання. Користувач вибирає теми, по яким хоче пройти тестування. Користувач натискає на кнопку отримання завдання
Очікуваний результат	Користувач потрапляє на сторінку із завданнями, де відображається отримане завдання
Фактичний результат	Користувач потрапляє на сторінку із завданнями, де відображається отримане завдання

Таблиця 3.14 – Тест 4.2

Тест	Початок виконання завдання
Модуль	Виконання завдання
Номер тесту	4.2
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданнями
Вхідні дані	Користувач має присвоєні йому завдання
Опис проведення тесту	Користувач натискає кнопку початку виконання завдання
Очікуваний результат	Користувач потрапляє на сторінку із завданням
Фактичний результат	Користувач потрапляє на сторінку із завданням

Таблиця 3.15 – Тест 4.3

Тест	Введення неправильної відповіді
Модуль	Виконання завдання
Номер тесту	4.3
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням

Продовження таблиці 3.15

Вхідні дані	Функція завдання – x^2
Опис проведення тесту	Користувач вводить 1 у поле для вводу відповіді. Користувач натискає кнопку надсилення відповіді
Очікуваний результат	Відображається інформація про значення функції для введеного значення та інформація про те, що ця відповідь є неправильною
Фактичний результат	Відображається інформація про значення функції для введеного значення та інформація про те, що ця відповідь є неправильною

Таблиця 3.16 – Тест 4.4

Тест	Введення правильної відповіді
Модуль	Виконання завдання
Номер тесту	4.4
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням
Вхідні дані	Функція завдання – x^2
Опис проведення тесту	Користувач вводить 0 у поле для вводу відповіді. Користувач натискає кнопку надсилення відповіді
Очікуваний результат	Відображається інформація про значення функції для введеного значення та інформація про те, що ця відповідь є правильною; блокується поле для введення відповіді; блокується кнопка для відправки відповіді
Фактичний результат	Відображається інформація про значення функції для введеного значення та інформація про те, що ця відповідь є правильною; блокується поле для введення відповіді; блокується кнопка для відправки відповіді

Таблиця 3.17 – Тест 4.5

Тест	Введення неправильної відповіді при кількості спроб, що залишилися, що дорівнює 1
Модуль	Виконання завдання
Номер тесту	4.5
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням; залишилася 1 спроба для виконання завдання
Вхідні дані	Функція завдання – x^2
Опис проведення тесту	Користувач вводить 1 у поле для вводу відповіді. Користувач натискає кнопку надсилення відповіді
Очікуваний результат	Відображається інформація про значення функції для введеного значення, інформація про те, що ця відповідь є неправильною, інформація про час, коли може бути надіслана наступна спроба; блокується поле для введення відповіді; блокується кнопка для відправки відповіді
Фактичний результат	Відображається інформація про значення функції для введеного значення, інформація про те, що ця відповідь є неправильною, інформація про час, коли може бути надіслана наступна спроба; блокується поле для введення відповіді; блокується кнопка для відправки відповіді

Таблиця 3.18 – Тест 4.6

Тест	Введення неправильної відповіді при типі обмеження завдання на кількість хвилин на 1 спробу
Модуль	Виконання завдання
Номер тесту	4.6

Продовження таблиці 3.18

Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням; тип обмеження завдання – кількість хвилин на 1 спробу
Вхідні дані	Функція завдання – x^2
Опис проведення тесту	Користувач вводить 1 у поле для вводу відповіді. Користувач натискає кнопку надсилення відповіді
Очікуваний результат	Відображається інформація про значення функції для введеного значення, інформація про те, що ця відповідь є неправильною, інформація про час, коли може бути надіслана наступна спроба; блокується поле для введення відповіді; блокується кнопка для відправки відповіді
Фактичний результат	Відображається інформація про значення функції для введеного значення, інформація про те, що ця відповідь є неправильною, інформація про час, коли може бути надіслана наступна спроба; блокується поле для введення відповіді; блокується кнопка для відправки відповіді

Таблиця 3.19 – Тест 4.7

Тест	Введення відповіді при типі обмеження завдання з дедлайном
Модуль	Виконання завдання
Номер тесту	4.7
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням; тип обмеження завдання – дедлайн; час виконання завдання – після дедлайну
Вхідні дані	Функція завдання – x^2

Продовження таблиці 3.19

Опис проведення тесту	Користувач вводить 1 у поле для вводу відповіді. Користувач натискає кнопку надсилання відповіді
Очікуваний результат	Відображається інформація про те, що відповідь не прийнята, бо пройшов дедлайн виконання завдання; блокується поле для введення відповіді; блокується кнопка для відправки відповіді
Фактичний результат	Відображається інформація про те, що відповідь не прийнята, бо пройшов дедлайн виконання завдання; блокується поле для введення відповіді; блокується кнопка для відправки відповіді

Таблиця 3.20 – Тест 4.8

Тест	Завершення виконання завдання
Модуль	Виконання завдання
Номер тесту	4.8
Початковий стан системи	Користувач авторизований як студент; користувач знаходиться на сторінці із завданням
Вхідні дані	Функція завдання – x^2
Опис проведення тесту	Користувач вводить 0 у поле для вводу відповіді. Користувач натискає кнопку надсилання відповіді. Користувач натискає кнопку завершення виконання завдання
Очікуваний результат	Користувач переходить на сторінку із завданнями; завершене завдання відображається у таблиці із завершеними завдання
Фактичний результат	Користувач переходить на сторінку із завданнями; завершене завдання відображається у таблиці із завершеними завдання

Таблиця 3.21 – Тест 5.1

Тест	Оцінювання завдання
Модуль	Оцінювання завдання
Номер тесту	5.1
Початковий стан системи	Користувач авторизований як викладач; користувач знаходиться на сторінці із завданнями; існує хоча б 1 завершене завдання студента, до якого викладач має доступ
Вхідні дані	Оцінка, коментар
Опис проведення тесту	Користувач натискає на кнопку виставлення оцінки. Користувач вводить коректне числове значення у поле для вводу оцінки та будь-який текст у поле для вводу коментаря. Користувач натискає на кнопку підтвердження оцінки
Очікуваний результат	Вікно для виставлення оцінки закривається; оцінка та коментар відображаються навпроти завдання, для якого виставлялася оцінка
Фактичний результат	Вікно для виставлення оцінки закривається; оцінка та коментар відображаються навпроти завдання, для якого виставлялася оцінка

Висновки до розділу

В першому підрозділі – «Опис схеми баз даних» – були описані всі сутності, що існують в розробленій базі даних, всі таблиці, що були створені в базі даних, а також показано зв'язки між полями таблиць бази даних у вигляді ER- діаграми бази даних.

В другому підрозділі – «Опис структури програмного продукту» – було розписано архітектуру програмного забезпечення на прикладі запиту, що проходить від клієнта на сервер і отримання відповіді назад до клієнта. Також було наведено всі основні контролери, сервіси, репозиторії на стороні серверу (та їх функціонал), а також всі модулі та окремі сервіси на стороні клієнтського

додатку (та їх функціонал).

В третьому підрозділі – «Опис інтерфейсу програмного забезпечення» – було продемонстровано графічний інтерфейс обох графічних додатків на прикладі основного функціоналу застосунку – отримання та проходження завдання.

В четвертому підрозділі – «Тестування програмного забезпечення» – були написані тест-кейси, що використовувалися при мануальному тестуванні для перевірки дотримання основних функціональних вимог додатку.

4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Провівши аналіз існуючих підходів до організації лабораторних занять, було виявлено недостатність інструментів, які дозволяють виконувати задачі з аналізу складних об'єктів, так щоб процес аналізу був максимально наближеним до реальних задач.

Тому було вирішено створити програмне забезпечення для проведення лабораторних робіт, яке:

- приховує формальний опис об'єкта («чорна скриня»);
- дозволяє студенту взаємодіяти з системою ітеративно;
- підтримує кілька типів обмежень (дедлайн, кількість спроб, ліміт часу);
- надає викладачу гнучкі налаштування завдань.

В таблиці 4.1 описано ідею стартап-проєкту.

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розроблено програмне забезпечення, що дозволяє виконувати лабораторні завдання з аналізу складних об'єктів у режимі «чорної скрині». Система забезпечує ітеративну взаємодію, приховує формальний опис задачі та підтримує декілька типів обмежень.	Вищі навчальні заклади; факультети, що викладають оптимізацію, машинне навчання, дискретну математику, чисельні методи тощо.	Студент отримує середовище, де можна досліджувати складні об'єкти без повного формального опису. Викладач – гнучкий інструмент для створення лабораторних робіт та контролю знань.

Таблиця 4.2 – Порівняльна характеристика з товарами-аналогами

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Moodle	Google Forms			
1	Гнучкість налаштувань	Середня	Висока	Висока		+	
2	Підтримка ітеративних методів	Висока	Низька	Низька			+
3	Модель «чорної скрині»	Висока	Середня	Середня			+
4	Обмеження (дедлайн/спроби/час)	Висока	Середня	Середня			+
5	Гнучкість оцінювання	Середня	Висока	Висока		+	
6	Інтеграція з освітніми платформами	Низька	Висока	Висока	+		
7	Розширюваність	Висока	Висока	Висока			+

4.2 Технологічний аудит ідеї проєкту

Таблиця 4.3 – Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність
1	Серверна частина	Java, Spring Framework	Наявна	Доступна
2	Веб-інтерфейс	Angular, TypeScript	Наявна	Доступна
3	База даних	PostgreSQL	Наявна	Доступна

Підсумовуючи, технологічна реалізація продукту – можлива, для цього було обрано технології Java, Spring, Typescript, Angular та PostgreSQL.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 4.4 – Попередня характеристика потенційного ринку

№	Показник стану ринку	Характеристика
1	Кількість головних гравців, од	5–7
2	Загальний обсяг продаж, грн./ум.од	650 000 ₴
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Ні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	12–20%

Враховуючи показники наведені в таблиці 4.4 можна зробити висновок, що наразі ринок є доволі перспективним для впровадження стартап продукту.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба	Цільова аудиторія	Поведінкові особливості	Вимоги
1	Необхідність проводити лабораторні роботи зі складними об'єктами (функціями, моделями), де важлива ітеративна взаємодія та аналіз	Університетські кафедри технічних спеціальностей (математика, інформатика, кібернетика)	Прагнуть мати гнучкий інструмент для проведення лабораторних робіт; потребують контролю за логікою роботи студентів і можливості налаштування умов завдання	Наявність «чорної скрині»; підтримка ітеративного введення даних; можливість налаштовувати обмеження; зручний веб-інтерфейс; розширюваність через API

Продовження таблиці 4.5

№	Потреба	Цільова аудиторія	Поведінкові особливості	Вимоги
2	Потреба забезпечити якісну перевірку знань у студентів, зменшивши ризик списування та підказок від AI-сервісів	Викладачі технічних курсів	Зосереджуються на об'єктивності оцінювання; часто модифікують умови завдань; потребують індивідуальних налаштувань під конкретні групи або дисципліни	Можливість приховувати формальний опис задач; система, стійкість до GPT-підказок; динамічне зміна параметрів завдання;

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкурентні освітні платформи	Поширення універсальних LMS із модулем практичних робіт може зменшити потребу у новому продукті.	Розробка унікального функціоналу («чорна скриня», динамічні параметри), інтеграції через API, надання тестового доступу.
2	Низька готовність викладачів до нових цифрових інструментів	Частина викладачів віддає перевагу традиційним методам або вже звичним системам.	Проведення навчальних вебінарів, створення методичних матеріалів, надання шаблонів лабораторних робіт.
3	Обмежене фінансування університетів	Деякі факультети мають недостатній бюджет для закупівлі нових інструментів.	Гнучкі тарифи, академічні ліцензії, безкоштовний базовий доступ, можливість поетапного впровадження.

Таблиця 4.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання цифровізації	Університети активно впроваджують цифрові	Інтеграція з існуючими платформами через REST API;

Продовження таблиці 4.7

№	Фактор	Зміст можливості	Можлива реакція компанії
	освіти	інструменти для лабораторних робіт і контролю знань.	позиціонування як інструмент для практичних обчислювальних дисциплін.
2	Попит на спеціалізовані засоби для обчислювальних дисциплін	Викладачам бракує рішень з підтримкою «чорної скрині», багаторівневих обмежень та покрокової взаємодії студента із задачею.	Розширення функціоналу під конкретні курси (чисельні методи, машинне навчання тощо), впровадження адаптивних налаштувань задач.
3	Розвиток змішаного та дистанційного навчання	До лабораторних занять висуваються вимоги віддаленого доступу, повторного проходження та автоматизованої перевірки.	Додавання модулів для віддалених груп, логів спроб, автоматичного формування індивідуальних варіантів.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	На ринку є кілька LMS-систем та тестових платформ, але жодна не підтримує ітеративні методи та підхід «чорної скрині».	Необхідність формування унікальної пропозиції: модульність, підтримка обчислень, ітеративність.
2	Рівень конкурентної боротьби: регіональний/міжнародний	Є продукти для масового тестування, але вони не орієнтовані на задачі оптимізації та дослідження складних функцій.	Можливість зайняти вузьку нішу, уникаючи прямої конкуренції.

Продовження таблиці 4.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3	Галузева ознака: навчальні продукти ІТ-спрямування	Сфокусованість на університетських курсах з математики та програмування.	Фокус на високотехнологічному сегменті навчальних продуктів.
4	Конкуренція за видами товарів: спеціалізовані продукти	Більшість аналогів – лише тестові системи без розширеної логіки.	Позиціювання як науково-орієнтованої системи.
5	Переваги: функціональні, а не цінові	Головний чинник – унікальність підходу «чорної скрині».	Стратегія диференціації, а не демпінгу.
6	За інтенсивністю: марочна	Більшість систем мають відомі бренди (Moodle, Google).	Треба формувати власний бренд та відкритість коду.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Платформи для автоматизації оцінювання та навчання (Moodle, Google Forms, Classtime), що забезпечують базові функції тестування та керування курсами.	Нові рішення та стартапи, що можуть з'явитися і частково повторювати функції наявних систем.	—	Шкільні та університетські навчальні заклади, вчителі, методисти, які потребують інструментів для автоматизації оцінювання.	ІІІ-системи та інструменти, які частково можуть замінити традиційне тестування.

Продовження таблиці 4.9

Ви сн ов ки	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари- замінники
	Здорова конкуренція	Потенційне збільшення конкуренції	Відсутність постачальників	Клієнтська база зацікавлена у розширених функціях і специфічних обмеженнях, що створює можливості для диференціації.	Товари- замінники частково впливають на користувацькі очікування, але унікальні функції проєкту зменшують ризик заміщення.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Актуальність товару	Зростаюча потреба ВНЗ у цифрових інструментах для проведення лабораторних робіт з акцентом на аналіз складних об'єктів і практичні навички; відсутність масових рішень з підтримкою режиму «чорної скрині».
2	Прозорість створеного рішення	Відкритий підхід до документування алгоритмів, можливість аудиту модулів обчислень та зрозуміла логіка взаємодії підвищують довіру викладачів і методистів.
3	Інтеграція з освітніми системами	Наявність API та сумісності з популярними LMS (Moodle тощо) спрощує впровадження і підвищує привабливість для адміністрацій університетів.
4	Простота використання	Інтуїтивний веб-інтерфейс на Angular, готові шаблони задач і методичні матеріали знижують поріг входу для викладачів і студентів.

Продовження таблиці 4.10

№	Фактор конкурентоспроможності	Обґрунтування
5	Врахування особливостей лабораторних завдань	Підтримка ітеративної взаємодії, мультиобмежень (дедлайн/спроби/час) та спеціалізованих типів завдань робить продукт придатним саме для технічних дисциплін.
6	Адаптація до змін	Модульна архітектура (Java + Spring), масштабованість та можливість швидкого оновлення модулів дозволяють оперативно реагувати на зміни навчальних програм і вимог.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін програмного забезпечення для проведення лабораторних занять

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Актуальність товару	15				+			
2	Прозорість створеного рішення	18	+						
3	Інтеграція з освітніми системами	10						+	
4	Простота використання	13					+		
5	Врахування особливостей лабораторних завдань	17			+				
6	Адаптація до змін	17				+			

Таблиця 4.12 – SWOT аналіз стартап-проекту

Сильні сторони (S): Можливість проводити лабораторні заняття з використанням «чорної скрині» та ітеративного аналізу об'єкта.	Слабкі сторони (W): Потреба у стабільній IT-інфраструктурі для повноцінної роботи системи.
--	---

Продовження таблиці 4.12

Сильні сторони (S): Підвищення об'єктивності оцінювання завдяки автоматизації процесів перевірки. Гнучка система обмежень (дедлайни, кількість спроб, час на виконання).	Слабкі сторони (W): Необхідність навчання викладачів роботі з новим інструментом. Частина задач потребує ручної перевірки або індивідуального доопрацювання.
Можливості (O): Зростаюча увага до цифровізації навчального процесу в університетах. Можливість масштабування системи на інші дисципліни, де потрібні ітеративні методи дослідження. Використання системи в дистанційному та змішаному форматах навчання.	Загрози (T): Обмежений бюджет навчальних закладів на впровадження нового ПЗ. Наявність традиційних підходів, до яких викладачі можуть бути більш прихильні. Вихід альтернативних продуктів з ширшим спектром функцій.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Надання доступу до системи проведення лабораторних робіт у демо-режимі для викладачів та кафедр з можливістю тестування базових функцій.	Висока, оскільки основні ресурси – уже створене ПЗ та підтримка розробника.	1-2 місяці
2	Пілотне впровадження на окремих кафедрах або навчальних програмах університетів з подальшим розширенням за результатами відгуків.	Середня, залежить від наявності партнерств та готовності кафедр тестувати нові рішення.	2-4 місяці
3	Інтеграція із внутрішніми освітніми платформами університетів (LMS, системи контролю знань) для підвищення цінності продукту.	Середня/низька, потребує технічних ресурсів і комунікації з адміністраціями університетів.	4-6 місяців

4.4 Розроблення ринкової стратегії проєкту

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит у сегменті	Інтенсивність конкуренції	Простота входу в сегмент
1	Кафедри університетів, що проводять лабораторні заняття з програмування, інформатики, чисельних методів	Середня	Середній попит	Середня	Середня
2	Факультети технічних спеціальностей, що потребують автоматизації контролю знань і проедення лабораторних робіт	Висока	Високий попит	Низька/середня	Середня/висока
3	Коледжі та училища, де лабораторні заняття часто мають стандартизовану структуру	Низька/середня	Низький/середній попит	Середня	Низька
Які цільові групи обрано: кафедри університетів та технічні факультети					

На основі проведеного аналізу можна зробити висновок, що цільовою аудиторією для розповсюдження цього програмного продукту є кафедри університетів та технічні факультети, оскільки вони демонструють більшу готовність впроваджувати цифрові інструменти та мають стабільну потребу в автоматизації проведення лабораторних занять.

Таблиця 4.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Забезпечення гнучких інструментів для проведення лабораторних занять у середовищі «чорної скрині», автоматизації перевірки результатів та підтримки різних типів обмежень.	Стратегія концентрованого охоплення – фокус на університетах і технічних факультетах, які активно цифровізують навчальні процеси.	Висока адаптивність ПЗ до різних курсів; можливість налаштування параметрів лабораторних завдань; інтеграція з наявними системами університетів; інноваційність підходу «чорної скрині».	Стратегія клієнтської орієнтації – розвиток продукту виходячи з потреб викладачів і кафедр, впровадження модульності та навчальних матеріалів.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні – на ринку вже існують різні платформи для тестування та перевірки завдань, але вони не реалізують підхід «чорної скрині» для лабораторних робіт	Компанія буде орієнтуватися як на нових користувачів (викладачів, університети), так і на користувачів конкурентних систем	Копіюватиметься загальний принцип автоматизації перевірки, але ключовий фокус – на унікальності ітеративних лабораторних завдань та розширених типах обмежень	Стратегія експансії – розширення ринку за рахунок унікальної функціональності та залучення користувачів конкурентних рішень

Таблиця 4.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію
1	Можливість створювати лабораторні завдання з різними типами обмежень (кількість спроб, дедлайн, час на відповідь)	Стратегія концентрованого зростання	Модульна структура ПЗ дозволяє швидко розширювати набір обмежень та механізмів перевірки	Гнучкість, адаптивність
2	Підтримка ітеративної роботи у форматі «чорної скрині»	Стратегія інноваційного розвитку	Унікальний механізм прихованого формального опису об'єкта та покрокового дослідження	Інтерактивність, дослідницький підхід
3	Інтеграція з існуючими навчальними платформами та університетськими системами	Стратегія диверсифікованого зростання	Серверна частина містить набір відкритих API для підключення зовнішніх систем	Інтегрованість, відкритість
4	Простота використання та зрозумілий інтерфейс для викладача і студента	Стратегія клієнтської орієнтації	Інтуїтивний веб-інтерфейс, навчальні матеріали та приклади лабораторних робіт	Зручність, доступність

На основі проведеного аналізу можна зробити висновок, що для базової стратегії розвитку буде доцільно обрати стратегію клієнтської орієнтації, а для базової стратегії конкурентної поведінки – стратегію експансії.

4.5 Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Можливість виконання завдань у форматі «чорної скрині»	Студент поступово досліджує поведінку об'єкта, формуючи власні гіпотези	Унікальний механізм ітеративного дослідження без розкриття формальної моделі, відсутній у типових тестових системах
2	Гнучке створення лабораторних робіт з обмеженнями	Викладач сам визначає правила виконання: дедлайн, спроби, час, динамічні зміни умов	Глибока кастомізація умов, недоступна у більшості звичних платформ

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
1. Товар за задумом	Програмне забезпечення, що дозволяє проводити лабораторні заняття з можливістю приховування формальної моделі («чорна скриня»), та підтримкою різних сценаріїв виконання завдань.		
2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Інтерактивний веб-інтерфейс	М	Е
	Реалізація моделі «чорної скрині»	Нм	Вр/Тх/Тл
	Багатофункціональність	М	Тх
	Якість: правильність та ефективність доведені проведення мануального тестування згідно описаних тест-кейсів		
	Пакування: інформація та характеристика продукту будуть розміщені на офіційному сайті веб застосунку та відкритому GitHub репозиторію		
	Марка: LabMaster		

Продовження таблиці 4.19

Рівні товару	Сутність та складові
3. Товар із підкріпленням	До продажу: приклади лабораторних, методичні матеріали, ознайомчий доступ, відеоінструкції.
	Після продажу: технічна підтримка, оновлення модулів, консультації для викладачів.
За рахунок чого потенційний товар буде захищено від копіювання: ліцензія, захист вихідного коду серверної частини, патентування	

Таблиця 4.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи	Верхня та нижня межі встановлення ціни на товар / послугу
100–300 \$ на рік (платформи онлайн-тестування, LMS із базовими лабораторними модулями)	150–500 \$ на рік (спеціалізовані системи для технічних лабораторних робіт)	Залежить від бюджету освітніх установ; середній рівень витрат на ПЗ	120–350 \$ на рік за індивідуальну ліцензію або 400–900 \$ за інституційну ліцензію

Таблиця 4.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Освітні установи (університети, коледжі, школи) орієнтуються на перевірені цифрові рішення та очікують можливість швидкого впровадження ПЗ у навчальний процес.	Надання ліцензій, технічної підтримки, оновлень та консультацій щодо впровадження лабораторних модулів.	Багатоканальний збут (прямий та через партнерів)	Прямі продажі навчальним закладам, співпраця з освітніми сервісами, інтеграція з LMS-платформами

Таблиця 4.22 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Навчальні заклади прагнуть підвищити якість лабораторних занять та зменшити навантаження на викладачів завдяки цифровим рішенням.	Освітні форуми, презентації для закладів освіти, професійні спільноти, інтернет-платформи	Автоматизація лабораторних робіт, гнучкість налаштування, інтеграція з освітніми системами	Продемонструвати ефективність продукту, підкреслити зручність та адаптивність для різних дисциплін	«Сучасне ПЗ для організації лабораторних занять: швидко, зручно, інтегровано з вашими навчальними курсами»

У результаті проведеного аналізу було сформовано комплексну маркетингову стратегію, яка охоплює визначення ключових переваг майбутнього продукту, побудову його концептуальної моделі, обґрунтування цінових меж, розроблення системи збуту та формування підходів до маркетингових комунікацій.

Висновки до розділу

У цьому розділі було здійснено комплексний маркетинговий аналіз стартап-проєкту, спрямованого на розроблення методів та програмного забезпечення для проведення лабораторних занять. Було визначено ключові особливості проєктної ідеї, здійснено аналіз технологічної здійсненності та окреслено потенційні можливості застосування продукту в освітньому середовищі. Результати ринкового аналізу засвідчили наявність попиту на інструменти, що забезпечують автоматизацію, спрощення та уніфікацію процесу виконання лабораторних робіт у закладах вищої освіти.

На основі проведеного дослідження були визначені характеристики цільових груп споживачів, ключові загрози й можливості, а також сформовано SWOT-модель проєкту. Додатково було розглянуто конкурентне середовище та проведено аналіз за моделлю М. Портера, що дозволило визначити ступінь впливу конкурентів і потенційних замінників на позицію продукту на ринку.

Було обґрунтовано фактори конкурентоспроможності, визначено ринкову стратегію розвитку та конкурентної поведінки, а також сформовано позиціонування продукту з урахуванням реальних вимог споживачів. На завершення було побудовано маркетингову програму, що включає формування ключових переваг продукту, визначення ціни, організацію системи збуту та концепцію комунікацій.

У підсумку встановлено, що стартап-проєкт має високі перспективи впровадження в освітню сферу, зокрема завдяки актуальності проблематики, технологічній здійсненності та наявному ринковому запиту. Проєкт здатен забезпечити істотне спрощення та систематизацію процесу виконання лабораторних занять, що робить його перспективним для подальшого розвитку та масштабування.

ВИСНОВКИ

У ході виконання магістерської дисертації було розглянуто актуальне питання удосконалення процесу проведення лабораторних занять у закладах вищої освіти. Сформовано та проаналізовано основні проблеми існуючих систем автоматизованого тестування та контролю знань, зокрема відсутність можливості аналізувати вхідний об'єкт без повного формального опису, обмежені засоби контролю кількості спроб і часу, а також труднощі, пов'язані з забезпеченням достовірності результатів у дистанційному навчанні. Проведено огляд сучасних рішень, таких як Moodle, Coursera, Google Forms та їх функціональних обмежень, що стало підґрунтям для формування вимог до програмного забезпечення.

На основі проведеного аналізу було визначено та сформульовано завдання дослідження і обрано напрямок розробки програмного забезпечення, яке забезпечує взаємодію із завданнями за принципом «чорної скрині». Запропонований підхід дозволяє користувачу ітеративно подавати відповіді, отримувати проміжні результати виконання функції та робити висновки про поведінку об'єкта, що створює умови, подібні до вирішення реальних інженерних задач.

Відповідно до сформованих вимог розроблено вебзастосунок із розмежуванням ролей користувачів, який реалізує взаємодію з лабораторними завданнями у вигляді «чорної скрині», забезпечуючи ітеративне введення відповідей та отримання результатів залежно від попередніх спроб. Реалізовано функціональні можливості налаштування різних типів обмежень виконання завдань, засоби оцінювання та коментування результатів, а також клієнтські застосунки, які забезпечують зручну роботу викладачів і студентів.

Проведено тестування розробленого програмного забезпечення відповідно до функціональних і нефункціональних вимог, включаючи перевірку коректності обробки відповідей, підтримки обмежень, стабільності роботи та сумісності з різними браузерами та пристроями. Тестові сценарії підтвердили працездатність системи та відповідність запланованим вимогам.

Проведено маркетинговий аналіз розробленого рішення як потенційного стартап-проєкту. Отримані результати свідчать про перспективність впровадження програмного забезпечення в освітні заклади та комерційні навчальні платформи за умови правильної стратегії позиціонування продукту. Визначені базові напрямки розвитку та можливості масштабування створюють основу для подальшого виходу продукту на ринок освітніх технологій.

Наукова новизна результатів магістерської дисертації полягає в удосконаленні архітектури програмного забезпечення для проведення лабораторних занять за рахунок реалізації принципу «чорної скрині» та алгоритмів обробки обчислювальних задач, що забезпечує можливість аналізу складного об'єкта без розкриття його внутрішньої структури.

Практична значимість одержаних результатів полягає у створенні програмного забезпечення, яке забезпечує можливість організації лабораторних робіт із високим рівнем інтерактивності, підтримкою системи обмежень та аналізу поведінки складних об'єктів без надання їхнього формального опису. Це наближає навчальні завдання до реальних інженерних умов, підвищує мотивацію студентів і сприяє формуванню практичних навичок розробки ефективних алгоритмів та прийняття технічних рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Darling-Hammond, L. The Right to Learn: A Blueprint for Creating Schools that Work. / Darling-Hammond, L. – San Francisco: Jossey-Bass, 1997. – 398 с.
- 2) Bloom, B. S. Taxonomy of Educational Objectives: The Classification of Educational Goals. / Bloom, B. S. – New York: Longmans, Green, 1956. – 207 с.
- 3) Руденко, О. В. «Сучасні підходи до оцінювання навчальних досягнень учнів у закладах освіти». / Руденко, О. В. // Освітній простір України. – 2021 – № 4 – С. 112-118.
- 4) Garcia, L. Adaptive Learning Systems in Higher Engineering Education. / Garcia, L. – London: Academic Press, 2021. – 284 с.
- 5) Bennett, R. E. Technology and Testing: Improving Educational Assessment. / Bennett, R. E. – New York: Routledge, 2015. – 240 с.
- 6) Соловйов, В. М. Педагогічна діагностика: теорія і практика. / Соловйов, В. М. – Київ: Либідь, 2010. – 256 с.
- 7) Smith, J. Digital Assessment Technologies in Modern Universities. / Smith, J. – New York: Springer, 2022. – 312 с.
- 8) Thompson, R. Engineering Education in the Age of Automation. / Thompson, R. – Cambridge University Press, 2024. – 341 с.
- 9) Lopez, M. Cognitive Approaches to Computer-Based Problem Solving in STEM. / Lopez, M. – Berlin: De Gruyter, 2025. – 229 с.
- 10) ChatGPT. [Електронний ресурс] – Режим доступу: <https://chatgpt.com/>.
- 11) Google Gemini. [Електронний ресурс] – Режим доступу: <https://gemini.google.com/>.
- 12) Grok. [Електронний ресурс] – Режим доступу: <https://grok.com/>.
- 13) Moodle. [Електронний ресурс] – Режим доступу: <https://moodle.org/>.
- 14) «На урок». [Електронний ресурс] – Режим доступу: <https://naurok.com.ua/>.
- 15) «Всеосвіта». [Електронний ресурс] – Режим доступу:

<https://vseosvita.ua/>.

16) Coursera. [Електронний ресурс] – Режим доступу: <https://www.coursera.org/>.

17) Udemy. [Електронний ресурс] – Режим доступу: <https://www.udemy.com/>.

18) Udacity. [Електронний ресурс] – Режим доступу: <https://www.udacity.com/>.

19) Google Forms. [Електронний ресурс] – Режим доступу: <https://workspace.google.com/intl/uk/products/forms/>.

20) Kahoot!. [Електронний ресурс] – Режим доступу: <https://kahoot.com/>.

21) Quizlet. [Електронний ресурс] – Режим доступу: <https://quizlet.com/ua>.

22) Quizizz. [Електронний ресурс] – Режим доступу: <https://wayground.com/?lng=en>.

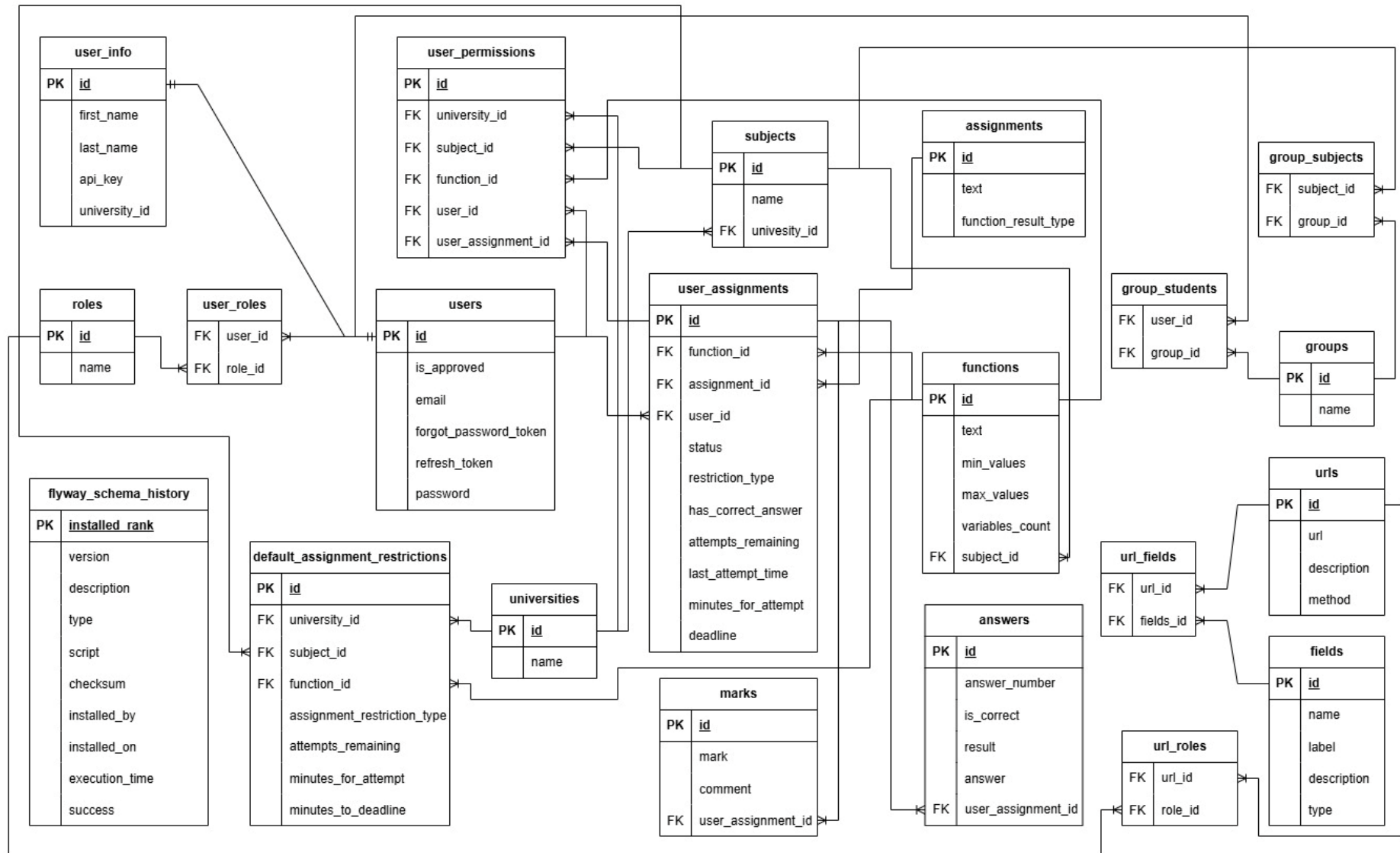
23) Кононенко О. Що краще моноліт чи мікросервіси? Як обрати архітектуру проекту? [Електронний ресурс] – Режим доступу: <https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu>.

24) Княжина Р. Архітектура вебзастосунків 2024: Ультимативний гайд для розробників. [Електронний ресурс] – Режим доступу: <https://robotdreams.cc/uk/blog/567-arhitektura-vebzastosunkiv>.

25) Java EE vs Spring: A Comparative Guide to Enterprise Java Development. [Електронний ресурс] – Режим доступу: <https://medium.com/@lktsdvd/java-ee-vs-spring-a-comparative-guide-to-enterprise-java-development-cdb5a5d0d2c7>.

26) Liquibase vs Flyway. [Електронний ресурс] – Режим доступу: <https://www.baeldung.com/liquibase-vs-flyway>.

27) Битва титанів: що краще – PostgreSQL чи MySQL?. [Електронний ресурс] – Режим доступу: <https://highload.tech/uk/postgresql-vs-mysql/>.



ДОДАТОК Б

Лістинг коду

Повний лістинг коду знаходиться у відкритому доступі за посиланням:
<https://github.com/Den-droid/MasterDissertation>.

Файл: `org/apiapplication/services/implementations/AssignmentRestrictionServiceImpl.java`

```
@Service
@Transactional
public class AssignmentRestrictionServiceImpl implements AssignmentRestrictionService {
    private final DefaultAssignmentRestrictionRepository defaultAssignmentRestrictionRepository;
    private final FunctionRepository functionRepository;
    private final SubjectRepository subjectRepository;
    private final UniversityRepository universityRepository;
    private final UserAssignmentRepository userAssignmentRepository;

    private final SessionService sessionService;
    private final PermissionService permissionService;

    public AssignmentRestrictionServiceImpl(DefaultAssignmentRestrictionRepository defaultAssignmentRestrictionRepository,
                                           FunctionRepository functionRepository,
                                           SubjectRepository subjectRepository,
                                           UniversityRepository universityRepository,
                                           SessionService sessionService,
                                           PermissionService permissionService,
                                           UserAssignmentRepository userAssignmentRepository) {
        this.defaultAssignmentRestrictionRepository = defaultAssignmentRestrictionRepository;
        this.functionRepository = functionRepository;
        this.subjectRepository = subjectRepository;
        this.universityRepository = universityRepository;
        this.userAssignmentRepository = userAssignmentRepository;

        this.sessionService = sessionService;
        this.permissionService = permissionService;
    }

    @Override
    public List<DefaultRestrictionDto> getDefault(Integer functionId, Integer subjectId, Integer universityId) {
        if (functionId != null) {
            Function function = functionRepository.findById(functionId)
                .orElseThrow(() -> new EntityWithIdNotFoundException(
                    EntityName.FUNCTION, String.valueOf(functionId)
                ));

            if (!permissionService.userCanAccessFunction(sessionService.getCurrentUser(),
                function)) {
                throw new PermissionException();
            }

            DefaultAssignmentRestriction restriction = getDefaultRestrictionForFunction(function);
            if (restriction != null) {
                return List.of(getDefaultAssignmentRestrictionDto(restriction));
            } else {
                return List.of();
            }
        } else if (subjectId != null) {
            Subject subject = subjectRepository.findById(subjectId)
                .orElseThrow(() -> new EntityWithIdNotFoundException(
                    EntityName.SUBJECT, String.valueOf(subjectId)
                ));

            if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
                subject)) {
                throw new PermissionException();
            }

            DefaultAssignmentRestriction restriction = getDefaultRestrictionForSubject(subject);
            if (restriction != null) {
                return List.of(getDefaultAssignmentRestrictionDto(restriction));
            } else {
                return List.of();
            }
        }
    }
}
```

```

        return List.of();
    }
} else if (universityId != null) {
    University university = universityRepository.findById(universityId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(
            EntityName.UNIVERSITY, String.valueOf(universityId)
        ));

    if (!permissionService.userCanAccessUniversity(sessionService.getCurrentUser(),
        university)) {
        throw new PermissionException();
    }

    DefaultAssignmentRestriction restriction = getDefaultRestrictionForUniversity(university);
    if (restriction != null) {
        return List.of(getDefaultAssignmentRestrictionDto(restriction));
    } else {
        return List.of();
    }
} else {
    List<UserPermission> permissions = sessionService.getCurrentUser().getUserPermissions();
    return permissions.stream()
        .map(p -> {
            if (p.getUniversity() != null) {
                return p.getUniversity().getDefaultAssignmentRestrictions();
            } else if (p.getSubject() != null) {
                return p.getSubject().getDefaultAssignmentRestrictions();
            } else {
                return p.getFunction().getDefaultAssignmentRestrictions();
            }
        })
        .flatMap(Collection::stream)
        .map(this::getDefaultAssignmentRestrictionDto)
        .toList();
}
}

@Override
public RestrictionDto getCurrent(Integer userAssignmentId) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId)));

    return getAssignmentRestrictionDto(userAssignment);
}

@Override
public DefaultAssignmentRestriction getDefaultRestrictionForFunction(Function function) {
    List<DefaultAssignmentRestriction> defaultAssignmentRestrictions =
        defaultAssignmentRestrictionRepository.findAll();

    Optional<DefaultAssignmentRestriction> defaultRestriction = defaultAssignmentRestrictions.stream()
        .filter(restriction -> restriction.getFunction() != null &&
            Objects.equals(restriction.getFunction().getId(),
                function.getId()))
        .findFirst();

    if (defaultRestriction.isPresent()) {
        return defaultRestriction.get();
    }

    Subject subject = function.getSubject();

    return getDefaultRestrictionForSubject(subject);
}

@Override
public DefaultAssignmentRestriction getDefaultRestrictionForSubject(Subject subject) {
    List<DefaultAssignmentRestriction> defaultAssignmentRestrictions =
        defaultAssignmentRestrictionRepository.findAll();

    Optional<DefaultAssignmentRestriction> defaultRestriction = defaultAssignmentRestrictions.stream()
        .filter(restriction -> restriction.getSubject() != null &&
            Objects.equals(restriction.getSubject().getId(),
                subject.getId()))
        .findFirst();

    if (defaultRestriction.isPresent()) {

```



```

        dar.getSubject().getId().equals(dto.subjectId())) ||
        (dar.getFunction() != null &&
         dar.getFunction().getId().equals(dto.functionId()))
    ).findFirst();

    if (existingRestriction.isPresent()) {
        defaultAssignmentRestriction = existingRestriction.get();
    } else {
        defaultAssignmentRestriction = new DefaultAssignmentRestriction();
        defaultAssignmentRestriction.setFunction(function);
        defaultAssignmentRestriction.setSubject(subject);
        defaultAssignmentRestriction.setUniversity(university);
    }

    setAssignmentRestriction(dto, defaultAssignmentRestriction);
    defaultAssignmentRestrictionRepository.save(defaultAssignmentRestriction);
}

@Override
public void setRestriction(RestrictionDto dto) {
    List<UserAssignment> userAssignments = new ArrayList<>();

    UserAssignment userAssignment = null;
    Function function = null;
    Subject subject = null;
    University university = null;

    if (dto.userAssignmentId() != null) {
        userAssignment = userAssignmentRepository.findById(dto.userAssignmentId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
                String.valueOf(dto.userAssignmentId())));

        if (!permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
            userAssignment)) {
            throw new PermissionException();
        }

        userAssignments.add(userAssignment);
    } else if (dto.functionId() != null) {
        function = functionRepository.findById(dto.functionId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.FUNCTION,
                String.valueOf(dto.functionId())));

        if (!permissionService.userCanAccessFunction(sessionService.getCurrentUser(),
            function)) {
            throw new PermissionException();
        }

        userAssignments.addAll(function.getUserAssignments());
    } else if (dto.subjectId() != null) {
        subject = subjectRepository.findById(dto.subjectId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.SUBJECT,
                String.valueOf(dto.subjectId())));

        if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
            subject)) {
            throw new PermissionException();
        }

        subject.getFunctions().stream()
            .flatMap(f -> f.getUserAssignments().stream())
            .forEach(userAssignments::add);
    } else if (dto.universityId() != null) {
        university = universityRepository.findById(dto.universityId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
                String.valueOf(dto.universityId())));

        if (!permissionService.userCanAccessUniversity(sessionService.getCurrentUser(),
            university)) {
            throw new PermissionException();
        }

        university.getSubjects().stream()
            .flatMap(s -> s.getFunctions().stream())
            .flatMap(f -> f.getUserAssignments().stream())
            .forEach(userAssignments::add);
    }
}

```

```

        if (university == null && subject == null && function == null && userAssignment == null) {
            return;
        }

        for (UserAssignment ua : userAssignments) {
            setAssignmentRestriction(dto, ua);
        }

        userAssignmentRepository.saveAll(userAssignments);
    }

    @Override
    public List<RestrictionTypeDto> getRestrictionTypes() {
        if (sessionService.getCurrentUser() == null) {
            throw new PermissionException();
        }

        return Arrays.stream(AssignmentRestrictionType.values())
            .map(rt -> new RestrictionTypeDto(rt.ordinal(), rt.name()))
            .toList();
    }

    private DefaultRestrictionDto getDefaultAssignmentRestrictionDto(DefaultAssignmentRestriction
        defaultAssignmentRestriction) {
        return new DefaultRestrictionDto(defaultAssignmentRestriction.getId(),
            defaultAssignmentRestriction.getAssignmentRestrictionType().ordinal(),
            defaultAssignmentRestriction.getFunction() != null ?
                defaultAssignmentRestriction.getFunction().getId() : null,
            defaultAssignmentRestriction.getSubject() != null ?
                defaultAssignmentRestriction.getSubject().getId() : null,
            defaultAssignmentRestriction.getUniversity() != null ?
                defaultAssignmentRestriction.getUniversity().getId() : null,
            defaultAssignmentRestriction.getAttemptsRemaining(),
            defaultAssignmentRestriction.getMinutesForAttempt(),
            defaultAssignmentRestriction.getDeadline()
        );
    }

    private RestrictionDto getAssignmentRestrictionDto(UserAssignment userAssignment) {
        return new RestrictionDto(userAssignment.getRestrictionType().ordinal(),
            null, null, null, userAssignment.getId(),
            userAssignment.getAttemptsRemaining(),
            userAssignment.getMinutesForAttempt(),
            userAssignment.getDeadline()
        );
    }

    private void setAssignmentRestriction(RestrictionDto dto, UserAssignment userAssignment) {
        userAssignment.setRestrictionType(
            Arrays.stream(AssignmentRestrictionType.values())
                .filter(x -> dto.restrictionType()
                    == x.ordinal())
                .findFirst().orElseThrow(() ->
                    new EntityNotFoundException(EntityName.RESTRICTION_TYPE,
                        String.valueOf(dto.restrictionType())))
        );

        if (dto.restrictionType() == AssignmentRestrictionType.N_ATTEMPTS.ordinal()) {
            userAssignment.setRestrictionType(
                AssignmentRestrictionType.N_ATTEMPTS);
            if (dto.attemptsRemaining() != null)
                userAssignment.setAttemptsRemaining(
                    dto.attemptsRemaining());
        } else if (dto.restrictionType() == AssignmentRestrictionType.DEADLINE.ordinal()) {
            userAssignment.setRestrictionType(
                AssignmentRestrictionType.DEADLINE);
            if (dto.deadline() != null)
                userAssignment.setDeadline(dto.deadline());
        } else {
            userAssignment.setRestrictionType(
                AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES);
            if (dto.minutesForAttempt() != null)
                userAssignment.setMinutesForAttempt(
                    dto.minutesForAttempt());
        }
    }

    private void setAssignmentRestriction(DefaultRestrictionDto dto,

```

```

        DefaultAssignmentRestriction defaultAssignmentRestriction) {
    defaultAssignmentRestriction.setAssignmentRestrictionType(
        Arrays.stream(AssignmentRestrictionType.values())
            .filter(x -> dto.restrictionType()
                == x.ordinal())
            .findFirst().orElseThrow(() ->
                new EntityWithIdNotFoundException(EntityName.RESTRICTION_TYPE,
                    String.valueOf(dto.restrictionType()))));
    );

    if (dto.restrictionType() == AssignmentRestrictionType.N_ATTEMPTS.ordinal()) {
        defaultAssignmentRestriction.setAssignmentRestrictionType(
            AssignmentRestrictionType.N_ATTEMPTS);
        if (dto.attemptsRemaining() != null)
            defaultAssignmentRestriction.setAttemptsRemaining(
                dto.attemptsRemaining());
    } else if (dto.restrictionType() == AssignmentRestrictionType.DEADLINE.ordinal()) {
        defaultAssignmentRestriction.setAssignmentRestrictionType(
            AssignmentRestrictionType.DEADLINE);
        if (dto.deadline() != null)
            defaultAssignmentRestriction.setDeadline(dto.deadline());
    } else {
        defaultAssignmentRestriction.setAssignmentRestrictionType(
            AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES);
        if (dto.minutesForAttempt() != null)
            defaultAssignmentRestriction.setMinutesForAttempt(
                dto.minutesForAttempt());
    }
}
}
}

```

Файл: org/apiapplication/services/implementations/AssignmentServiceImpl.java

```

@Service
@Transactional
public class AssignmentServiceImpl implements AssignmentService {
    private final UserAssignmentRepository userAssignmentRepository;
    private final AssignmentRepository assignmentRepository;
    private final AnswerRepository answerRepository;
    private final SubjectRepository subjectRepository;
    private final GroupRepository groupRepository;

    private final PermissionService permissionService;
    private final AssignmentRestrictionService assignmentRestrictionService;
    private final SessionService sessionService;

    public AssignmentServiceImpl(UserAssignmentRepository userAssignmentRepository,
        AssignmentRepository assignmentRepository,
        AnswerRepository answerRepository,
        SubjectRepository subjectRepository, GroupRepository groupRepository,
        PermissionService permissionService,
        AssignmentRestrictionService
            assignmentRestrictionService,
        SessionService sessionService) {
        this.userAssignmentRepository = userAssignmentRepository;
        this.assignmentRepository = assignmentRepository;
        this.answerRepository = answerRepository;
        this.subjectRepository = subjectRepository;
        this.groupRepository = groupRepository;

        this.permissionService = permissionService;
        this.assignmentRestrictionService = assignmentRestrictionService;
        this.sessionService = sessionService;
    }

    @Override
    public AssignmentDto getById(int userAssignmentId) {
        UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
                String.valueOf(userAssignmentId))
        );
    };

    if (!permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
        userAssignment)) {
        throw new PermissionException();
    }

    return new AssignmentDto(
        userAssignment.getAssignment().getText(),

```

```

        userAssignment.getFunction().getVariablesCount(),
        userAssignment.getStatus().ordinal(),
        userAssignment.getRestrictionType().ordinal(),
        userAssignment.getAttemptsRemaining(),
        userAssignment.getDeadline(),
        userAssignment.getLastAttemptTime() != null ?
            userAssignment.getLastAttemptTime()
                .plusMinutes(userAssignment.getMinutesForAttempt())
            : LocalDateTime.now()
    );
}

@Override
public List<UserAssignmentDto> get() {
    List<UserAssignment> assignments;
    User user = sessionService.getCurrentUser();

    Role role = user.getRoles().get(0);

    if (role.getName().equals(UserRole.STUDENT)) {
        assignments = user.getUserAssignments();
    } else {
        assignments = getByUser(user).stream().toList();
    }

    List<UserAssignmentDto> userAssignmentDtos = assignments.stream()
        .map(userAssignment -> {
            List<Mark> marks = userAssignment.getMarks();
            Mark mark = null;
            if (marks != null && !marks.isEmpty()) {
                mark = marks.get(0);
            }

            Assignment assignment = userAssignment.getAssignment();

            return new UserAssignmentDto(userAssignment.getId(),
                assignment.getText(),
                userAssignment.getStatus().ordinal(),
                assignment.getFunctionResultType().ordinal(),
                userAssignment.getRestrictionType().ordinal(),
                userAssignment.getAttemptsRemaining(),
                userAssignment.getDeadline(),
                userAssignment.getLastAttemptTime() != null ?
                    userAssignment.getLastAttemptTime()
                        .plusMinutes(userAssignment.getMinutesForAttempt())
                    : LocalDateTime.now(),
                mark != null ? new MarkDto(mark.getId(), mark.getMark(), mark.getComment())
                    : null,
                getUserDto(userAssignment.getUser())
            );
        })
        .toList();

    return userAssignmentDtos;
}

@Override
public void assign(AssignDto dto) {
    User user = sessionService.getCurrentUser();

    if (!sessionService.isUserStudent(user)) {
        throw new PermissionException();
    }

    boolean allPermittedSubjects = subjectRepository.findAll().stream()
        .filter(s -> dto.subjectIds().contains(s.getId()))
        .map(Subject::getUniversity)
        .allMatch(u -> user.getUserInfo().getUniversity().equals(u));

    if (!allPermittedSubjects) {
        throw new PermissionException();
    }

    List<Subject> subjects = subjectRepository.findAll().stream()
        .filter(s -> dto.subjectIds().contains(s.getId()))
        .toList();

    assign(List.of(user), subjects);
}

```

```

}

@Override
public void assign(AssignGroupDto dto) {
    User currentUser = sessionService.getCurrentUser();

    Group group = groupRepository.findById(dto.groupId())
        .orElseThrow(() -> new EntityWithIdNotFoundException(
            EntityName.GROUP, String.valueOf(dto.groupId())
        ));

    if (!group.getOwner().getId().equals(currentUser.getId()))
        throw new PermissionException();

    List<Subject> subjects = group.getSubjects().stream().toList();

    assign(group.getStudents().stream().toList(), subjects);
}

private void assign(List<User> users, List<Subject> subjects) {
    List<UserAssignment> allUserAssignments = new ArrayList<>();

    List<Function> allFunctions = subjects.stream()
        .flatMap(f -> f.getFunctions().stream())
        .toList();
    List<Assignment> allAssignments = assignmentRepository.findAll();

    for (User user : users) {
        List<UserAssignment> possibleUserAssignments = new ArrayList<>();

        for (Function function : allFunctions) {
            for (Assignment assignment : allAssignments) {
                if (assignment.getFunctionResultType().equals(FunctionResultType.MIN) &&
                    !getMinMaxValuesForFunction(function, FunctionResultType.MIN).isEmpty())
                    possibleUserAssignments.add(new UserAssignment(user, function, assignment));
                else if (assignment.getFunctionResultType().equals(FunctionResultType.MAX) &&
                    !getMinMaxValuesForFunction(function, FunctionResultType.MAX).isEmpty()) {
                    possibleUserAssignments.add(new UserAssignment(user, function, assignment));
                }
            }
        }

        if (possibleUserAssignments.isEmpty())
            throw new NoAvailableAssignmentsException();

        List<UserAssignment> userAssignments = user.getUserAssignments();
        possibleUserAssignments.removeAll(userAssignments);

        Random random = new Random();
        int possibleFunctionsCount = possibleUserAssignments.size();
        int possibleFunctionIndex = random.nextInt(possibleFunctionsCount);

        UserAssignment userAssignment = new UserAssignment();
        userAssignment.setAssignment(possibleUserAssignments.get(possibleFunctionIndex).getAssignment());
        userAssignment.setUser(user);
        userAssignment.setFunction(possibleUserAssignments.get(possibleFunctionIndex).getFunction());
        userAssignment.setHasCorrectAnswer(false);
        userAssignment.setStatus(AssignmentStatus.ASSIGNED);

        DefaultAssignmentRestriction defaultAssignmentRestriction =
            assignmentRestrictionService.getDefaultRestrictionForFunction(userAssignment.getFunction());

        if (defaultAssignmentRestriction == null) {
            defaultAssignmentRestriction = assignmentRestrictionService.getDefaultRestriction();
        }

        if (defaultAssignmentRestriction.getAssignmentRestrictionType()
            .equals(AssignmentRestrictionType.N_ATTEMPTS)) {
            userAssignment.setRestrictionType(AssignmentRestrictionType.N_ATTEMPTS);
            userAssignment.setAttemptsRemaining(defaultAssignmentRestriction.getAttemptsRemaining());
        } else if (defaultAssignmentRestriction.getAssignmentRestrictionType()
            .equals(AssignmentRestrictionType.DEADLINE)) {
            userAssignment.setRestrictionType(AssignmentRestrictionType.DEADLINE);
            userAssignment.setDeadline(defaultAssignmentRestriction.getDeadline());
        } else {
            userAssignment.setRestrictionType(AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES);
            userAssignment.setMinutesForAttempt(defaultAssignmentRestriction.getMinutesForAttempt());
        }
    }
}

```

```

        allUserAssignments.add(userAssignment);
    }

    userAssignmentRepository.saveAll(allUserAssignments);
}

@Override
public void startContinue(int userAssignmentId) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId))
    );

    if (!sessionService.isUserStudent(sessionService.getCurrentUser()) ||
        !permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
            userAssignment)) {
        throw new PermissionException();
    }

    userAssignment.setStatus(AssignmentStatus.ACTIVE);
    userAssignmentRepository.save(userAssignment);
}

@Override
public void finish(int userAssignmentId) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId))
    );

    if (!sessionService.isUserStudent(sessionService.getCurrentUser()) ||
        !permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
            userAssignment)) {
        throw new PermissionException();
    }

    userAssignment.setStatus(AssignmentStatus.FINISHED);
    userAssignmentRepository.save(userAssignment);
}

@Override
public AssignmentResponseDto answerAssignment(int userAssignmentId,
        AssignmentAnswerDto dto) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId)));

    if (!sessionService.isUserStudent(sessionService.getCurrentUser()) ||
        !permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
            userAssignment)) {
        throw new PermissionException();
    }

    if (userAssignment.getRestrictionType().equals(AssignmentRestrictionType.N_ATTEMPTS) &&
        userAssignment.getAttemptsRemaining() == 0) {
        throw new AttemptsNotLeftException();
    } else if (userAssignment.getRestrictionType().equals(AssignmentRestrictionType.DEADLINE) &&
        userAssignment.getDeadline().isBefore(LocalDateTime.now())) {
        throw new AttemptAfterDeadlineException(userAssignment.getDeadline());
    } else if (userAssignment.getRestrictionType().equals(AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES) &&
        (userAssignment.getLastAttemptTime() != null &&
            userAssignment.getLastAttemptTime().plusMinutes(userAssignment.getMinutesForAttempt())
                .isAfter(LocalDateTime.now())))
    ) {
        throw new AttemptLimitReachedException(userAssignment.getLastAttemptTime()
            .plusMinutes(userAssignment.getMinutesForAttempt()));
    }

    if (userAssignment.isHasCorrectAnswer()) {
        throw new AlreadyCorrectAnswerException();
    }

    double result;
    try {
        Map<String, Double> answerVariables = AnswerParser.parseAnswer(dto.answer());
        result = ExpressionParser.parse(userAssignment.getFunction().getText(), answerVariables);
    } catch (RuntimeException e) {

```

```

        throw new AnswerFormatIncorrectException();
    }

    Function function = userAssignment.getFunction();
    FunctionResultType functionResultType = userAssignment.getAssignment().getFunctionResultType();

    List<Double> correctValues = getMinMaxValuesForFunction(function, functionResultType);

    Answer answer = new Answer();
    answer.setCorrect(false);
    userAssignment.setHasCorrectAnswer(false);

    for (Double correctValue : correctValues) {
        if (correctValue == result) {
            answer.setCorrect(true);
            userAssignment.setHasCorrectAnswer(true);
            break;
        }
    }

    if (userAssignment.getRestrictionType().equals(AssignmentRestrictionType.N_ATTEMPTS))
        userAssignment.setAttemptsRemaining(userAssignment.getAttemptsRemaining() - 1);
    else if (userAssignment.getRestrictionType()
        .equals(AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES))
        userAssignment.setLastAttemptTime(LocalDateTime.now());

    answer.setUserAssignment(userAssignment);
    answer.setAnswer(dto.answer());
    answer.setResult(result);

    List<Answer> answers = userAssignment.getAnswers();
    answer.setAnswerNumber(answers.size() + 1);

    userAssignmentRepository.save(userAssignment);
    answerRepository.save(answer);

    return new AssignmentResponseDto(result, answer.isCorrect(),
        userAssignment.getRestrictionType().ordinal(),
        userAssignment.getAttemptsRemaining(),
        userAssignment.getDeadline(),
        LocalDateTime.now().plusMinutes(userAssignment.getMinutesForAttempt())
    );
}

@Override
public List<AnswerDto> getAnswersForAssignment(int userAssignmentId) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId)));

    if (!permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
        userAssignment)) {
        throw new PermissionException();
    }

    List<Answer> answers = userAssignment.getAnswers();
    List<AnswerDto> answerDtos = answers.stream()
        .sorted(Comparator.comparingInt(Answer::getAnswerNumber).reversed())
        .map(a1 -> new AnswerDto(a1.getAnswerNumber(), a1.getAnswer(),
            a1.getResult(), a1.isCorrect()))
        .toList();

    return answerDtos;
}

private Set<UserAssignment> getByUser(User user) {
    Set<UserAssignment> userAssignments = new HashSet<>();
    List<UserPermission> userPermissions = user.getUserPermissions();

    for (UserPermission userPermission : userPermissions) {
        if (userPermission.getUserAssignment() != null) {
            userAssignments.add(userPermission.getUserAssignment());
        } else if (userPermission.getFunction() != null) {
            userAssignments.addAll(userPermission.getFunction().getUserAssignments());
        } else if (userPermission.getSubject() != null) {
            userAssignments.addAll(
                userPermission.getSubject().getFunctions().stream()
                    .flatMap(f -> f.getUserAssignments().stream())
            );
        }
    }
}

```

```

        .toList()
    );
} else if (userPermission.getUniversity() != null) {
    userAssignments.addAll(
        userPermission.getUniversity().getSubjects().stream()
            .flatMap(s -> s.getFunctions().stream())
            .flatMap(f -> f.getUserAssignments().stream())
            .toList()
    );
}
}
return userAssignments;
}

private List<Double> getMinMaxValuesForFunction(Function function,
        FunctionResultType functionResultType) {
    return function.getFunctionMinMaxValues().stream()
        .filter(fmmv -> fmmv.getFunctionResultType() == functionResultType)
        .map(FunctionMinMaxValue::getValue)
        .toList();
}

private UserDto getUserDto(User user) {
    return new UserDto(user.getId(), user.getUserInfo().getFirstName(),
        user.getUserInfo().getLastName(),
        user.getEmail(),
        user.getRoles().get(0).getName().name(),
        user.isApproved(),
        new UniversityDto(user.getUserInfo().getUniversity().getId(),
            user.getUserInfo().getUniversity().getName()));
}
}

```

Файл: org/apiapplication/services/implementations/FieldServiceImpl.java

```

@Service
public class FieldServiceImpl implements FieldService {
    private final UrlRepository urlRepository;

    private final SessionService sessionService;

    public FieldServiceImpl(UrlRepository urlRepository,
        SessionService sessionService) {
        this.urlRepository = urlRepository;

        this.sessionService = sessionService;
    }

    @Override
    public List<FieldDto> getByUrlId(Integer urlId) {
        if (sessionService.getCurrentUser() == null)
            throw new PermissionException();

        Url url = urlRepository.findById(urlId)
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.URL, urlId.toString()));

        List<FieldDto> fieldDtos = url.getUrlFields().stream()
            .map(f -> new FieldDto(f.getField().getId(),
                f.getField().getName(), f.getField().getLabel(),
                f.getField().getDescription(),
                f.getField().getType().ordinal(),
                f.isRequired(),
                f.isMultiple()))
            .toList();

        return fieldDtos;
    }
}

```

Файл: org/apiapplication/services/implementations/FunctionServiceImpl.java

```

@Service
@Transactional
public class FunctionServiceImpl implements FunctionService {
    private final FunctionRepository functionRepository;
    private final SubjectRepository subjectRepository;
    private final FunctionMinMaxValueRepository functionMinMaxValueRepository;
    private final UserPermissionRepository userPermissionRepository;
    private final UserAssignmentRepository userAssignmentRepository;
}

```

```

private final PermissionService permissionService;
private final SessionService sessionService;

public FunctionServiceImpl(FunctionRepository functionRepository,
    SubjectRepository subjectRepository,
    FunctionMinMaxValueRepository functionMinMaxValueRepository,
    UserPermissionRepository userPermissionRepository,
    UserAssignmentRepository userAssignmentRepository,
    PermissionService permissionService,
    SessionService sessionService) {
    this.functionRepository = functionRepository;
    this.subjectRepository = subjectRepository;
    this.functionMinMaxValueRepository = functionMinMaxValueRepository;
    this.userPermissionRepository = userPermissionRepository;
    this.userAssignmentRepository = userAssignmentRepository;

    this.permissionService = permissionService;
    this.sessionService = sessionService;
}

@Override
public FunctionDto getFunctionById(int id) {
    return getFunctionDtoFromFunction(List.of(
        functionRepository.findById(id).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.FUNCTION, String.valueOf(id)))
    )).get(0);
}

@Override
public List<AssignmentFunctionDto> getFunctionsByAssignmentIds(List<Integer> assignmentIds) {
    List<UserAssignment> userAssignments = userAssignmentRepository.findAllById(assignmentIds);
    List<Integer> functionIds = userAssignments.stream()
        .map(ua -> ua.getFunction().getId())
        .distinct().toList();
    List<Function> functions = functionRepository.findAllById(functionIds);

    List<AssignmentFunctionDto> assignmentFunctionDtos = new ArrayList<>();
    for (UserAssignment userAssignment : userAssignments) {
        Function function = userAssignment.getFunction();
        AssignmentFunctionDto assignmentFunctionDto =
            new AssignmentFunctionDto(getFunctionDtoFromFunction(List.of(function)).get(0),
                userAssignment.getId());
        assignmentFunctionDtos.add(assignmentFunctionDto);
    }

    return assignmentFunctionDtos;
}

@Override
public List<FunctionDto> get(Integer subjectId) {
    List<Function> functions;
    if (subjectId != null) {
        Subject subject = subjectRepository.findById(subjectId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.SUBJECT,
                String.valueOf(subjectId))
        );

        if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(), subject)) {
            functions = getByUser(sessionService.getCurrentUser()).stream()
                .filter(f -> f.getSubject().getId().equals(subjectId))
                .toList();
        } else {
            functions = subject.getFunctions();
        }
    } else {
        functions = getByUser(sessionService.getCurrentUser()).stream().toList();
    }

    return getFunctionDtoFromFunction(functions);
}

@Override
public IdDto add(AddFunctionDto dto) {
    Optional<Function> existingFunction = functionRepository.findAll().stream()
        .filter(u -> u.getText().equalsIgnoreCase(dto.text()))
        .findFirst();

    if (existingFunction.isPresent()) {

```

```

        throw new EntityWithNameAlreadyFoundException(EntityName.FUNCTION,
            dto.text());
    }

    Subject subject = subjectRepository
        .findById(dto.subjectId())
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.FUNCTION,
            String.valueOf(dto.subjectId())));

    if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
        subject)) {
        throw new PermissionException();
    }

    checkFunctionText(dto.text(), dto.variablesCount());

    Function function = new Function();
    function.setText(dto.text());

    List<FunctionMinMaxValue> functionMinMaxValues = new ArrayList<>();
    for (double minValue : dto.minValues()) {
        FunctionMinMaxValue functionMinMaxValue = new FunctionMinMaxValue();
        functionMinMaxValue.setFunctionResultType(FunctionResultType.MIN);
        functionMinMaxValue.setValue(minValue);
        functionMinMaxValue.setFunction(function);

        functionMinMaxValues.add(functionMinMaxValue);
    }

    for (double maxValue : dto.maxValues()) {
        FunctionMinMaxValue functionMinMaxValue = new FunctionMinMaxValue();
        functionMinMaxValue.setFunctionResultType(FunctionResultType.MAX);
        functionMinMaxValue.setValue(maxValue);
        functionMinMaxValue.setFunction(function);

        functionMinMaxValues.add(functionMinMaxValue);
    }

    function.setFunctionMinMaxValues(functionMinMaxValues);
    function.setVariablesCount(dto.variablesCount());
    function.setSubject(subject);

    functionMinMaxValueRepository.saveAll(functionMinMaxValues);
    functionRepository.save(function);

    return new IdDto(function.getId());
}

@Override
public void update(int id, UpdateFunctionDto dto) {
    Function existingFunction = functionRepository.findById(id)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.FUNCTION,
            String.valueOf(id)));

    if (!permissionService.userCanAccessFunction(sessionService.getCurrentUser(),
        existingFunction)) {
        throw new PermissionException();
    }

    if (dto.subjectId() != null) {
        Subject subject = subjectRepository
            .findById(dto.subjectId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.SUBJECT,
                String.valueOf(dto.subjectId())));

        if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
            subject)) {
            throw new PermissionException();
        }

        existingFunction.setSubject(subject);
    }

    if (dto.text() != null && !dto.text().isEmpty()) {
        checkFunctionText(dto.text(), dto.variablesCount() != null ? dto.variablesCount()
            : existingFunction.getVariablesCount());

        Optional<Function> functionWithSameText = functionRepository.findAll()

```

```

        .stream()
        .filter(f -> f.getText().equalsIgnoreCase(dto.text())
            && !f.getId().equals(id))
        .findFirst();

    if (functionWithSameText.isPresent()) {
        throw new FunctionSameExistsException();
    }

    existingFunction.setText(dto.text());
}

List<FunctionMinMaxValue> functionMinMaxValues = existingFunction.getFunctionMinMaxValues();
List<FunctionMinMaxValue> minMaxValuesToAdd = new ArrayList<>();
List<FunctionMinMaxValue> minMaxValuesToRemove = new ArrayList<>();

if (!dto.minValues().isEmpty()) {
    List<Double> minValues = new ArrayList<>()
        functionMinMaxValues.stream()
            .filter(mv ->
                mv.getFunctionResultType().equals(FunctionResultType.MIN))
            .map(FunctionMinMaxValue::getValue)
            .toList();

    for (double minValue : dto.minValues()) {
        if (minValues.contains(minValue)) {
            continue;
        }

        FunctionMinMaxValue functionMinMaxValue = new FunctionMinMaxValue();
        functionMinMaxValue.setFunctionResultType(FunctionResultType.MIN);
        functionMinMaxValue.setValue(minValue);
        functionMinMaxValue.setFunction(existingFunction);

        minMaxValuesToAdd.add(functionMinMaxValue);
        minValues.add(minValue);
    }

    List<FunctionMinMaxValue> functionMinValues = functionMinMaxValues.stream()
        .filter(mv ->
            mv.getFunctionResultType().equals(FunctionResultType.MIN))
        .toList();

    for (FunctionMinMaxValue fmmv : functionMinValues) {
        if (!dto.minValues().contains(fmmv.getValue())) {
            minMaxValuesToRemove.add(fmmv);
        }
    }
} else {
    List<FunctionMinMaxValue> functionMaxValues = functionMinMaxValues.stream()
        .filter(mv ->
            mv.getFunctionResultType().equals(FunctionResultType.MAX))
        .toList();

    minMaxValuesToRemove.addAll(functionMaxValues);
}

if (!dto.maxValues().isEmpty()) {
    List<Double> maxValues = new ArrayList<>()
        functionMinMaxValues.stream()
            .filter(mv ->
                mv.getFunctionResultType().equals(FunctionResultType.MAX))
            .map(FunctionMinMaxValue::getValue)
            .toList();

    for (double maxValue : dto.maxValues()) {
        if (maxValues.contains(maxValue)) {
            continue;
        }

        FunctionMinMaxValue functionMinMaxValue = new FunctionMinMaxValue();
        functionMinMaxValue.setFunctionResultType(FunctionResultType.MAX);
        functionMinMaxValue.setValue(maxValue);
        functionMinMaxValue.setFunction(existingFunction);

        minMaxValuesToAdd.add(functionMinMaxValue);
        maxValues.add(maxValue);
    }
}

```

```

List<FunctionMinMaxValue> functionMaxValues = functionMinMaxValues.stream()
    .filter(mv ->
        mv.getFunctionResultType().equals(FunctionResultType.MAX))
    .toList();

for (FunctionMinMaxValue fmmv : functionMaxValues) {
    if (!dto.maxValues().contains(fmmv.getValue())) {
        minMaxValuesToRemove.add(fmmv);
    }
}
} else {
    List<FunctionMinMaxValue> functionMaxValues = functionMinMaxValues.stream()
        .filter(mv ->
            mv.getFunctionResultType().equals(FunctionResultType.MAX))
        .toList();

    minMaxValuesToRemove.addAll(functionMaxValues);
}

if (dto.variablesCount() != null) {
    existingFunction.setVariablesCount(dto.variablesCount());
}

functionMinMaxValueRepository.deleteAll(minMaxValuesToRemove);
functionMinMaxValueRepository.saveAll(minMaxValuesToAdd);
functionRepository.save(existingFunction);
}

@Override
public void delete(int functionId) {
    Function function = functionRepository.findById(functionId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(
            EntityName.FUNCTION, String.valueOf(functionId)
        ));

    if (!permissionService.userCanAccessFunction(sessionService.getCurrentUser(),
        function)) {
        throw new PermissionException();
    }

    if (!function.getUserAssignments().isEmpty()) {
        throw new EntityCantBeDeletedException();
    }

    userPermissionRepository.deleteAll(function.getUserPermissions());
    functionRepository.delete(function);
}

private List<FunctionDto> getFunctionDtoFromFunction(List<Function> functions) {
    return functions.stream()
        .map(f -> {
            Subject subject = f.getSubject();
            University university = subject.getUniversity();
            UniversityDto universityDto = new UniversityDto(university.getId(), university.getName());
            SubjectDto subjectDto = new SubjectDto(subject.getId(), subject.getName(), universityDto);

            return new FunctionDto(f.getId(), f.getText(), f.getVariablesCount(),
                getMinMaxValuesForFunction(f, FunctionResultType.MIN),
                getMinMaxValuesForFunction(f, FunctionResultType.MAX),
                subjectDto);
        }).toList();
}

private List<Double> getMinMaxValuesForFunction(Function function,
    FunctionResultType functionResultType) {
    return function.getFunctionMinMaxValues().stream()
        .filter(fmmv -> fmmv.getFunctionResultType() == functionResultType)
        .map(FunctionMinMaxValue::getValue)
        .toList();
}

private Set<Function> getByUser(User user) {
    Set<Function> functions = new HashSet<>();
    List<UserPermission> userPermissions = user.getUserPermissions();

    for (UserPermission userPermission : userPermissions) {
        if (userPermission.getFunction() != null) {

```

```

        functions.add(userPermission.getFunction());
    } else if (userPermission.getSubject() != null) {
        functions.addAll(
            userPermission.getSubject().getFunctions().stream()
                .toList()
        );
    } else if (userPermission.getUniversity() != null) {
        functions.addAll(
            userPermission.getUniversity().getSubjects().stream()
                .flatMap(s -> s.getFunctions().stream())
                .toList()
        );
    }
}
return functions;
}

private void checkFunctionText(String text, int variablesCount) {
    try {
        Map<String, Double> answerVariables = new HashMap<>();
        for (int i = 0; i < variablesCount; i++) {
            answerVariables.put("x" + (i + 1), 0D);
        }
        ExpressionParser.parse(text, answerVariables);
    } catch (RuntimeException e) {
        throw new FunctionTextFormatIncorrectException();
    }
}
}
}

```

Файл: org/apiapplication/services/implementations/GroupServiceImpl.java

```

@Service
@Transactional
public class GroupServiceImpl implements GroupService {
    private final GroupRepository groupRepository;
    private final UserRepository userRepository;
    private final SubjectRepository subjectRepository;

    private final SessionService sessionService;

    public GroupServiceImpl(GroupRepository groupRepository,
        UserRepository userRepository,
        SubjectRepository subjectRepository,
        SessionService sessionService) {
        this.groupRepository = groupRepository;
        this.userRepository = userRepository;
        this.subjectRepository = subjectRepository;

        this.sessionService = sessionService;
    }

    @Override
    public List<GroupDto> get() {
        User currentUser = sessionService.getCurrentUser();
        if (!sessionService.isUserTeacher(currentUser)) {
            throw new PermissionException();
        }

        return currentUser.getGroups().stream()
            .map(this::getGroupDto)
            .toList();
    }

    @Override
    public GroupDto getById(int id) {
        User currentUser = sessionService.getCurrentUser();
        if (!sessionService.isUserTeacher(currentUser))
            throw new PermissionException();

        Group group = groupRepository.findById(id)
            .orElseThrow(() -> new EntityWithIdNotFoundException(
                EntityName.GROUP, String.valueOf(id)
            ));

        if (!group.getOwner().getId().equals(currentUser.getId()))
            throw new PermissionException();
    }
}

```

```

    return getGroupDto(group);
}

@Override
public IdDto add(AddGroupDto dto) {
    User currentUser = sessionService.getCurrentUser();
    if (!sessionService.isUserTeacher(currentUser))
        throw new PermissionException();

    Optional<Group> existingGroup = groupRepository.findAll().stream()
        .filter(g -> g.getName().equalsIgnoreCase(dto.name()))
        .findFirst();

    if (existingGroup.isPresent())
        throw new EntityWithNameAlreadyFoundException(EntityName.GROUP, dto.name());

    Group group = new Group();
    group.setName(dto.name());
    group.setOwner(currentUser);

    List<Subject> subjects = subjectRepository.findAllById(dto.subjectIds());
    group.setSubjects(new HashSet<>(subjects));

    List<User> users = userRepository.findAllById(dto.userIds());
    group.setStudents(new HashSet<>(users));

    groupRepository.save(group);

    return new IdDto(group.getId());
}

@Override
public void update(int id, UpdateGroupDto dto) {
    Group group = groupRepository.findById(id)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,
            String.valueOf(id)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    Optional<Group> existingGroup = groupRepository.findAll().stream()
        .filter(g -> g.getName().equalsIgnoreCase(dto.name()) &&
            !g.getId().equals(id))
        .findFirst();

    if (existingGroup.isPresent())
        throw new EntityWithNameAlreadyFoundException(EntityName.GROUP, dto.name());

    if (dto.name() != null && !dto.name().isEmpty())
        group.setName(dto.name());

    List<Subject> subjects = subjectRepository.findAllById(dto.subjectIds());
    group.getSubjects().clear();
    group.getSubjects().addAll(new HashSet<>(subjects));

    List<User> users = userRepository.findAllById(dto.userIds());
    group.getStudents().clear();
    group.getStudents().addAll(new HashSet<>(users));

    groupRepository.save(group);
}

@Override
public void delete(int id) {
    Group group = groupRepository.findById(id)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,
            String.valueOf(id)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    groupRepository.delete(group);
}

@Override
public void addStudents(int groupId, SetStudentsDto dto) {
    Group group = groupRepository.findById(groupId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,

```

```

        String.valueOf(groupId)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    List<User> users = userRepository.findAllById(dto.userIds());
    group.getStudents().addAll(users);
    groupRepository.save(group);
}

@Override
public void removeStudents(int groupId, SetStudentsDto dto) {
    Group group = groupRepository.findById(groupId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,
            String.valueOf(groupId)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    List<User> users = userRepository.findAllById(dto.userIds());
    users.forEach(group.getStudents()::remove);
    groupRepository.save(group);
}

@Override
public void addSubjects(int groupId, SetSubjectsDto dto) {
    Group group = groupRepository.findById(groupId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,
            String.valueOf(groupId)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    List<Subject> subjects = subjectRepository.findAllById(dto.subjectIds());
    group.getSubjects().addAll(subjects);
    groupRepository.save(group);
}

@Override
public void removeSubjects(int groupId, SetSubjectsDto dto) {
    Group group = groupRepository.findById(groupId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.GROUP,
            String.valueOf(groupId)));

    if (!Objects.equals(sessionService.getCurrentUser().getId(), group.getOwner().getId()))
        throw new PermissionException();

    List<Subject> subjects = subjectRepository.findAllById(dto.subjectIds());
    subjects.forEach(group.getSubjects()::remove);
    groupRepository.save(group);
}

private GroupDto getGroupDto(Group group) {
    return new GroupDto(
        group.getId(),
        group.getName(),
        group.getStudents().stream()
            .map(s -> new GroupStudentDto(s.getId(),
                s.getUserInfo().getFirstName(),
                s.getUserInfo().getLastName()))
            .toList(),
        group.getSubjects().stream()
            .map(s -> new GroupSubjectDto(s.getId(),
                s.getName()))
            .toList());
}
}

```

Файл: org/apiapplication/services/implementations/MarkServiceImpl.java

```

@Service
@Transactional
public class MarkServiceImpl implements MarkService {
    private final MarkRepository markRepository;
    private final UserAssignmentRepository userAssignmentRepository;

    private final PermissionService permissionService;
    private final SessionService sessionService;
}

```

```

public MarkServiceImpl(UserAssignmentRepository userAssignmentRepository,
    MarkRepository markRepository,
    PermissionService permissionService,
    SessionService sessionService) {
    this.userAssignmentRepository = userAssignmentRepository;
    this.markRepository = markRepository;

    this.permissionService = permissionService;
    this.sessionService = sessionService;
}

@Override
public void markAssignment(int userAssignmentId, MarkDto dto) {
    Mark mark;
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId))
    );

    if (!sessionService.isUserTeacher(sessionService.getCurrentUser()) ||
        !permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
            userAssignment)) {
        throw new PermissionException();
    }

    if (dto.id() != null) {
        mark = markRepository.findById(dto.id()).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.MARK,
                String.valueOf(dto.id()))
        );
        mark.setMark(dto.mark());
        mark.setComment(dto.comment());
    } else {
        mark = new Mark();
        mark.setMark(dto.mark());
        mark.setComment(dto.comment());
    }

    mark.setUserAssignment(userAssignment);
}

markRepository.save(mark);
}

@Override
public List<MarkDto> getByUserAssignmentId(int userAssignmentId) {
    UserAssignment userAssignment = userAssignmentRepository.findById(userAssignmentId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
            String.valueOf(userAssignmentId))
    );

    if (!permissionService.userCanAccessAssignment(sessionService.getCurrentUser(),
        userAssignment)) {
        throw new PermissionException();
    }

    List<MarkDto> dtos = userAssignment.getMarks().stream()
        .map(m -> new MarkDto(m.getId(), m.getMark(), m.getComment()))
        .toList();

    return dtos;
}
}

```

Файл: org/apiapplication/services/implementations/PermissionServiceImpl.java

```

@Service
@Transactional
public class PermissionServiceImpl implements PermissionService {
    private final UserPermissionRepository userPermissionRepository;
    private final UserRepository userRepository;
    private final UniversityRepository universityRepository;
    private final SubjectRepository subjectRepository;
    private final UserAssignmentRepository userAssignmentRepository;
    private final FunctionRepository functionRepository;

    private final SessionService sessionService;

    public PermissionServiceImpl(UserPermissionRepository userPermissionRepository,
        UserRepository userRepository,

```

```

        UniversityRepository universityRepository,
        SubjectRepository subjectRepository,
        UserAssignmentRepository userAssignmentRepository,
        FunctionRepository functionRepository,
        SessionService sessionService) {
    this.userPermissionRepository = userPermissionRepository;
    this.userRepository = userRepository;
    this.universityRepository = universityRepository;
    this.subjectRepository = subjectRepository;
    this.userAssignmentRepository = userAssignmentRepository;
    this.functionRepository = functionRepository;

    this.sessionService = sessionService;
}

@Override
public List<PermissionDto> get(Integer userId) {
    if (userId != null) {
        User user = userRepository.findById(userId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER, String.valueOf(userId))
        );

        if (sessionService.isCurrentUser(userId) ||
            sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            List<UserPermission> userPermissions = user.getUserPermissions();
            return userPermissions.stream()
                .map(this::getPermissionDto)
                .toList();
        } else {
            return List.of();
        }
    } else {
        if (sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            List<UserPermission> userPermissions = userPermissionRepository.findAll();
            return userPermissions.stream().map(this::getPermissionDto).toList();
        } else {
            User user = sessionService.getCurrentUser();
            List<UserPermission> userPermissions = user.getUserPermissions();
            return userPermissions.stream().map(this::getPermissionDto).toList();
        }
    }
}

@Override
public boolean userCanAccessAssignment(User user, UserAssignment userAssignment) {
    if (userAssignment.getUser().getId().equals(user.getId())) {
        return true;
    }

    List<UserPermission> userPermissions = user.getUserPermissions();

    Optional<UserPermission> userPermission = userPermissions.stream()
        .filter(up -> up.getUserAssignment() != null)
        .filter(up -> up.getUserAssignment().getId().equals(userAssignment.getId()))
        .findFirst();

    if (userPermission.isPresent()) {
        return true;
    }

    Function function = userAssignment.getFunction();

    return userCanAccessFunction(user, function);
}

@Override
public boolean userCanAccessFunction(User user, Function function) {
    List<UserPermission> userPermissions = user.getUserPermissions();

    Optional<UserPermission> userPermission = userPermissions.stream()
        .filter(up -> up.getFunction() != null)
        .filter(up -> up.getFunction().getId().equals(function.getId()))
        .findFirst();

    if (userPermission.isPresent()) {
        return true;
    }
}

```

```

        Subject subject = function.getSubject();

        return userCanAccessSubject(user, subject);
    }

    @Override
    public boolean userCanAccessSubject(User user, Subject subject) {
        List<UserPermission> userPermissions = user.getUserPermissions();

        Optional<UserPermission> userPermission = userPermissions.stream()
            .filter(up -> up.getSubject() != null)
            .filter(up -> up.getSubject().getId().equals(subject.getId()))
            .findFirst();

        if (userPermission.isPresent()) {
            return true;
        }

        University university = subject.getUniversity();

        return userCanAccessUniversity(user, university);
    }

    @Override
    public boolean userCanAccessUniversity(User user, University university) {
        List<UserPermission> userPermissions = user.getUserPermissions();

        Optional<UserPermission> userPermission = userPermissions.stream()
            .filter(up -> up.getUniversity() != null)
            .filter(up -> up.getUniversity().getId().equals(university.getId()))
            .findFirst();

        return userPermission.isPresent();
    }

    @Override
    public void updatePermissions(UpdatePermissionDto dto) {
        if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            throw new PermissionException();
        }

        User user = userRepository.findById(dto.userId()).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER,
                String.valueOf(dto.userId()))
        );

        if (sessionService.isUserStudent(user)) {
            throw new PermissionException();
        }

        List<UserPermission> userPermissionsToAdd = getUserPermissionsToAdd(dto, user);
        List<UserPermission> userPermissionsToRemove = new ArrayList<>();

        user.getUserPermissions().stream()
            .filter(up -> up.getUniversity() != null)
            .filter(up -> !dto.universityIds().contains(up.getUniversity().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getSubject() != null)
            .filter(up -> !dto.subjectIds().contains(up.getSubject().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getFunction() != null)
            .filter(up -> !dto.functionIds().contains(up.getFunction().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getUserAssignment() != null)
            .filter(up -> !dto.userAssignmentIds().contains(
                up.getUserAssignment().getId()
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);
    }

```

```

        userPermissionRepository.deleteAll(userPermissionsToRemove);
        userPermissionRepository.saveAll(userPermissionsToAdd);
    }

    @Override
    public void removePermissions(UpdatePermissionDto dto) {
        if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            throw new PermissionException();
        }

        User user = userRepository.findById(dto.userId()).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER,
                String.valueOf(dto.userId()))
        );

        List<UserPermission> userPermissionsToRemove = new ArrayList<>();
        user.getUserPermissions().stream()
            .filter(up -> up.getUniversity() != null)
            .filter(up -> dto.universityIds().contains(up.getUniversity().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getSubject() != null)
            .filter(up -> dto.subjectIds().contains(up.getSubject().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getFunction() != null)
            .filter(up -> dto.functionIds().contains(up.getFunction().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        user.getUserPermissions().stream()
            .filter(up -> up.getUserAssignment() != null)
            .filter(up -> dto.userAssignmentIds().contains(
                up.getUserAssignment().getId())
                && up.getUser().getId().equals(user.getId()))
            .forEach(userPermissionsToRemove::add);

        userPermissionRepository.deleteAll(userPermissionsToRemove);
    }

    @Override
    public void givePermissions(UpdatePermissionDto dto) {
        if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            throw new PermissionException();
        }

        User user = userRepository.findById(dto.userId()).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER,
                String.valueOf(dto.userId()))
        );

        List<UserPermission> userPermissionsToAdd = getUserPermissionsToAdd(dto, user);

        userPermissionRepository.saveAll(userPermissionsToAdd);
    }

    private List<UserPermission> getUserPermissionsToAdd(UpdatePermissionDto dto,
        User user) {
        List<UserPermission> userPermissionsToAdd = new ArrayList<>();
        for (int universityId : dto.universityIds()) {
            if (user.getRoles().get(0).getName().equals(UserRole.TEACHER)) {
                throw new PermissionException();
            }

            University university = universityRepository.findById(universityId).orElseThrow(
                () -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
                    String.valueOf(universityId))
            );

            UserPermission userPermission = new UserPermission();
            userPermission.setUser(user);
            userPermission.setUniversity(university);

            if (!user.getUserPermissions().contains(userPermission)) {

```

```

        userPermissionsToAdd.add(userPermission);
    }
}

for (int subjectId : dto.subjectIds()) {
    Subject subject = subjectRepository.findById(subjectId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.SUBJECT,
            String.valueOf(subjectId))
    );

    UserPermission userPermission = new UserPermission();
    userPermission.setUser(user);
    userPermission.setSubject(subject);

    if (!user.getUserPermissions().contains(userPermission)) {
        userPermissionsToAdd.add(userPermission);
    }
}

for (int functionId : dto.functionIds()) {
    Function function = functionRepository.findById(functionId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.FUNCTION,
            String.valueOf(functionId))
    );

    UserPermission userPermission = new UserPermission();
    userPermission.setUser(user);
    userPermission.setFunction(function);

    if (!user.getUserPermissions().contains(userPermission)) {
        userPermissionsToAdd.add(userPermission);
    }
}

for (int assignmentId : dto.userAssignmentIds()) {
    UserAssignment userAssignment = userAssignmentRepository.findById(assignmentId)
        .orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER_ASSIGNMENT,
                String.valueOf(assignmentId))
        );

    UserPermission userPermission = new UserPermission();
    userPermission.setUser(user);
    userPermission.setUserAssignment(userAssignment);

    if (!user.getUserPermissions().contains(userPermission)) {
        userPermissionsToAdd.add(userPermission);
    }
}

return userPermissionsToAdd;
}

private PermissionDto getPermissionDto(UserPermission userPermission) {
    return new PermissionDto(userPermission.getId(), userPermission.getUser().getId(),
        userPermission.getUniversity() != null ? userPermission.getUniversity().getId() : null,
        userPermission.getSubject() != null ? userPermission.getSubject().getId() : null,
        userPermission.getFunction() != null ? userPermission.getFunction().getId() : null,
        userPermission.getUserAssignment() != null ? userPermission.getUserAssignment().getId()
            : null);
}
}

```

Файл: org/apiapplication/services/implementations/SubjectServiceImpl.java

```

@Service
@Transactional
public class SubjectServiceImpl implements SubjectService {
    private final SubjectRepository subjectRepository;
    private final UniversityRepository universityRepository;
    private final UserPermissionRepository userPermissionRepository;

    private final SessionService sessionService;
    private final PermissionService permissionService;

    public SubjectServiceImpl(SubjectRepository subjectRepository,
        UniversityRepository universityRepository,
        UserPermissionRepository userPermissionRepository,
        SessionService sessionService,

```

```

        PermissionService permissionService) {
    this.subjectRepository = subjectRepository;
    this.universityRepository = universityRepository;
    this.userPermissionRepository = userPermissionRepository;
    this.sessionService = sessionService;
    this.permissionService = permissionService;
}

@Override
public SubjectDto getSubjectById(int id) {
    return getSubjectDtoFromSubjects(List.of(subjectRepository.findById(id).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.SUBJECT, String.valueOf(id))
    ))).get(0);
}

@Override
public List<SubjectDto> get(Integer universityId) {
    User user = sessionService.getCurrentUser();
    List<Subject> subjects = getForUser(user).stream().toList();

    if (universityId != null) {
        University university = universityRepository.findById(universityId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
                String.valueOf(universityId)));

        if (!permissionService.userCanAccessUniversity(user, university)) {
            throw new PermissionException();
        }

        subjects = subjects.stream()
            .filter(s -> s.getUniversity().equals(university))
            .toList();
    }

    return getSubjectDtoFromSubjects(subjects);
}

@Override
public IdDto add(AddSubjectDto dto) {
    University university = universityRepository
        .findById(dto.universityId())
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
            String.valueOf(dto.universityId())));

    if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
        throw new PermissionException();
    }

    Optional<Subject> existingSubject = subjectRepository.findAll().stream()
        .filter(u -> u.getName().equalsIgnoreCase(dto.name()))
        .findFirst();

    if (existingSubject.isPresent()) {
        throw new EntityWithNameAlreadyFoundException(EntityName.SUBJECT,
            dto.name());
    }

    Subject subject = new Subject();
    subject.setName(dto.name());
    subject.setUniversity(university);

    subjectRepository.save(subject);

    return new IdDto(subject.getId());
}

@Override
public void update(int id, UpdateSubjectDto dto) {
    Subject existingSubject = subjectRepository.findById(id)
        .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.SUBJECT,
            String.valueOf(id)));

    if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
        existingSubject)) {
        throw new PermissionException();
    }

    if (dto.universityId() != null) {

```

```

        University university = universityRepository
            .findById(dto.universityId())
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
                String.valueOf(dto.universityId())));

        if (!permissionService.userCanAccessUniversity(sessionService.getCurrentUser(),
            university)) {
            throw new PermissionException();
        }

        existingSubject.setUniversity(university);
    }

    if (dto.name() != null && !dto.name().isEmpty()) {
        Optional<Subject> subjectWithSameName = subjectRepository.findAll()
            .stream()
            .filter(s -> s.getName().equalsIgnoreCase(dto.name())
                && !s.getId().equals(id))
            .findFirst();

        if (subjectWithSameName.isPresent()) {
            throw new EntityWithNameAlreadyFoundException(EntityName.SUBJECT, dto.name());
        }

        existingSubject.setName(dto.name());
    }

    subjectRepository.save(existingSubject);
}

@Override
public void delete(int subjectId) {
    Subject subject = subjectRepository.findById(subjectId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(
            EntityName.SUBJECT, String.valueOf(subjectId)
        ));

    if (!permissionService.userCanAccessSubject(sessionService.getCurrentUser(),
        subject)) {
        throw new PermissionException();
    }

    if (!subject.getFunctions().isEmpty()) {
        throw new EntityCantBeDeletedException();
    }

    userPermissionRepository.deleteAll(subject.getUserPermissions());
    subjectRepository.delete(subject);
}

private List<SubjectDto> getSubjectDtoFromSubjects(List<Subject> subjects) {
    return subjects.stream()
        .map(s -> new SubjectDto(s.getId(), s.getName(), new UniversityDto(
            s.getUniversity().getId(), s.getUniversity().getName()
        )))
        .toList();
}

public Set<Subject> getUser(User user) {
    Set<Subject> subjects = new HashSet<>();
    List<UserPermission> userPermissions = user.getUserPermissions();

    if (user.getRoles().get(0).getName().equals(UserRole.STUDENT)) {
        subjects.addAll(user.getUserInfo().getUniversity().getSubjects());
    } else {
        for (UserPermission userPermission : userPermissions) {
            if (userPermission.getUniversity() != null) {
                subjects.addAll(userPermission.getUniversity().getSubjects());
            } else if (userPermission.getSubject() != null) {
                subjects.add(userPermission.getSubject());
            }
        }
    }

    return subjects;
}
}

```

Файл: org/apiapplication/services/implementations/UniversityServiceImpl.java

```

@Service
@Transactional
public class UniversityServiceImpl implements UniversityService {
    private final UniversityRepository universityRepository;
    private final UserPermissionRepository userPermissionRepository;

    private final SessionService sessionService;

    public UniversityServiceImpl(UniversityRepository universityRepository,
                                UserPermissionRepository userPermissionRepository,
                                SessionService sessionService) {
        this.universityRepository = universityRepository;
        this.userPermissionRepository = userPermissionRepository;
        this.sessionService = sessionService;
    }

    @Override
    public UniversityDto getUniversityById(int id) {
        University university = universityRepository.findById(id).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY, String.valueOf(id))
        );

        return new UniversityDto(university.getId(), university.getName());
    }

    @Override
    public List<UniversityDto> get() {
        Set<University> universities;
        User currentUser = sessionService.getCurrentUser();
        if (currentUser != null) {
            universities = getForUser(sessionService.getCurrentUser());
        } else {
            universities = new HashSet<>(universityRepository.findAll());
        }

        List<UniversityDto> universityDtos = universities.stream()
            .map(u -> new UniversityDto(u.getId(), u.getName()))
            .toList();

        return universityDtos;
    }

    @Override
    public IdDto add(AddUniversityDto dto) {
        if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            throw new PermissionException();
        }

        Optional<University> existingUniversity = universityRepository.findAll().stream()
            .filter(u -> u.getName().equalsIgnoreCase(dto.name()))
            .findFirst();

        if (existingUniversity.isPresent()) {
            throw new EntityWithNameAlreadyFoundException(EntityName.UNIVERSITY, dto.name());
        }

        University university = new University();
        university.setName(dto.name());

        universityRepository.save(university);

        return new IdDto(university.getId());
    }

    @Override
    public void update(int id, UpdateUniversityDto dto) {
        University existingUniversity = universityRepository.findById(id)
            .orElseThrow(() -> new EntityWithIdNotFoundException(EntityName.UNIVERSITY,
                String.valueOf(id)));

        if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
            throw new PermissionException();
        }

        if (dto.name() != null && !dto.name().isEmpty()) {
            Optional<University> universityWithSameName = universityRepository.findAll().stream()
                .filter(u -> u.getName().equalsIgnoreCase(dto.name()) &&
                    !u.getId().equals(id))
    
```

```

        .findFirst();

        if (universityWithSameName.isPresent()) {
            throw new EntityWithNameAlreadyFoundException(EntityName.UNIVERSITY,
                dto.name());
        }

        existingUniversity.setName(dto.name());
    }

    universityRepository.save(existingUniversity);
}

@Override
public void delete(int universityId) {
    University university = universityRepository.findById(universityId)
        .orElseThrow(() -> new EntityWithIdNotFoundException(
            EntityName.UNIVERSITY, String.valueOf(universityId)
        ));

    if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
        throw new PermissionException();
    }

    if (!university.getSubjects().isEmpty()) {
        throw new EntityCantBeDeletedException();
    }

    userPermissionRepository.deleteAll(university.getUserPermissions());
    universityRepository.delete(university);
}

public Set<University> getForUser(User user) {
    Set<University> universities = new HashSet<>();
    List<UserPermission> userPermissions = user.getUserPermissions();

    for (UserPermission userPermission : userPermissions) {
        if (userPermission.getUniversity() != null) {
            universities.add(userPermission.getUniversity());
        }
    }
    return universities;
}
}

```

Файл: org/apiapplication/services/implementations/UrlServiceImpl.java

```

@Service
public class UrlServiceImpl implements UrlService {
    private final UrlRepository urlRepository;

    private final SessionService sessionService;

    public UrlServiceImpl(UrlRepository urlRepository, SessionService sessionService) {
        this.urlRepository = urlRepository;
        this.sessionService = sessionService;
    }

    @Override
    public List<UrlDto> get(String url, Integer method) {
        User user = sessionService.getCurrentUser();
        if (user == null) {
            throw new PermissionException();
        }

        Role role = user.getRoles().get(0);

        if ((url == null || url.isEmpty()) && method == null) {
            List<UrlDto> urls = urlRepository.findAll().stream()
                .filter(u -> u.getRoles().contains(role))
                .map(this::getUrlDtoFromUrl)
                .toList();
            return urls;
        } else {
            List<Url> urls = urlRepository.findAll();
            Url neededUrl = null;

            urls = urls.stream()
                .sorted((u1, u2) -> Comparator.comparing(Url::getUrl).compare(u1, u2))

```

```

        .toList();

        for (Url urlToMatchWith : urls) {
            if (UrlMatcher.areMatched(urlToMatchWith.getUrl(), url)
                && urlToMatchWith.getMethod().ordinal() == method) {
                neededUrl = urlToMatchWith;
                break;
            }
        }

        if (neededUrl != null) {
            return List.of(getUrlDtoFromUrl(neededUrl));
        } else {
            throw new UrlWithNameNotFoundException(url,
                getMethods().stream().filter(m -> m.method() == method)
                    .findFirst().get().label());
        }
    }
}

@Override
public List<MethodTypeDto> getMethods() {
    if (sessionService.getCurrentUser() == null) {
        throw new PermissionException();
    }

    return Arrays.stream(MethodType.values())
        .map(mt -> new MethodTypeDto(mt.ordinal(), mt.name()))
        .toList();
}

private UrlDto getUrlDtoFromUrl(Url url) {
    return new UrlDto(url.getId(), url.getUrl(),
        url.getDescription(), url.getMethod().ordinal());
}
}

```

Файл: org/apiapplication/services/implementations/UserServiceImpl.java

```

@Service
@Transactional
public class UserServiceImpl implements UserService {
    private final UserInfoRepository userInfoRepository;
    private final UserRepository userRepository;
    private final RoleRepository roleRepository;

    private final SessionService sessionService;
    private final PermissionService permissionService;

    private final PasswordEncoder passwordEncoder;

    public UserServiceImpl(UserInfoRepository userInfoRepository,
        UserRepository userRepository,
        RoleRepository roleRepository,
        SessionService sessionService,
        PermissionService permissionService,
        PasswordEncoder passwordEncoder) {
        this.userInfoRepository = userInfoRepository;
        this.userRepository = userRepository;
        this.roleRepository = roleRepository;

        this.sessionService = sessionService;
        this.permissionService = permissionService;

        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public List<UserDto> get(Integer roleId) {
        User currentUser = sessionService.getCurrentUser();
        if (sessionService.isUserStudent(currentUser)) {
            return List.of(getUserDto(currentUser));
        }

        List<User> users;
        if (currentUser.getRoles().get(0).getName().equals(UserRole.TEACHER)) {
            users = currentUser.getUserInfo().getUniversity().getUserInfos()
                .stream()
                .map(UserInfo::getUser)

```

```

        .toList();
    } else {
        users = currentUser.getUserPermissions()
            .stream()
            .filter(up -> up.getUniversity() != null)
            .collect(ArrayList::new,
                (lst, up) ->
                    lst.addAll(up.getUniversity().getUserInfos()
                        .stream().map(UserInfo::getUser)
                        .toList()
                    ), ArrayList::addAll);
    }

    if (roleId != null) {
        return users.stream()
            .filter(u -> u.getRoles().get(0).getId().equals(roleId))
            .map(this::getUserDto)
            .toList();
    } else {
        return users.stream()
            .map(this::getUserDto)
            .toList();
    }
}

@Override
public UserDto getById(int userId) {
    User currentUser = sessionService.getCurrentUser();

    User user = userRepository.findById(userId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER, String.valueOf(userId))
    );

    if (!currentUser.getId().equals(userId) &&
        !permissionService.userCanAccessUniversity(currentUser,
            user.getUserInfo().getUniversity())) {
        throw new PermissionException();
    }

    return getUserDto(user);
}

@Override
public ApiKeyDto getApiKey(int userId) {
    UserInfo userInfo = userInfoRepository.findById(userId).orElseThrow(
        () -> new EntityWithIdNotFoundException(EntityName.USER, String.valueOf(userId))
    );

    if (!sessionService.isCurrentUser(userInfo.getId())) {
        throw new PermissionException();
    }

    String newApiToken = generateApiKey();
    userInfo.setApiKey(passwordEncoder.encode(newApiToken));

    userInfoRepository.save(userInfo);

    return new ApiKeyDto(newApiToken);
}

@Override
public void createAdmin(CreateAdminDto dto) {
    if (!sessionService.isUserAdmin(sessionService.getCurrentUser())) {
        throw new PermissionException();
    }

    Optional<User> optionalUser = userRepository.findByEmailIgnoreCase(dto.email());
    if (optionalUser.isPresent()) {
        throw new UserWithEmailExistsException(dto.email());
    }

    User user = new User();
    user.setEmail(dto.email());
    user.setPassword(passwordEncoder.encode(dto.password()));
    user.setApproved(true);

    UserInfo userInfo = new UserInfo();
    userInfo.setFirstName(dto.firstName());

```

```

        userInfo.setLastName(dto.lastName());
        userInfo.setApiKey(generateApiKey());
        userInfo.setUser(user);

        List<Role> roles = new ArrayList<>();
        Role userRole = roleRepository.findByName(UserRole.valueOf("ADMIN"))
            .orElseThrow(() -> new EntityWithNameNotFoundException(EntityName.ROLE, "ADMIN"));
        roles.add(userRole);
        user.setRoles(roles);

        userRepository.save(user);
        userInfoRepository.save(userInfo);
    }

    @Override
    public void approve(int userId) {
        User user = userRepository.findById(userId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER, String.valueOf(userId))
        );

        user.setApproved(true);
        userRepository.save(user);
    }

    @Override
    public void reject(int userId) {
        User user = userRepository.findById(userId).orElseThrow(
            () -> new EntityWithIdNotFoundException(EntityName.USER, String.valueOf(userId))
        );

        user.setApproved(false);
        userRepository.save(user);
    }

    private UserDto getUserDto(User user) {
        return new UserDto(user.getId(), user.getUserInfo().getFirstName(),
            user.getUserInfo().getLastName(),
            user.getEmail(),
            user.getRoles().get(0).getName().name(),
            user.isApproved(),
            new UniversityDto(user.getUserInfo().getUniversity().getId(),
                user.getUserInfo().getUniversity().getName()));
    }

    private String generateApiKey() {
        return UUID.randomUUID().toString();
    }
}

```

Файл: frontend-project/src/app/assignment/assignment.component.html

```

<div class="main-content">
  <h1 class="mb-4">
    <i class="fa-solid fa-clipboard-list me-2"></i> Виконання завдання
  </h1>

  <div class="card shadow-sm border-0 mb-4">
    <div class="card-body">

      <h5 class="card-title mb-3">Завдання</h5>
      <p class="card-text">
        {{assignmentDto?.hint}}
      </p>

      <p class="text-muted">
        @if(assignmentDto && isRestrictionNAttempts()){
          <i class="fa-solid fa-hourglass-half me-2"></i>
          Залишилось спроб:
          <span class="fw-bold">
            {{assignmentDto.attemptsRemaining}}
          </span>
        }
        @if(assignmentDto && isRestrictionDeadline()){
          <i class="fa-solid fa-calendar-alt me-2"></i>
          Дедлайн надсилання відповідей:
          <span class="fw-bold">
            {{assignmentDto.deadline | date: 'dd.MM.yyyy hh:mm:ss'}}
          </span>
        }
      </p>
    </div>
  </div>

```

```

@if(assignmentDto && isRestrictionAttemptInNMinutes()){
<i class="fa-solid fa-redo me-2"></i>
Наступна відповідь може бути надіслана:
<span class="fw-bold">
  @if(isNextAttemptTimeAfterNow()){
    {{assignmentDto.nextAttemptTime | date: 'dd.MM.yyyy hh:mm:ss'}}
  }
  @else {
    зараз
  }
</span>
}
</p>

<form [formGroup]="form">
<div class="row g-3 mb-3" formArrayName="variables">
  @for (i of variablesIndices; track i) {
    <div class="col-md-4">
      <label [for]="inputVariable' + (i + 1)" class="form-label">
        {{ 'x' + (i + 1) }}
      </label>

      <input type="text" class="form-control" [id]="inputVariable' + (i + 1)"
        [placeholder]="Введіть x' + (i + 1)" [formControlName]="i"
        [readonly]="disableInputsAndButtons() || isAnswersMode" />

      @if (variables.at(i).errors?.['required'] && (variables.at(i).touched || variables.at(i).dirty) &&
        !(disableInputsAndButtons() || isAnswersMode)) {
        <div class="text-danger mt-1">{{variablesErrorRequired}}</div>
      }
      @if (variables.at(i).errors?.['pattern'] && (variables.at(i).touched || variables.at(i).dirty) &&
        !(disableInputsAndButtons() || isAnswersMode)) {
        <div class="text-danger mt-1">{{variablesErrorNotNumber}}</div>
      }
    </div>
  }
</div>

</div>
</form>

@if(showAnswerNotCorrect()){
<div class="alert alert-danger">
  Ваша відповідь неправильна. Значення функції для ваших змінних: {{assignmentResponseDto.result}}
</div>
}
@if(showAnswerCorrect()){
<div class="alert alert-success">
  Ваша відповідь правильна. Значення функції для ваших змінних: {{assignmentResponseDto.result}}.
  Натисніть кнопку Завершити, щоб завершити виконання завдання.
</div>
}
@if(showErrorMessage()){
<div class="alert alert-danger">
  Ви більше не можете відправити відповідь. <br>
  Причина:
  @if(isRestrictionNAttempts()){
    <span class="fw-bold">У вас залишилось 0 спроб</span>
    <br>
  }
  @if(isRestrictionDeadline()){
    <span class="fw-bold">Минув дедлайн надання відповідей</span>
    <br>
  }
  Натисніть кнопку Завершити, щоб завершити виконання завдання.
</div>
}

<div class="d-flex justify-content-between gap-2 mt-3">
<button type="button" class="btn btn-secondary" (click)="goBack()">
  <i class="fa-solid fa-arrow-left me-1"></i> Назад
</button>
<div class="d-flex gap-2">
  <button class="btn btn-primary" (click)="answer()" [disabled]="disableInputsAndButtons() || isAnswersMode">
    <i class="fa-solid fa-paper-plane me-1"></i> Дати відповідь
  </button>
  <button class="btn btn-success" (click)="finish()" [disabled]="isAnswersMode">
    <i class="fa-solid fa-check me-1"></i> Завершити
  </button>
</div>
</div>

```

```

</div>
</div>

<div class="accordion mt-4" id="assignmentAccordion">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse"
        data-bs-target="#collapseOne" aria-expanded="false" aria-controls="collapseOne">
        Попередні спроби
      </button>
    </h2>
    <div id="collapseOne" class="accordion-collapse collapse" aria-labelledby="headingOne"
      data-bs-parent="#assignmentAccordion">
      <div class="accordion-body">
        @for (previousAnswer of previousAnswersDto; track previousAnswer) {
          <div>
            <strong>{{previousAnswer.numberOfAnswer + '. '}}</strong>
            Введена відповідь: <strong>{{previousAnswer.answer}}</strong>.
            Значення функції для цих значень: <strong>{{previousAnswer.result}}</strong>
          </div>
        }
        @if(previousAnswersDto.length === 0) {
          <strong>Попередніх відповідей не знайдено!</strong>
        }
      </div>
    </div>
  </div>
</div>

<div class="alert alert-info mt-4" role="alert">
  <i class="fa-solid fa-lightbulb me-2"></i>
  Підказка: уважно перевіряйте свої розрахунки перед тим, як дати відповідь.
</div>

</div>

```

Файл: frontend-project\src\app\assignment\assignment\assignment.component.ts

```

@Component({
  selector: 'app-assignment',
  templateUrl: './assignment.component.html',
  styleUrls: ['./assignment.component.css'],
  imports: [CommonModule, FormsModule, ReactiveFormsModule]
})
export class AssignmentComponent {
  constructor(public assignmentService: AssignmentService, public jwtService: JWTTokenService,
    public router: Router, private route: ActivatedRoute, private fb: FormBuilder, private location: Location) {
  }

  form!: FormGroup;

  assignmentId!: number;
  assignmentDto!: AssignmentDto;
  assignmentResponseDto!: AssignmentResponseDto;
  previousAnswersDto: AnswerDto[] = [];
  variablesErrorRequired = assignmentLabels['variable-empty'];
  variablesErrorNotNumber = assignmentLabels['variable-not-number'];
  isAnswersMode = false;

  nextAttemptTimer!: Subscription;
  canAttempt: boolean = false;

  ngOnInit() {
    this.form = this.fb.group({
      variables: this.fb.array([])
    });

    this.route.queryParamMap.subscribe(params => {
      this.isAnswersMode = params.get('answerMode')?.toLowerCase() === "true";
    });

    this.route.paramMap.subscribe(params => {
      this.assignmentId = Number(params.get('id'));
    });

    this.getById();
    this.getPreviousAnswers();
  }

```

```

    });
}

get variables(): FormArray {
    return this.form.get('variables') as FormArray;
}

get variablesIndices(): number[] {
    return Array(this.variables.length).fill(0).map((_, i) => i);
}

getById() {
    this.assignmentService.getById(this.assignmentId).subscribe({
        next: (assignmentDto: AssignmentDto) => {
            this.assignmentDto = assignmentDto;
            for (let i = 0; i < this.assignmentDto.variablesCount; i++) {
                this.variables.push(this.fb.control("", [Validators.required, Validators.pattern(/^-?\d+(\.\d+)?$/)]));
            }

            if (this.assignmentDto.status === AssignmentStatus.FINISHED) {
                this.isAnswersMode = true;
            } else {
                if (this.isRestrictionAttemptInNMinutes()) {
                    this.nextAttemptTimer = interval(1000).subscribe(() => {
                        this.canAttempt = !this.isNextAttemptTimeAfterNow();

                        if (this.canAttempt)
                            this.nextAttemptTimer.unsubscribe();
                    })
                }
            },
            error: (error: any) => {
                this.router.navigate(['/error/404']);
            }
        })
    }

    getPreviousAnswers() {
        this.assignmentService.getAnswers(this.assignmentId).subscribe({
            next: (answerDtos: AnswerDto[]) => {
                this.previousAnswersDto = answerDtos;
                if (this.previousAnswersDto.length > 0) {
                    this.assignmentResponseDto = new AssignmentResponseDto(this.previousAnswersDto[0].result,
                        this.previousAnswersDto[0].isCorrect, this.assignmentDto.restrictionType,
                        this.assignmentDto.attemptsRemaining, this.assignmentDto.deadline, this.assignmentDto.nextAttemptTime);
                }
            }
        })
    }

    answer() {
        this.onValidate();

        if (this.form.invalid) {
            return;
        }

        let variableNames = [], variableValues = [];
        for (let i = 0; i < this.variables.length; i++) {
            variableNames.push('x$ {i + 1} ');
            variableValues.push(this.variables.at(i).value);
        }
        let answer = joinVariables(variableNames, variableValues);

        this.assignmentService.answer(this.assignmentId, new AssignmentAnswerDto(answer))
            .subscribe({
                next: (assignmentResponseDto: AssignmentResponseDto) => {
                    this.assignmentResponseDto = assignmentResponseDto;
                    this.assignmentDto.attemptsRemaining = this.assignmentResponseDto.attemptsRemaining;
                    this.assignmentDto.deadline = this.assignmentResponseDto.deadline;
                    this.assignmentDto.nextAttemptTime = this.assignmentResponseDto.nextAttemptTime;
                    this.assignmentDto.restrictionType = this.assignmentResponseDto.restrictionType;

                    this.getPreviousAnswers();

                    if (this.isRestrictionAttemptInNMinutes()) {
                        this.nextAttemptTimer = interval(1000).subscribe(() => {

```

```

        this.canAttempt = !this.isNextAttemptTimeAfterNow();

        if (this.canAttempt)
            this.nextAttemptTimer.unsubscribe();
        })
    }
}
})
}

finish() {
    this.assignmentService.finish(this.assignmentId).subscribe({
        next: () => {
            this.router.navigate(['/assignments']);
        }
    })
}

onValidate() {
    if (this.form.invalid) {
        this.form.markAllAsTouched();
    }
}

disableInputsAndButtons() {
    return (this.assignmentResponseDto.hasCorrectAnswer)
        || (this.assignmentDto.hasRestrictionNAttempts() && this.isNoAttemptsLeft())
        || (this.assignmentDto.hasRestrictionDeadline() && this.isDeadlineBeforeNow())
        || (this.assignmentDto.hasRestrictionAttemptInNMinutes() && !this.canAttempt);
}

showAnswerNotCorrect() {
    return this.assignmentResponseDto && !this.assignmentResponseDto.hasCorrectAnswer && !this.isAnswersMode;
}

showAnswerCorrect() {
    return this.assignmentResponseDto && this.assignmentResponseDto.hasCorrectAnswer && !this.isAnswersMode;
}

showErrorMessage() {
    return this.assignmentDto &&
        ((this.isRestrictionNAttempts() && this.isNoAttemptsLeft()
            && (this.assignmentResponseDto && !this.assignmentResponseDto.hasCorrectAnswer))
            ||
            (this.isRestrictionDeadline() && this.isDeadlineBeforeNow()));
}

isRestrictionNAttempts() {
    return this.assignmentDto.restrictionType === AssignmentRestrictionType.N_ATTEMPTS
}

isRestrictionDeadline() {
    return this.assignmentDto.restrictionType === AssignmentRestrictionType.DEADLINE
}

isRestrictionAttemptInNMinutes() {
    return this.assignmentDto.restrictionType === AssignmentRestrictionType.ATTEMPT_PER_N_MINUTES
}

isNoAttemptsLeft() {
    return this.assignmentDto.attemptsRemaining === 0;
}

isNextAttemptTimeAfterNow() {
    return new Date(this.assignmentDto.nextAttemptTime).getTime() >=
        new Date().getTime()
}

isDeadlineBeforeNow() {
    return new Date(this.assignmentDto.deadline).getTime() <
        new Date().getTime()
}

goBack() {
    this.location.back();
}

```

```
}
```

Файл: api-tool\src\app\tool\tool.component.html

```
<div style="
  position: absolute;
  left: 0;
  top: 0;
  width: 75%;
  height: 100vh;
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
  overflow-y: auto;
  box-sizing: border-box;
  padding: 1.5rem
">

<div class="d-flex justify-content-between align-items-center mb-3">
  <h1><i class="fa-solid fa-network-wired me-2"></i> API Tool</h1>
  <button class="btn btn-danger" (click)="logout()">Вийти</button>
</div>

<form [formGroup]="urlForm">
  <div class="card p-3 border shadow-sm" style="width: 100%;">
    <h5>API запит</h5>
    <div class="d-flex gap-2 align-items-center mb-2">
      <select class="form-select" style="width: 10%;" formControlName="method">
        @for(method of methods; track method){
          <option [label]="method.label" [value]="method.method"></option>
        }
      </select>
      <input type="text" class="form-control flex-grow-1" placeholder="Введіть запит" formControlName="url" />
      <button class="btn btn-primary" (click)="sendRequest()">Надіслати</button>
    </div>
    @if(urlForm.get('url')?.errors?.['required'] &&
      (urlForm.get('url')?.touched || urlForm.get('url')?.dirty)) {
      <div class="text-danger mt-1">{{urlRequiredErrorMessage}}</div>
    }

    @if(urlForm.get('method')?.errors?.['required'] &&
      (urlForm.get('method')?.touched || urlForm.get('method')?.dirty)) {
      <div class="text-danger mt-1">{{methodRequiredErrorMessage}}</div>
    }

    @if(urlError){
      <div class="text-danger">{{urlErrorMessage}}</div>
    }
  </div>
</form>

@if(fields.length > 0){
  <form [formGroup]="fieldsForm">
    <div class="card p-3 border shadow-sm"
      style="width: 100%; display: flex; flex-direction: column; gap: 1rem; flex-shrink: 0;" formArrayName="fields">
      <h5>Поля запиту</h5>

      @for (control of fieldsForm.controls; track control) {
        <div class="d-flex align-items-start mb-2">
          <div style="width: 30%; flex-shrink: 0; padding-right: 20px;">
            <label class="form-label mb-1">{{fields[$index].label}} {{fields[$index].required ? 'обов\'язкове':
              'опціональне'}}</label>
            <div style="font-size: 0.9rem; color: #555;">{{mapFieldTypeToLabel(fields[$index].type)}}:
              {{fields[$index].description}}</div>
          </div>
          <div style="width: 70%">
            <div class="d-flex align-items-center">
              <input [type]="isDatetime(fields[$index]) ? 'datetime-local': 'text'"
                class="form-control" [placeholder]="Введіть ' + fields[$index].label"
                [formControlName]="`$index`" #fieldInput />
              @if(fields[$index].multiple){
                <button type="button" class="btn btn-danger ms-2" (click)="removeField($index)">
                  <i class="fa-solid fa-trash"></i>
                </button>
              }
            </div>
            @if (control.errors?.['required'] && (control.touched || control.dirty)) {
              <div class="text-danger mt-1">{{requiredErrorMessage}}</div>
            }
          </div>
        </div>
      }
    </div>
  </form>
}
```

```

    @if (control.errors?.['isInteger'] && (control.touched || control.dirty)) {
    <div class="text-danger mt-1">{{notIntegerErrorMessage}}</div>
    }

    @if (control.errors?.['isDecimal'] && (control.touched || control.dirty)) {
    <div class="text-danger mt-1">{{notDecimalErrorMessage}}</div>
    }

    @if (control.errors?.['isDatetime'] && (control.touched || control.dirty)) {
    <div class="text-danger mt-1">{{notDatetimeErrorMessage}}</div>
    }

  </div>
</div>

@if(fields[$index].multiple && isLastFieldWithName($index)){
<div class="d-flex justify-content-center">
  <button type="button" class="btn btn-primary me-3" style="width: 20%;" (click)="addField($index)">
    + Додати поле
  </button>
</div>
}
}

</div>
</form>
}

@if(showResponse){
<div class="card p-3 border shadow-sm" style="width: 100%;">
  <h5>Відповідь</h5>
  <p>Статус: {{responseStatus}}</p>
  @if(response){
  <pre class="bg-light p-2 border rounded" style="overflow-x: auto;">
    {{response | json}}
  </pre>
  }
</div>
}
</div>

<div style="
  position: fixed;
  right: 0;
  top: 0;
  width: 25%;
  height: 100vh;
  display: flex;
  flex-direction: column;
  overflow-y: auto;
  box-sizing: border-box;
">
  <div class="card border shadow-sm" style="overflow-y: auto; flex-grow: 1;">
    <div style="position: sticky; top: 0; background: #fff; z-index: 10; padding: 1rem;
      display: flex; flex-direction: column; gap: 0.5rem;">
      <h5>Всі запити</h5>
      <input type="text" class="form-control" placeholder="Знайти запит..." [formControl]="searchControl" />
    </div>
    <ul class="list-group list-group-flush">
      @for(possibleUrl of filteredPossibleUrls; track possibleUrl){
      <li class="list-group-item">
        <strong>{{getMethodTypeLabel(possibleUrl.method)}} {{possibleUrl.url}}</strong>
        <div style="font-size: 0.9rem; color: #555;">{{possibleUrl.description}}</div>
      </li>
      }
    </ul>
  </div>
</div>

```

Файл: api-tool\src\app\tool\tool.component.ts

```

@Component({
  selector: 'app-tool',
  templateUrl: './tool.component.html',
  imports: [ReactiveFormsModule, JsonPipe]
})
export class ToolComponent {

```

```

constructor(private fieldService: FieldService, private urlService: UrlService,
  private apiService: ApiService, private fb: FormBuilder, private authService: AuthService,
  private router: Router) {
}

@ViewChildren('fieldInput') fieldInputs!: QueryList<ElementRef>;

fieldsForm!: FormGroup;
urlForm!: FormGroup;
searchControl = new FormControl("");

allPossibleUrls: UrlDto[] = [];
filteredPossibleUrls: UrlDto[] = [];

methods: MethodTypeDto[] = [];

url!: UrlDto;
fields: FieldDto[] = [];

responseStatus!: number;
response: any;
showResponse = false;

urlError = false;
urlErrorMessage = "";

notIntegerErrorMessage = toolErrorLabels['field-not-integer'];
notDecimalErrorMessage = toolErrorLabels['field-not-decimal'];
notDatetimeErrorMessage = toolErrorLabels['field-not-datetime'];
requiredErrorMessage = toolErrorLabels['field-required'];

urlRequiredErrorMessage = toolErrorLabels['url-required'];
methodRequiredErrorMessage = toolErrorLabels['method-required'];

scrollTargetId!: string;

ngOnInit() {
  this.urlForm = this.fb.group({
    url: ['', [Validators.required]],
    method: ['', Validators.required]
  });

  this.urlForm.get('url')?.valueChanges
    .pipe(debounceTime(500))
    .subscribe((value: string | null) => {
      if (value !== null && value.length >= 6) {
        this.getUrlDtoByUrl();
      }
    })

  this.urlForm.get('method')?.valueChanges
    .pipe(debounceTime(500))
    .subscribe((value: string | null) => {
      if (this.urlForm.value.url !== null && this.urlForm.value.url.length >= 6) {
        this.getUrlDtoByUrl();
      }
    })

  this.searchControl.valueChanges
    .pipe(debounceTime(500))
    .subscribe((value: string | null) => {
      if (value !== null)
        this.filteredPossibleUrls = this.filterAllPossibleUrls(value);
    });

  this.urlService.getAll().subscribe({
    next: (urlDtos: UrlDto[]) => {
      this.allPossibleUrls = urlDtos;
      this.filteredPossibleUrls = this.allPossibleUrls;
    }
  });

  this.urlService.getMethods().subscribe({
    next: (methodTypeDto: MethodTypeDto[]) => {
      this.methods = methodTypeDto;
    }
  })
}

```

```

get fieldsFormControls(): FormArray {
  return this.fieldsForm.get('fields') as FormArray;
}

getUrlDtoByUrl() {
  if (this.urlForm.invalid) {
    this.urlError = false;
    this.validateForm(this.urlForm);
    return;
  }

  this.urlService.getByUrl(this.urlForm.value.url, this.urlForm.value.method).subscribe({
    next: (urlDto: UrlDto[]) => {
      this.url = urlDto[0];
      this.urlError = false;

      this.getFieldsForUrl();
    },
    error: (error: any) => {
      if (error.error.status === 404) {
        this.urlError = true;
        this.urlErrorMessage = error.error.message;
        this.fields = [];
        this.showResponse = false;
      }
    },
    complete: () => {
      this.fields = [];
      this.showResponse = false;
    }
  })
}

getFieldsForUrl() {
  this.fieldService.getByUrlId(this.url.id).subscribe({
    next: (fieldDtos: FieldDto[]) => {
      this.fields = fieldDtos;

      this.fieldsForm = this.fb.group({
        fields: this.fb.array([])
      });

      for (let fieldDto of fieldDtos) {
        this.fieldsFormControls.push(new FormControl("", this.getValidatorsForField(fieldDto)));
      }
    }
  })
}

sendRequest() {
  this.validateForm(this.urlForm);

  if (this.urlForm.invalid) {
    return;
  }

  if (this.fields.length > 0) {
    this.validateForm(this.fieldsForm);

    if (this.fieldsForm.invalid) {
      const firstInvalidIndex = this.fieldsFormControls.controls.findIndex(c => c.invalid);
      this.scrollToTarget(firstInvalidIndex);
      return;
    }
  }

  let queryOrBody: Record<string, any[]> = {};
  for (let i = 0; i < this.fields.length; i++) {
    if (this.fields[i].multiple === true) {
      Object.assign(queryOrBody, {
        [this.fields[i].name]: [...(queryOrBody[this.fields[i].name] || []),
          this.fieldsFormControls.at(i).value]
      })
    } else {
      Object.assign(queryOrBody, { [this.fields[i].name]: this.fieldsFormControls.at(i).value })
    }
  }
}

```

```

this.showResponse = true;
this.apiService.sendRequest(this.urlForm.value.url, this.urlForm.value.method, queryOrBody).subscribe({
  next: (resp: HttpResponse<any>) => {
    this.responseStatus = resp.status;
    if (resp.body)
      this.response = resp.body;
    else
      this.response = null;
  },
  error: (err) => {
    this.responseStatus = err.status;
    if (err.error)
      this.response = err.error;
  }
})
}

addField(index: number) {
  const field = this.fields[index];
  const nextItemIndex = index + 1;
  this.fields.splice(nextItemIndex, 0, field);
  this.fieldsFormControls.insert(nextItemIndex, new FormControl("", this.getValidatorsForField(field)));
}

removeField(index: number) {
  let name = this.fields[index].name;
  let fieldsWithNameLength = this.fields.filter(f => f.name === name).length;
  if (fieldsWithNameLength === 1) {
    this.fieldsFormControls.at(index).setValue("");
  } else {
    this.fields.splice(index, 1);
    this.fieldsFormControls.removeAt(index);
  }
}

logout() {
  this.authService.logout();
  this.router.navigateByUrl("/apitool/auth/signin");
}

getValidatorsForField(fieldDto: FieldDto): ValidatorFn[] {
  let validators: ValidatorFn[] = [];
  if (fieldDto.required)
    validators.push(Validators.required)

  switch (fieldDto.type) {
    case FieldType.DATETIME:
      validators.push(datetimeValidator());
      break;
    case FieldType.DECIMAL:
      validators.push(decimalValidator());
      break;
    case FieldType.ENUM:
    case FieldType.INTEGER:
      validators.push(integerValidator());
      break;
  }
  return validators;
}

validateForm(form: FormGroup) {
  if (form.invalid) {
    form.markAllAsTouched();
    return;
  }
}

getMethodTypeLabel(methodType: MethodType) {
  return mapMethodTypeToLabel(methodType);
}

filterAllPossibleUrls(searchValue: string) {
  return this.allPossibleUrls.filter(url =>
    url.description.toLowerCase().includes(searchValue.toLowerCase())
    || url.url.toLowerCase().includes(searchValue.toLowerCase()));
}

```

```
isDatetime(field: FieldDto) {  
  return field.type === FieldType.DATETIME;  
}  
  
mapFieldTypeToLabel(fieldType: FieldType) {  
  return mapFieldTypeToLabel(fieldType);  
}  
  
scrollToTarget(index: number) {  
  const el = this.fieldInputs.get(index)?.nativeElement;  
  el?.scrollIntoView({ behavior: 'smooth', block: 'center' });  
  el?.focus();  
}  
  
isLastFieldWithName(index: number) {  
  if (index === this.fields.length - 1)  
    return true;  
  return this.fields[index + 1].name !== this.fields[index].name;  
}  
}
```

ДОДАТОК В

Результати перевірки роботи на співпадіння



Дата звіту 12/5/2025
Дата редагування 12/5/2025

Документ прийнятий

Звіт подібності

Метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-43мп_ЯремчукД_ПЗ
Автор
Науковий керівник / Експерт
ЯремчукДФіногенов О.Д.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



10612
Кількість слів

82559
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	Б	0
Інтервали	A→	0
Мікропробіли	␣	0
Білі знаки	␣	0
Парафрази (SmartMarks)	a	40

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Копію тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	68 0.64 %
2	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	35 0.33 %

3	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	29 0.27 %
4	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	28 0.26 %
5	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	27 0.25 %
6	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	22 0.21 %
7	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	17 0.16 %
8	https://ela.kpi.ua/bitstreams/bd6b3a15-b4f9-409c-891b-7c2b8e0d73fa/download	15 0.14 %
9	https://ela.kpi.ua/bitstreams/bd6b3a15-b4f9-409c-891b-7c2b8e0d73fa/download	13 0.12 %
10	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	13 0.12 %
з домашньої бази даних (3.78 %)		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Вебзастосунок збору та аналізу наукометричних даних 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	346 (20) 3.26 %
2	Програмне забезпечення рекомендаційної системи підбору комп'ютерних ігор 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	55 (7) 0.52 %
з програми обміну базами даних (0.25 %)		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
3	Вань Анастасія Валеріївна.pdf 4/21/2023 Odessa National Polytechnic University (ІКС, кафедра інформ. систем)	15 (2) 0.14 %
4	Курсова робота 5/7/2025 King Danylo University (King Danylo University)	12 (2) 0.11 %
з Інтернету (0.95 %)		
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

5	https://ela.kpi.ua/bitstreams/bd6b3a15-b4f9-409c-891b-7c2b8e0d73fa/download	86 (8) 0.81 %
6	https://ela.kpi.ua/bitstreams/3c6a001c-9d77-4cb6-b587-e9b2c95ae301/download	8 (1) 0.08 %
7	https://ela.kpi.ua/bitstreams/e9cda8e5-c3a6-498a-993d-4ceff6f6c78a/download	7 (1) 0.07 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
------------------	-------	---------------------------------------