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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»

Виконала: студентка IV курсу, групи ІС-81

Бишовець Наталія Миколаївна _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: доц., к.т.н., доц. Жданова Олена Григорівна _____ _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому індивідуальному дослідницькому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти	Перший (бакалаврський)
Спеціальність	126 «Інформаційні системи та технології»
Освітньо-професійна програма	Інформаційні управляючі системи та технології

ЗАВДАННЯ

на індивідуальний дослідницький проєкт студентці

Бишовець Наталії Миколаївні

(прізвище, ім'я, по батькові)

1. Тема проєкту: «Інформаційна система підтримки визначення першочерговості виконання робіт»
керівник проєкту: Жданова Олена Григорівна, к.т.н., доцент
2. Термін подання студентом проєкту: 15 червня 2022 року
3. Вихідні дані до проекту: Технічне завдання
4. Зміст пояснювальної записки:
1. Загальні положення: опис предметного середовища, опис процесу діяльності, опис функціональної моделі, огляд наявних аналогів, постановка задачі, призначення розробки, цілі та задачі розробки.
2. Інформаційне забезпечення: вхідні та вихідні дані, опис структури масивів інформації.
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення.
5. Технологічний розділ: керівництво користувача, методика випробувань Програмного продукту.
5. Перелік графічного матеріалу:
1. Схема структурна варіантів використання.
2. Схема структурна функціональних компонентів програмного забезпечення.
3. Креслення вигляду екранних форм.
6. Дата видачі завдання: 1 грудня 2021 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення рекомендованої літератури	03.05.2022	
2.	Аналіз існуючих методів розв’язання задачі	06.05.2022	
3.	Постановка та формалізація задачі	10.05.2022	
4.	Розробка інформаційного забезпечення	13.05.2022	
5.	Алгоритмізація задачі	17.05.2022	
6.	Обґрунтування використовуваних технічних засобів	20.05.2022	
7.	Розробка програмного забезпечення	26.05.2022	
8.	Налагодження програми	02.06.2022	
9.	Виконання графічних документів	06.06.2022	
10.	Оформлення пояснювальної записки	10.06.2022	
11.	Подання індивідуального дослідницького проєкту	15.06.2022	

Студентка Наталія БИШОВЕЦЬ

Керівник Олена ЖДАНОВА

№ з/п	Формат	Позначення	Найменування	Кількість аркушів	Примітка
1			Документація загальна	2	
2	А4	ІС81.040БАК.003 ПЗ	Пояснювальна записка	144	
3	А4	ІС81.040БАК.003 ТЗ	Технічне завдання	10	
4	А3	ІС81.040БАК.003 Д1	Інформаційна система підтримки визначення першочерговості виконання. робіт. Схема бази даних	1	
5	А3	ІС81.040БАК.003 Д2	Інформаційна система підтримки визначення першочерговості виконання. робіт. Діаграма варіантів використання	1	
6	А3	ІС81.040БАК.003 Д3	Інформаційна система підтримки визначення першочерговості виконання. робіт. Діарграма функціональних компонентів	1	
7	А3	ІС81.040БАК.003 КЕ	Інформаційна система підтримки визначення першочерговості виконання. робіт. Креслення вигляду екранних форм	1	
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					

Зм.	Арк.	ПІБ	Підп.	Дата	ІС81.040БАК.003 ТП		
Розробила		Бишовець Н.М.			Інформаційна система підтримки визначення першочерговості виконання робіт. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівник		Жданова О.Г.				Т	
						КПІ ім. Ігоря Сікорського Група ІС-81	
Затв.							

Пояснювальна записка до дослідницького індивідуального проєкту на тему: «Інформаційна система підтримки визначення першочерговості виконання робіт»

Київ – 2022 року

АНОТАЦІЯ

Структура та обсяг роботи: пояснювальна записка індивідуального дослідницького проєкту складається з п’яти розділів, містить 144 сторінок, 66 рисунків, 46 таблиць, 1 додаток, 16 джерел.

Індивідуальний дослідницький проєкт присвячений розробці інформаційної системи підтримки визначення першочерговості виконання робіт.

Метою даної системи є підвищення якості визначення першочерговості виконання робіт особою, що приймає рішення (ОПР), та зменшення часу на їх прийняття.

Завдання розробки: реалізація функціоналу ведення експертів, робіт, критеріїв та їх оцінок, реалізація алгоритмів для переведення оцінок за суб’єктивними критеріями у кількісну форму та для вирішення задачі про визначення першочерговості виконання робіт, візуалізація результатів.

У розділі інформаційного забезпечення описані вхідні та вихідні дані, а також структура бази даних.

Розділ математичного забезпечення присвячений обґрунтуванню і опису методів розв’язання задачі про визначення першочерговості виконання робіт.

Розділ програмного та технічного забезпечення надає опис засобів розробки, вимог до технічного забезпечення, архітектури програмного застосунку, що створений з метою підвищення якості визначення першочерговості виконання робіт особою, що приймає рішення (ОПР).

У технологічному розділі описані керівництво користувача та випробування програмного продукту.

ПЕРШОЧЕРГОВІСТЬ ВИКОНАННЯ РОБІТ, TOPSIS, JAVASCRIPT, REACT.JS, NODE.JS, EXPRESS.JS, POSTGRESSQL, ЕКСПЕРТИ.

					ІС81.040БАК.003 ПЗ			
		Прізвище	Підпис	Дата				
Розробила		Бишовець Н.М.			Інформаційна система підтримки визначення першочерговості виконання робіт. Пояснювальна записка.	Літ.	Аркуш	Аркушів
						Т	2	144
Перевірила		Жданова О.Г.				КПІ ім. Ігоря Сікорського Група ІС-81		
Затв.								

ABSTRACT

Bachelor’s thesis: 144 pages, 66 figures, 46 tables, 1 appendix, 16 references.

An individual research project is dedicated to the development of an information system to support the determination of the priority of tasks.

Purpose of the work – improving the quality of determining the priority of tasks by the decision-maker, and reducing the time for their adoption.

Tasks of the work – implementation of the functionality of conducting experts, tasks, criteria and their evaluations, implementation of algorithms for translating evaluations of subjective criteria into quantitative form and to solve the problem of determining the priority of work, visualization of results.

The information support chapter describes the input and output data, the database structure.

The mathematical support chapter is dedicated to the substantiation and description of methods for solving the problem of determining the priority of work.

The software and hardware chapter provides a description of the development tools, hardware requirements, software application architecture, which is designed to improve the quality of determining the priority of work by the decision maker.

The technology chapter describes the user guide and software product tests.

PRIORITY OF TASKS, TOPSIS, JAVASCRIPT, REACT.JS, EXPERTS, NODE.JS, EXPRESS.JS, POSTGRESSQL.

ЗМІСТ

ВСТУП

Потреба вибрати найважливіше з багатьох різноманітних варіантів виникає в повсякденному житті постійно. Визначаючи план роботи на наступний квартал, формуючи бюджет, плануючи відпустку чи продумуючи методи оптимізації процесів, ми змушені розставляти пріоритети. Проте, не завжди у цьому питанні варто довіряти лише інтуїції. Незважаючи на те, що інтуїція – це наш вірний помічник у багатьох життєвих питаннях, приймаючи важливі рішення, потрібно покладатися на більш «холодний» підхід. Тож, аби не стати жертвою власної ірраціональності, люди використовують різноманітні методи пріоритезації.

Встановлення пріоритетності виконання тих чи інших робіт залежить від того, які критерії використовуються. Наприклад, аналізовані роботи можуть порівнюватись за такими критеріями:

– складність виконання;
– вартість реалізації;
– кількість часу, потрібного на виконання;
– нагальність проблеми;
– кількість робітників, необхідних для виконання роботи тощо.

Якщо критерії є об’єктивними, то сама процедура визначення пріоритетності зрозуміла, але якщо критерії є суб’єктивними, необхідно залучати експертів.

В цьому індивідуальному дослідницькому проєкті запропонований шлях визначення пріоритетності виконання робіт за об’єктивними критеріями, який базується на використанні методів аналізу прийняття групових рішень та способі обробки експертних оцінок.

Експертне оцінювання – це найбільш доступний і універсальний метод отримання та аналізу інформації про стан різних об'єктів і суб'єктів, а також є єдиним способом отримання необхідної інформації для об'єктів, що не мають

статистичної інформації функціонування та які характеризуються своєю структурно-параметричною невизначеністю [1].

Крім того, щоб нівелювати суб'єктивність у процесі прийняття рішень людьми, порівнюючи роботи та оцінюючи компроміси щодо їх важливості, були використані методи для визначення пріоритетності суджень, отриманих від експертів у цій галузі.

Надалі, після визначення всіх об'єктивних оцінок, було застосовано метод TOPSIS (the Technique for Order Preferences by Similarity to an Ideal Solution), який використовується для розв'язання задач за умови, коли необхідно серед декількох альтернатив, які оцінюються за двома чи більше кількісними критеріями, знайти найкращий [2]. Метод впорядковує альтернативи у порядку першочерговості за важливістю. Його основна ідея полягає у концепціях пошуку компромісного розв'язку при виборі найкращої альтернативи, яка б була найближчою до позитивного ідеального розв'язку (скорочено PIS, або утопічної точки, оптимального розв'язку) та найвіддаленішим від негативного ідеального розв'язку (скорочено NIS, або антиутопічної точки). В якості розв'язку вибирається альтернатива, що є найкращою згідно побудованого впорядкування [3]. Оскільки для кожної альтернативи розраховується деяка величина (за заданою метрикою), то згідно цієї величини можна відсортувати аналізовані альтернативи і тим самим встановити їх пріоритетне упорядкування.

Практичне значення одержаних результатів: розроблена інформаційна система дозволяє визначати першочерговість виконання робіт у будь-яких практичних задачах.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.1.1 Опис процесу діяльності

Теорія прийняття рішень широко застосовується в багатьох галузях: виробничій, військовій, IT, освітній, транспортній, будівельній тощо.

Нехай існує певна організація (фірма чи компанія), перед керівництвом якої стоїть необхідність визначення першочерговості виконання робіт. Роботою можна вважати будь-який вид діяльності, реалізацію множини яких необхідно пріоритезувати: розробку функціоналу програмних продуктів, розробку та реалізацію сервісів, вдосконалення будь-якого виду продукції, оновлення технічного забезпечення на підприємстві тощо.

Кожен із видів робіт характеризується деякими критеріями, які можуть бути об’єктивними: наприклад, кількість років експлуатації, кінцевий строк реалізації, вартість оновлення та інше, та суб’єктивними: наприклад, зовнішній вигляд пристрою, складність реалізації функціоналу програмного продукту тощо. Для визначення оцінки суб’єктивного критерію необхідно залучати експертів, кожен з яких виразить свою думку, на основі множини яких буде формуватися об’єктивна узагальнена оцінка.

В основі даної роботи знаходиться особа, що приймає рішення (ОПР), метою якої є розв’язання задачі визначення першочерговості виконання робіт за певними критеріями.

В даному індивідуальному дослідницькому проєкті розглядаються алгоритми визначення пріоритетності виконання робіт за якісними критеріями, які базуються на використанні методів аналізу прийняття групових рішень та способі обробки експертних оцінок. Крім того, щоб нівелювати суб’єктивність у процесі прийняття рішень людьми, порівнюючи роботи та оцінюючи компроміси щодо їх важливості, були використані методи для визначення пріоритетності суджень, отриманих від експертів у цій галузі.

1.1.2 Опис функціональної моделі

Функціональна модель описана за допомогою діаграми варіантів використання – Use Case Diagram, на котрій зображено відношення між акторами та прецедентами в системі. Тобто діаграма є графом, який складається з множини акторів, прецедентів, відношень між ними та відношень узагальнення між акторами [4].

В даній функціональній моделі актором виступає особа, що приймає рішення (ОПР), яка формує загальні оцінки по кожному з критеріїв із використанням інформації про значення об’єктивних та суб’єктивних оцінок критеріїв, останній з яких формуються за допомогою експертів.

Загальна функціональність моделі представлена в графічних матеріалах на листі 2 (Д2 – діаграма варіантів використання).

1.2 Огляд наявних аналогів

В наш час зросла актуальність застосування комп’ютерних технологій для вирішення задач різного типу в усіх областях людської діяльності. Саме тому на ринку програмних продуктів існує велика кількість програм, які вирішують масу нагальних проблем людини, тим самим полегшуючи їй роботу. Проте систем, які б визначали першочерговість виконання робіт із залученням експертів або без них, наразі не виявлено у відкритому доступі.

1.3 Постановка задачі

1.3.1 Призначення розробки

Інформаційна система підтримки визначення першочерговості виконання робіт (далі – Система) призначена для підтримки процесу визначення порядку виконання робіт за різними критеріями.

1.3.2 Цілі та задачі розробки

Метою даної Системи є підвищення якості визначення першочерговості виконання робіт особою, що приймає рішення (ОПР), та зменшення часу на їх прийняття.

Задля досягнення поставленої мети, програмний продукт повинен містити в собі ряд функцій:

- ведення експертів (додавання, видалення та редагування);
- ведення робіт (додавання, видалення та редагування);
- ведення критеріїв (додавання, видалення та редагування);
- ведення значень критеріїв (додавання, видалення та редагування);
- обчислення значень критеріїв оцінки робіт на основі експертних оцінок;
- визначення першочерговості виконання робіт;
- формування звітів (у розгорнутому та нерозгорнутому виглядах).

Висновок до розділу

В даному розділі був наданий опис предметного середовища, де може застосовуватись розроблена система, призначення та мета розробки. Було описано функціональні вимоги, які мають бути реалізовані.

Отже, у даній індивідуальній дослідницькій роботі буде розроблений програмний продукт, який дозволить значно спростити роботу особі, що приймає рішення (ОПР), а також пришвидшить сам процес. Існування такого інструменту закладе основу для подальшого дослідження цієї теми.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані – це дані, що надходять до системи та необхідні для її подальшої роботи. Вхідні дані реєструються в системі за допомогою форм введення. Перша форма – додавання нового експерта до бази даних (далі – БД). Друга форма – додавання нової роботи до БД. Третя форма – додавання нового критерію до БД. Четверта форма – додавання оцінок за обраними критеріями по обраній роботі до БД. Нижче наведені дані, які надходять до системи за допомогою цих форм.

Дані про експерта:

– ім’я – використовується для ідентифікації експерта.

Дані про роботу:

– назва – використовується для ідентифікації роботи.

Дані про критерій:

– назва – використовується для ідентифікації критерію;

– тип (об’єктивний чи суб’єктивний) – використовується для визначення, чи необхідні попередні обчислення, пов’язані з переведенням оцінок критерію у кількісну форму.

Дані про оцінки об’єктивних критеріїв:

– назва роботи, яку необхідно оцінити;

– назва критерію, за яким необхідно оцінити роботу;

– кількісна оцінка – основна інформація, що використовується при розрахунках.

Дані про оцінки суб’єктивних критеріїв:

– назва роботи, яку необхідно оцінити;

– назва критерію, за яким необхідно оцінити роботу;

– ім’я експерта, який оцінює;

– суб’єктивна оцінка, яка при подальших розрахунках буде переведена у кількісну.

2.2 Вихідні дані

Вихідними даними результату виконання алгоритмів визначення значень критеріїв оцінки робіт на основі експертних оцінок та визначення першочерговості виконання робіт на основі отриманих значень критеріїв оцінки усіх альтернатив є сформований порядок виконання робіт у порядку їх важливості. Приклад звіту, який відображає порядок виконання робіт у порядку їх важливості наведено на рисунку 2.1.

Порядок виконання робіт:

Порядок виконання	**1**	**2**	**3**	**4**	**…**	**n**
Назви робіт	Робота №1	Робота №2	Робота №3	Робота №4	…	Робота №n
Ваги	Вага №1	Вага №2	Вага №3	Вага №4	…	Вага №n

Рисунок 2.1 – Приклад звіту

2.3 Опис структури бази даних

В якості СУБД (системи управління базами даних) було обрано PostgreSQL (або Postgres) – це об’єктно-реляційна СУБД, що використовує SQL як основну структуровану мову запитів.

PostgreSQL підтримує безліч типів даних, таких як числові, рядкові, булеани, дату та час. Також вона підтримує типи даних для геометричних фігур, зображень, мережевих адрес, бітових рядків, пошуку тексту та JSON-записів [5].

В реляційній структурі БД всі дані представлені у вигляді простих таблиць; кожна таблиця складається із стовпців (поля або атрибути) та рядків (записи або кортежі) тощо [6].

Таблиці проектованої БД відносяться до реляційної бази даних, тому наведемо ряд їх властивостей:

– кожен запис створюється з унікальним індексом, який є унікальною характеристикою запису, отже, в БД не може бути двох однакових рядків;

– у кожного стовпця може бути лише унікальна назва, а всі дані у стовпці можуть бути лише одного типу (число, рядок, булеан тощо);

– таблиця може бути порожньою (не мати жодних рядків), проте обов’язково повинна мати хоча б один стовпець, який не є індексом (цей стовпець створюється за замовчуванням).

Структура бази даних представлена в графічних матеріалах на листі 1 (Д1 – схема бази даних).

Для детального опису структури БД дані будуть подані у табличному форматі: ім’я таблиці, назва поля, тип даних та опис поля (таблиці 2.1 – 2.4).

Таблиця 2.1 – Структура таблиці Criteria (таблиця критеріїв)

Назва поля	Тип даних	Опис поля
id	integer	Унікальний ідентифікатор критерію
Name	text	Назва критерію
IsQualityCriteria	boolean	Тип критерію (об'єктивний чи суб'єктивний)

Унікальними є поля: id та Name (може бути лише один критерій з такою назвою).

Таблиця 2.2 – Структура таблиці Task (таблиця робіт)

Назва поля	Тип даних	Опис поля
id	integer	Унікальний ідентифікатор роботи
Name	text	Назва роботи

Унікальними є поля: id та Name (може бути лише одна робота з такою назвою).

Таблиця 2.3 – Структура таблиці Experts (таблиця експертів)

Назва поля	Тип даних	Опис поля
id	integer	Унікальний ідентифікатор експерта
Name	text	Ім'я експерта

Унікальними є поля: id та Name (може бути лише одна робота з такою назвою).

Таблиця 2.4 – Структура таблиці Task_Criteria (таблиця оцінок)

Назва поля	Тип даних	Опис поля
id	integer	Унікальний ідентифікатор оцінки
Task_ID	integer	Ідентифікатор роботи
Criteria_ID	integer	Ідентифікатор критерію
Expert_ID	integer	Ідентифікатор експерта (null, якщо критерій об'єктивний)
CriteriaValue	integer	Значення оцінки критерію

Унікальними є поля: id, Task_ID + Criteria_ID (у випадку об’єктивного критерію може бути лише одна оцінка за одним критерієм) та Task_ID + Criteria_ID + Expert_ID (у випадку суб’єктивного критерію може бути лише оцінена одним експертом оцінка за одним критерієм).

Висновок до розділу

В даному розділі наведений опис вхідних та вихідних даних розробленого програмного продукту, а також описана структура використаної системи управління базами даних.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

В процесі визначення першочерговості послідовно розв’язуються дві задачі:

– задача визначення значень критеріїв оцінки робіт на основі експертних оцінок (для тих критеріїв, для яких немає змоги об’єктивно визначити (виміряти) кількісне значення);

– задача визначення першочерговості виконання робіт (розв’язується після того, як отримані усі значення критеріїв оцінки усіх альтернатив).

3.1 Обґрунтування методів розв’язання

Задача визначення першочерговості виконання робіт має наступні особливості:

– існує декілька критеріїв, що впливають на визначення першочерговості виконання робіт;

– ці критерії часто не мають чіткого кількісного вимірювання (можуть стосуватись аспектів матеріального чи нематеріального характеру, ретельно виміряних або грубо оцінених, добре або погано зрозумілих тощо).

З урахуванням перерахованих особливостей постановки задачі для її вирішення часто використовують метод аналізу ієрархій (MAI) Сааті (Analytic hierarchy process), який набув популярності як інструмент підтримки прийняття рішень щодо багатокритеріальних, багатоцільових та багатоієрархічних проблем. Суть MAI полягає в тому, що при проведенні оцінок можна використовувати людські судження, а не лише основну (числову) інформацію.

MAI [7] розкладає проблему на більш прості і зрозумілі підпроблеми, кожну з яких можна проаналізувати окремо. MAI підходить для визначення першочерговості виконання робіт з використанням інформації про експертні оцінки групою експертів, які приймають рішення, оскільки:

– він має добре розроблені теорії для визначення суджень та оцінки послідовності оцінюваних пріоритетів;

– існують визначені методи покращення узгодженості пріоритетів;

– він також пропонує різні методи агрегування групових уподобань;

– існують надійні теорії для досягнення консенсусу в групах, за допомогою яких можлива компромісна, але прийнятна пріоритезація.

В основі застосування методу аналізу ієрархій Сааті лежать такі етапи:

– побудова ієрархії для встановлення чинників, що впливають на вирішення проблеми;

– визначення локальних пріоритетів на усіх рівнях ієрархії;

– визначення глобальних пріоритетів альтернатив, що дозволяють оцінити вплив кожної з альтернатив на досягнення поставленої мети.

3.2 Задача визначення значень критеріїв оцінки робіт на основі експертних оцінок

3.2.1 Змістовна постановка задачі

Нехай існує:

– множина робіт (множину альтернатив);

– група експертів, кожен з яких оцінює всі альтернативи за шкалою від 1 до 10;

– початковий рівень компетентності кожного з експертів (ваговий коефіцієнт компетентності);

– матриця експертних оцінок важливості робіт заданої множини.

Необхідно визначити першочерговість виконання робіт.

3.2.2 Математична постановка задачі

Розглядається група з k експертів та множина $P = \{1, \dots, n\}$ робіт.

Для кожного з експертів можуть бути встановлені початкові (приблизні) вагові коефіцієнти компетентності y_i. Вагові коефіцієнти компетентності є нормованими, тобто $\sum_{i=1}^{k} \gamma_i = 1$. За відсутністю інформації, згідно якої можна встановити початкові значення величин γ_i, вважається, що ступінь кваліфікації експертів однаковий, тобто $\gamma_i = \frac{1}{k},\ i = 1, \dots, k$.

Кожен з експертів оцінює важливість кожної альтернативи множини $P = \{1, \dots, n\}$ за заданою шкалою оцінювання. В результаті експертного оцінювання отримана матриця оцінок:

$$C = \{c\_ij\}, i = 1, \dots, k, j = 1, \dots, n, \qquad (3.1)$$

де $c_{ij} \in [c^H,\ c^B], i = 1, \dots, k, j = 1, \dots, n$ – оцінка і-м експертом j-го критерія; c^H, c^B – відповідно нижня та верхня границі шкали оцінок. На виході необхідно отримати множину всіх U – векторів, що відображають важливість критеріїв та визначити за допомогою них пріоритетність робіт.

3.2.3 Опис методу розв’язання

Метод узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів рекомендується застосовувати у випадках, коли попередньо неможливо встановити рівень (ступінь) компетентності експертів, або цей рівень визначено дуже приблизно.

Дано для задачі визначення значень критеріїв оцінки робіт на основі експертних оцінок**:**

– група експертів;

– для кожного з експертів може бути задано (початковий) рівень його компетентності (ваговий коефіцієнт компетентності);

– множина робіт (множина альтернатив);

– матриця експертних оцінок важливості робіт заданої множини.

Мета: виконати ранжування робіт за їх важливістю, тобто визначити вектор зважених результуючих оцінок важливості робіт.

Ідея методу: припускається, що деякі експерти можуть оцінювати альтернативи краще від інших експертів і, відповідно, в середньому, при оцінці робіт вони повинні мати більші значення вагових коефіцієнтів компетентності. Таким чином, коефіцієнт компетентності кожного експерта є змінним і залежить від того, наскільки його думка відрізняється від узагальненої думки групи в цілому. В процесі розв'язання задачі величини вагових коефіцієнтів рівня компетентності експертів ітеративно уточняються (перераховуються).

Складність виконання такого алгоритму визначається одним циклом, тому вона складає O(n) [8].

Нижче наведений псевдокод алгоритму методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів:

1 Вхід: C – матриця оцінок альтернатив експертами;
$\mathrm{coefsMatrix}$ – матриця коефіцієнтів компетентності експертів;
l – порогова величина сумарної зміни вагових коефіцієнтів експертів;
k – кількість експертів;
i – кількість робіт.

2 Вихід: $\mathrm{sumRates}$ – об'єкт, що відображає важливість критеріїв на кожній з пройдених ітерацій.

3 Знайти зважену матрицю оцінок альтернатив експертами:
$\mathrm{weighedC} = \{c_{ij}\}$, де $c_{ij} = C_{ij} / \sum C_i,\ i = 1, \dots, k, j = 1, \dots, n.$

4 Знайти початкову матрицю компетенцій експертів:
$\mathrm{coefsMatrix} = \{y_{ij}\}$, де $y_{ij} = 1/k,\ i = 1, \dots, k, j = 1, \dots, n.$

5 $G := \infty$

6 величина середньоарифметичного значення вагових коефіцієнтів перевищує задану
п

7 З

8 Знайти матрицю відповідей експертів з урахуванням їх компетенції:
$S = \{s_{ij}\},\ \text{де } s_{ij} = c_{ij} \cdot y_{ij},\ i = 1, \dots, k, j = 1, \dots, n.$

9 Знайти вектор сумарних оцінок об'єктів:
$P = \{p_i\},\ \text{де } p_i \mathrel{+}= c_{ji},\ i = 1, \dots, k, j = 1, \dots, n.$

10 Знайти середньоарифметичне значення оцінок альтернатив із врахуванням компетентності експертів:
$U = \{u_j\},\ \text{де } u_j = \sum_{i=1}^{k} \gamma_i \cdot c_{ij},\ j = 1, \dots, n.$

11 Записати U та iteration до об’єкту sumRates.

12 Обчислити матрицю відхилень відповідей від середньозважених: $Z = \{z_{ij}\}$, де $z_{ij} = |s_{ij} - u_j|,\ i = 1, \dots, k, j = 1, \dots, n.$

13 Для кожного з експертів знайти та нормувати суму відхилень відповідей від середньозважених (чим більша ця величина, тим менша компетентність експерта):
$\Delta = \{\delta_i\}$, де $\delta_i = \left|\sum_{j=1}^{n} z_{ij}\right|,\ i = 1, \dots, k, j = 1, \dots, n.$

14 Визначити та нормувати нові значення компетенцій експертів:
$\text{coefsMatrixNew} = \{\hat{\gamma}_{ij}\}$, де $\hat{\gamma}_{ij} = y_{ij}/\delta_i,\ i = 1, \dots, k, j = 1, \dots, n.$
Перерахувати середньоарифметичне значення оцінок альтернатив із врахуванням компетентності експертів:
$\hat{U} = \{\hat{u}_j\},$ де $\hat{u}_j = \sum_{i=1}^{k} \hat{\gamma}_i \cdot c_{ij},\ j = 1, \dots, n.$

15 If (якщо) $\text{iteration} > 1$:

16 Обчислити величину сумарної зміни вагових коефіцієнтів:

$=$

18 end if

18 Збільшити кількість ітерацій $\text{iteration} \mathrel{+}= 1$

19

3.2.4 Застосування розробленого алгоритму на прикладі

В цьому прикладі буде наведене виконання методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів для розв’язання задачі з 12-ма експертами, які надали свої оцінки 4-ом альтернативам (роботам). Для наочності буде наведений результат виконання першої та останньої частин ітерації програмного продукту.

Нехай порогова величина сумарної зміни вагових коефіцієнтів буде дорівнювати 0.001. Тоді матриця оцінок альтернатив експертами та порогова величина будуть мати такий вигляд (рисунок 3.1).

Матриця оцінок альтернатив експертами:

(index)	0	1	2	3
0	1	3	2	4
1	2	3	4	1
2	1	4	3	2
3	1	3	4	2
4	1	4	3	2
5	2	1	3	4
6	3	2	1	4
7	1	2	4	3
8	4	2	3	1
9	1	2	3	4
10	3	2	4	1
11	1	3	2	4

Порогова величина сумарної зміни вагових коефіцієнтів експертів: 0.001

Рисунок 3.1 – Матриця оцінок альтернатив експертами та значення порогової величини сумарної зміни вагових коефіцієнтів

Далі необхідно пронормувати матрицю оцінок альтернатив експертами. Для цього потрібно знайти суму всіх оцінок (рисунок 3.2) та поділити кожну з оцінок на отриману величину для кожного фахівця окремо. Буде отримана зважена матриця оцінок альтернатив експертами (рисунок 3.3).

Сума оцінок експертів:

(index)	Values
0	10
1	10
2	10
3	10
4	10
5	10
6	10
7	10
8	10
9	10
10	10
11	10

Рисунок 3.2 – Сума оцінок кожного експерта

Зважена матриця оцінок альтернатив експертами:

(index)	0	1	2	3
0	0.1	0.3	0.2	0.4
1	0.2	0.3	0.4	0.1
2	0.1	0.4	0.3	0.2
3	0.1	0.3	0.4	0.2
4	0.1	0.4	0.3	0.2
5	0.2	0.1	0.3	0.4
6	0.3	0.2	0.1	0.4
7	0.1	0.2	0.4	0.3
8	0.4	0.2	0.3	0.1
9	0.1	0.2	0.3	0.4
10	0.3	0.2	0.4	0.1
11	0.1	0.3	0.2	0.4

Рисунок 3.3 – Зважена матриця оцінок альтернатив експертами

Далі необхідно обчислити початкову матрицю компетенцій експертів. Так як інформація, згідно якої можна встановити початкові значення даних величин, відсутня, вважається, що компетенції експертів однакові, тобто необхідно розділити одиницю на кількість експертів $= 1/12 = 0.083$ (рисунок 2.4).

Компетенції експертів:

(index)	0	1	2	3
0	0.08333	0.08333	0.08333	0.08333
1	0.08333	0.08333	0.08333	0.08333
2	0.08333	0.08333	0.08333	0.08333
3	0.08333	0.08333	0.08333	0.08333
4	0.08333	0.08333	0.08333	0.08333
5	0.08333	0.08333	0.08333	0.08333
6	0.08333	0.08333	0.08333	0.08333
7	0.08333	0.08333	0.08333	0.08333
8	0.08333	0.08333	0.08333	0.08333
9	0.08333	0.08333	0.08333	0.08333
10	0.08333	0.08333	0.08333	0.08333
11	0.08333	0.08333	0.08333	0.08333

Рисунок 3.4 – Початкова матриця компетенції експертів

Перша ітерація алгоритму методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів буде наведена у наступних абзацах та буде включати в себе рисунки 3.5-3.13.

Необхідно розрахувати, якими будуть оцінки експертів з урахуванням їх компетенцій (рисунок 3.5). Для цього потрібно помножити зважену матрицю оцінок експертів на їхні компетенції.

Відповіді експертів з урахуванням їх компетенцій:

(index)	0	1	2	3
0	0.00833	0.025	0.01667	0.03333
1	0.01667	0.025	0.03333	0.00833
2	0.00833	0.03333	0.025	0.01667
3	0.00833	0.025	0.03333	0.01667
4	0.00833	0.03333	0.025	0.01667
5	0.01667	0.00833	0.025	0.03333
6	0.025	0.01667	0.00833	0.03333
7	0.00833	0.01667	0.03333	0.025
8	0.03333	0.01667	0.025	0.00833
9	0.00833	0.01667	0.025	0.03333
10	0.025	0.01667	0.03333	0.00833
11	0.00833	0.025	0.01667	0.03333

Рисунок 3.5 – Матриця оцінок альтернатив експертами з урахуванням їхньої компетенції

Для подальших розрахунків необхідно знайти сумарні оцінки об’єктів (рисунок 3.6) та середньозважене відповідей експертів з урахуванням їх компетенцій (рисунок 3.7). Другий вектор буде використовуватись при розрахунку величини сумарної зміни вагових коефіцієнтів.

Сумарні оцінки об'єктів:

(index)	Values
0	0.17498
1	0.25834
2	0.29999
3	0.26665

Рисунок 3.6 – Вектор сумарної оцінки об’єктів.

Середньоарифметичне відповідей експертів із врахуванням їх компетентності:

(index)	Values
0	0.01458
1	0.02153
2	0.025
3	0.02222

Рисунок 3.7 – Вектор середньоарифметичного відповідей експертів з урахуванням їх компетентності

Далі була розрахована матриця відхилень відповідей від середньоарифметичних значень оцінок (рисунок 3.8).

Матриця відхилень відповідей від середньозважених:

(index)	0	1	2	3
0	0.00625	0.00347	0.00833	0.01111
1	0.00209	0.00347	0.00833	0.01389
2	0.00625	0.0118	0	0.00555
3	0.00625	0.00347	0.00833	0.00555
4	0.00625	0.0118	0	0.00555
5	0.00209	0.0132	0	0.01111
6	0.01042	0.00486	0.01667	0.01111
7	0.00625	0.00486	0.00833	0.00278
8	0.01875	0.00486	0	0.01389
9	0.00625	0.00486	0	0.01111
10	0.01042	0.00486	0.00833	0.01389
11	0.00625	0.00347	0.00833	0.01111

Рисунок 3.8 – Матриця відхилень відповідей експертів від середньозважених

Щоб визначити компетентність експертів, необхідно розрахувати суму модулів відхилень відповідей від середньозважених (рисунок 3.9) та нормалізувати їх значення (рисунок 3.10). Чим вища величина, тим меншою є компетентність експерта.

Сума модулів відхилень:

(index)	Values
0	0.02916
1	0.02778
2	0.0236
3	0.0236
4	0.0236
5	0.0264
6	0.04306
7	0.02222
8	0.0375
9	0.02222
10	0.0375
11	0.02916

Рисунок 3.9 – Вектор суми модулів відхилень

Сума модулів відхилень (нормалізована):

(index)	Values
0	0.08433
1	0.08034
2	0.06825
3	0.06825
4	0.06825
5	0.07634
6	0.12452
7	0.06426
8	0.10844
9	0.06426
10	0.10844
11	0.08433

Рисунок 3.10 – Вектор нормалізованої суми модулів відхилень

З рисунку 3.10 можна побачити, що найменшу компетентність має 6-ий експерт, а найвищу – 7-ий та 9-ий (чим вища величина, тим меншою є компетентність експерта).

Тепер необхідно розрахувати матрицю нових компетенцій експертів (рисунок 3.11) та нормувати її (рисунок 3.12), поділивши значення матриці нових значень компетенцій на їхню суму.

Нові значення компетенцій:

(index)	0	1	2	3
0	2.85768	2.85768	2.85768	2.85768
1	2.99964	2.99964	2.99964	2.99964
2	3.53093	3.53093	3.53093	3.53093
3	3.53093	3.53093	3.53093	3.53093
4	3.53093	3.53093	3.53093	3.53093
5	3.15644	3.15644	3.15644	3.15644
6	1.93521	1.93521	1.93521	1.93521
7	3.75023	3.75023	3.75023	3.75023
8	2.22213	2.22213	2.22213	2.22213
9	3.75023	3.75023	3.75023	3.75023
10	2.22213	2.22213	2.22213	2.22213
11	2.85768	2.85768	2.85768	2.85768

Рисунок 3.11 – Матриця значень нових компетенцій експертів

Нормовані нові значення компетенцій:

(index)	0	1	2	3
0	0.07863	0.07863	0.07863	0.07863
1	0.08253	0.08253	0.08253	0.08253
2	0.09715	0.09715	0.09715	0.09715
3	0.09715	0.09715	0.09715	0.09715
4	0.09715	0.09715	0.09715	0.09715
5	0.08685	0.08685	0.08685	0.08685
6	0.05325	0.05325	0.05325	0.05325
7	0.10319	0.10319	0.10319	0.10319
8	0.06114	0.06114	0.06114	0.06114
9	0.10319	0.10319	0.10319	0.10319
10	0.06114	0.06114	0.06114	0.06114
11	0.07863	0.07863	0.07863	0.07863

Рисунок 3.12 – Матриця нормованих значень компетенцій

Результати виконання першої ітерації такі (рисунок 3.13).

```
{ 'Iteration №1': [ 0.01458, 0.02153, 0.025, 0.02222 ] }
```

Рисунок 3.13 – Середньозважені значення оцінок альтернатив експертами

На всіх наступних ітераціях для розрахунків використовується нова матриця компетенцій експертів. Ітерації виконуються доти, доки величина сумарної зміни вагових коефіцієнтів (сума різниці значень середньозважених значень оцінок альтернатив поточної ітерації та попередньої). Якщо це число буде менше, ніж порогова величина зміни вагових коефіцієнтів (0.001), то виконання програми зупиняється.

Остання ітерація алгоритму методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів буде наведена у наступних абзацах та буде включати в себе рисунки 3.14-3.17.

На рисунку 3.14 можна побачити отриману матрицю нормованих значень на останній ітерації №7 та як змінювались компетенції протягом всіх ітерацій (рисунок 3.15).

Нормовані нові значення компетенцій:

(index)	0	1	2	3
0	0.11124	0.11124	0.11124	0.11124
1	0.08685	0.08685	0.08685	0.08685
2	0.11904	0.11904	0.11904	0.11904
3	0.11904	0.11904	0.11904	0.11904
4	0.11904	0.11904	0.11904	0.11904
5	0.08916	0.08916	0.08916	0.08916
6	0.00178	0.00178	0.00178	0.00178
7	0.11837	0.11837	0.11837	0.11837
8	0.00099	0.00099	0.00099	0.00099
9	0.11837	0.11837	0.11837	0.11837
10	0.00486	0.00486	0.00486	0.00486
11	0.11124	0.11124	0.11124	0.11124

Рисунок 3.14 – Матриця нормованих значень компетенцій експертів на останній ітерації

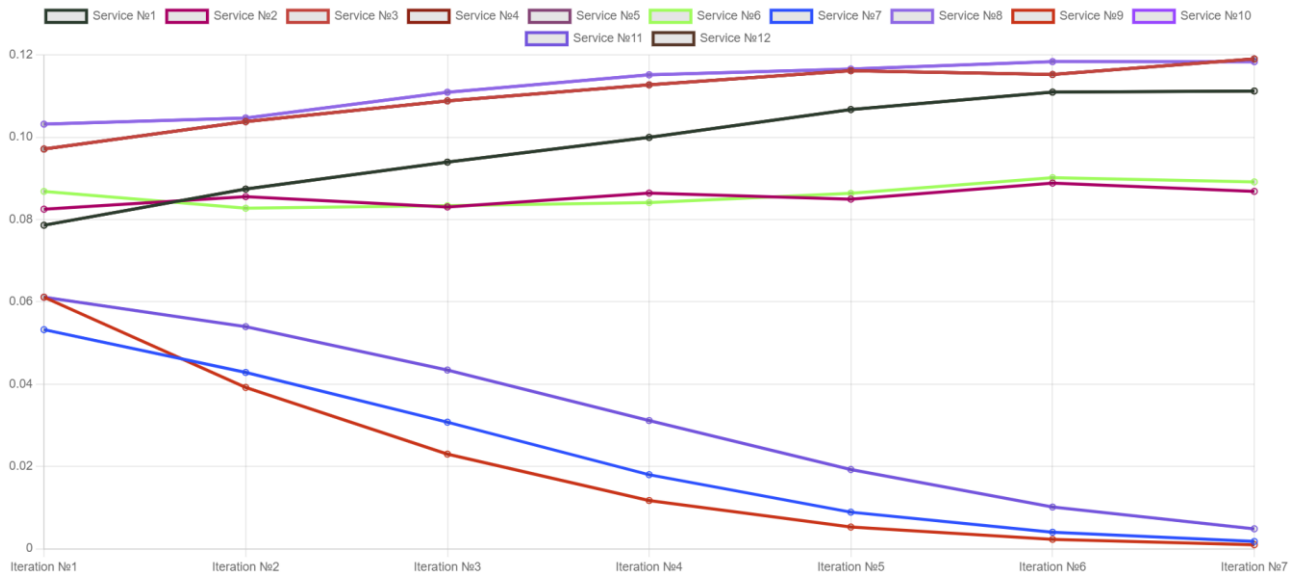

Рисунок 3.15 – Поітераційна зміна компетенцій

З рисунку 3.15 видно, що при нормалізації компетенцій пріоритет реалізації робіт змінювався. У результаті виконання програми були отримані результати, що найшвидше необхідно реалізувати роботи 2, 3 та 4. Робота ж під номером 8 має найменший пріоритет реалізації.

Можна порівняти, як змінювались вагові коефіцієнти протягом виконання програми в матричному вигляді (рисунок 3.16) та графічному (рисунок 3.17). Як тільки була досягнута вказана точність (0.001), програма завершила своє виконання.

```
'Iteration №1': [ 0.01458, 0.02153, 0.025, 0.02222 ],
'Iteration №2': [ 0.01318, 0.02199, 0.02567, 0.02249 ],
'Iteration №3': [ 0.01233, 0.02247, 0.02573, 0.0228 ],
'Iteration №4': [ 0.01153, 0.02277, 0.02581, 0.02323 ],
'Iteration №5': [ 0.01087, 0.02305, 0.02591, 0.0235 ],
'Iteration №6': [ 0.01036, 0.02328, 0.02588, 0.02382 ],
'Iteration №7': [ 0.01012, 0.02331, 0.02585, 0.02405 ]
```

Рисунок 3.16 – Узагальнені результати виконання програми

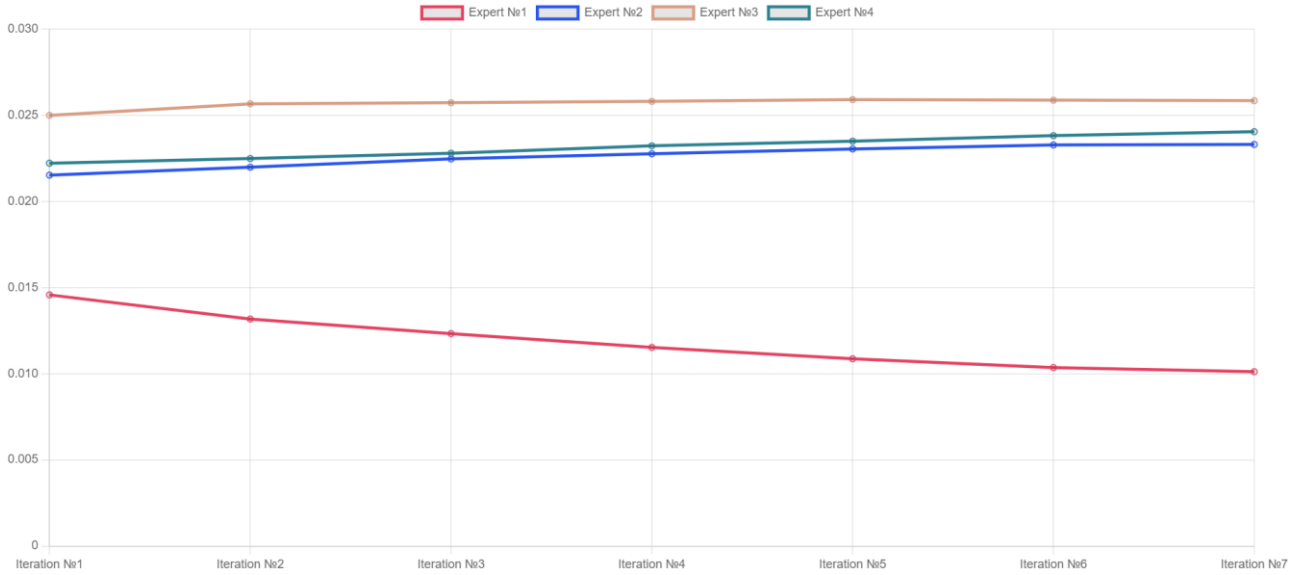

Рисунок 3.17 – Поітераційна зміна вагових коефіцієнтів

3.3 Задача визначення першочерговості виконання робіт

3.3.1 Постановка задачі

Нехай існує:

– $A = \{A_k \mid k = 1, \dots, n\}$ – множина альтернатив (робіт);

– $C = \{C_j \mid j = 1, \dots, m\}$ – множина критеріїв, за якими оцінюються альтернативи (роботи);

– $X = \{x_{kj} \mid k = 1, \dots, n; j = 1, \dots, m\}$ – множина альтернатив в просторі оцінок критеріїв (оцінок важливостей робіт за критеріями);

– $w = \{w_j \mid j = 1, \dots, m\}$ – множина вагових коефіцієнтів критеріїв.

Необхідно за інформацією про оцінки альтернатив за множиною критеріїв та за інформацією про порівняльну важливість критеріїв (у вигляді вагових коефіцієнтів критеріїв) визначити одну найкращу альтернативу або підмножину найкращих альтернатив з множини A.

3.3.2 Опис методу розв'язання

Після визначення всіх об’єктивних оцінок методом узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів застосовується метод TOPSIS (the Technique for Order Preferences by Similarity to an Ideal Solution), який використовується для розв’язання задач за умови, коли необхідно серед декількох альтернатив, які оцінюються за двома чи більше кількісними критеріями, знайти найкращий [2].

Метод TOPSIS впорядковує альтернативи у порядку першочерговості за важливістю. Його основна ідея полягає у концепціях пошуку компромісного розв’язку при виборі найкращої альтернативи, яка б була найближчою до позитивного ідеального розв’язку (скорочено PIS, або утопічної точки, оптимального розв’язку) та найвіддаленішим від негативного ідеального розв’язку (скорочено NIS, або антиутопічної точки). В якості розв’язку вибирається альтернатива, що є найкращою згідно побудованого впорядкування [3].

Оскільки для кожної альтернативи розраховується деяка величина (за заданою метрикою), то згідно цієї величини можна відсортувати аналізовані альтернативи і тим самим встановити їх пріоритетне упорядкування.

Значення складності алгоритму, отримане в результаті розрахунку нормалізації та зважування множини альтернатив, дорівнює $O(n^2)$. Складність позитивно-негативного ідеального рішення (PIS-NIS) і С відстані до кожної з альтернатив дорівнює $O(n)$ [10].

Нижче буде наведений алгоритм вирішення задачі [9]:

Крок 1. Обчислення нормалізованих оцінок альтернатив.

$$r_{kj}(x) = \frac{x_{kj}}{\sqrt{\sum_{k=1}^{n} x_{kj}^2}}, k = 1, \dots, n; j = 1, \dots m \qquad (3.2)$$

Крок 2. Обчислення зважених нормалізованих оцінок альтернатив:

$$v_{ij} = w_j \cdot r_{ij}, i = \overline{1, n}, j = \overline{1, m} \qquad (3.3)$$

де w_j – ваговий коефіцієнт критерія $C_j (\sum_{j=1}^{m} w_j = 1)$; r_{ij} – нормалізовані оцінки, що визначаються співвідношенням 3.2.

Крок 3. Побудова позитивної ідеальної точки PIS (утопічної точки) і негативної ідеальної точки NIS (антиутопічної точки):

$$PIS = A^+ = \left\{v_1^+(x), v_2^+(x), \ldots, v_j^+(x), \ldots, v_m^+(x)\right\} = \left\{\left(\max_k v_{kj}(x) \mid j \in C^+\right), \left(\min_k v_{kj} \mid j \in C^-\right) \mid k = 1, \ldots, n\right\} \quad (3.4)$$

$$NIS = A^- = \left\{v_1^-(x), v_2^-(x), \ldots, v_j^-(x), \ldots, v_m^-(x)\right\} = \left\{\left(\min_k v_{kj}(x) \mid j \in C^+\right), \left(\max_k v_{kj}(x) \mid j \in C^-\right) \mid k = 1, \ldots, n\right\} \quad (3.5)$$

Крок 4. Обчислення відстаней кожної альтернативи до позитивної ідеальної точки PIS (3.6) і негативної ідеальної точки NIS (3.7):

$$D_k^* = \sqrt{\sum_{j=1}^{m} \left[v_{kj}(x) - v_j^+(x)\right]^2}, \quad k = 1, \ldots, n \quad (3.6)$$

$$D_k^- = \sqrt{\sum_{j=1}^{m} \left[v_{kj}(x) - v_j^-(x)\right]^2}, \quad k = 1, \ldots, n \quad (3.7)$$

Крок 5. Встановлення наближеності кожної альтернативи до позитивної ідеальної точки PIS (подібності до PIS):

$$C_k^* = D_k^- / (D_k^* + D_k^-), k = 1, \ldots, n \quad (3.8)$$

$C_k^* \in [0,1] \quad \forall k = 1, \ldots, n.$

Крок 6. Впорядкування альтернатив множини $A = \{A_k \mid k = 1, \ldots, n\}$ за спаданням значення C_k^* - схожості із позитивною ідеальною точкою PIS. Найкраща альтернатива визначається наступним чином:

$$A^* = A_p \mid (C_p^* = \max_k C_k^*) \quad (3.9)$$

3.3.3 Застосування розробленого алгоритму на прикладі

В даному прикладі буде наведено виконання алгоритму TOPSIS для задачі визначення першочерговості виконання робіт.

Нехай існує продукт менеджер в IT-компанії, якому необхідно визначити пріоритет реалізації різних програмних покращень для застосунку. Є 4 альтернативи (різні програмні покращення) та 3 критерії, за якими необхідно визначити пріоритет реалізації, а також вагові коефіцієнти критеріїв, які виражають важливість кожного з них (табл. 3.1).

Таблиця 3.1 – Матриця оцінок альтернатив за трьома критеріями, а також вагові коефіцієнти критеріїв

Альтернативи	C1	C2	C3
A1	5	8	4
A2	7	6	8
A3	8	8	6
A4	7	4	6
Вага	0.3	0.4	0.3

За співвідношенням 3.2 була обчислена матриця нормалізованих оцінок альтернатив (табл. 3.2).

Таблиця 3.2 – Матриця нормалізованих оцінок альтернатив

Альтернативи	C1	C2	C3
A1	0.37	0.6	0.32
A2	0.51	0.45	0.65
A3	0.59	0.6	0.49
A4	0.51	0.3	0.49

За співвідношенням 3.3 була обчислена матриця зважених нормалізованих оцінок альтернатив (табл. 3.3).

Таблиця 3.3 – Матриця зважених нормалізованих оцінок альтернатив

Альтернативи	C1	C2	C3
A1	0.11	0.24	0.10
A2	0.15	0.18	0.19
A3	0.18	0.24	0.15
A4	0.15	0.12	0.15

За співвідношеннями 3.4 та 3.5 були встановлені позитивна ідеальна точка (PIS) та негативна ідеальна точка (NIS) за кожним із критеріїв (табл. 3.4). Зелений колір означає NIS, а жовтий – PIS.

Таблиця 3.4 – Визначення NIS та PIS

Альтернативи	C1	C2	C3
A1	0.11	0.24	0.10
A2	0.15	0.18	0.19
A3	0.18	0.24	0.15
A4	0.15	0.12	0.15

За співвідношеннями 3.6 та 3.7 були розраховані відстані кожної з альтернатив до позитивної ідеальної точки (PIS) – S^+, а також до негативної ідеальної точки (NIS) – S^- (табл. 3.5).

Таблиця 3.5 – Відстані S^+ та S^-

Альтернативи	S^+	S^-
A1	0.1175	0.1193
A2	0.0635	0.1223
A3	0.0487	0.1446
A4	0.1307	0.0655

Тепер необхідно за співвідношенням 3.8 встановити наближеності кожної альтернативи до позитивної ідеальної точки PIS (подібності до PIS) (табл. 3.6).

Таблиця 3.5 – Наближеності $C_i^*, i = 1, 2, 3, 4$

Альтернативи	C^*
A1	0.5037
A2	0.6581
A3	0.7482
A4	0.3340

Отже, за відношенням 3.9 необхідно визначити порядок виконання робіт. Робота з найбільшою наближеністю до позитивної ідеальної точки (PIS) буде мати найвищий пріоритет реалізації.

Менеджеру необхідно реалізувати програмне покращення №3 в першу чергу. Загалом же порядок реалізації програмних покращень для застосунку такий: $A_3 \rightarrow A_2 \rightarrow A_1 \rightarrow A_4$.

Висновок до розділу

В даному розділі були наведені математичні постановки двох типів задач:

– визначення значень критеріїв оцінки робіт на основі експертних оцінок методом узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів;

– визначення першочерговості виконання робіт методом TOPSIS.

До кожної з задач була наведена послідовність кроків алгоритму, яким вона розв'язується, а також приклад використання алгоритмів на практичних задачах.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

В якості опису засобів розробки буде описана мова програмування, що була використана при створенні застосунку, а також використані фреймворки цієї мови та середовище розробки.

Використана мова програмування – JavaScript. Особливості даної мови, що слугували причиною її вибору [11]:

– JavaScript – найпопулярніша високорівнева мова програмування, яка підтримує імперативний та функціональний підходи. Це мова з динамічною типізацією, яка застосовується для запису послідовних операцій – «скриптів». «Скрипти» інтерпретуються, а не компілюються, що значно пришвидшує виконання програми та надає мові JavaScript перевагу над іншими.

– В теперішні часи майже завжди компанії не обмежуються роботою із JavaScript лише в браузері, а й розробляють серверну частину застосунку на платформі Node.js (середовище виконання JavaScript-коду), що робить розробку значно дешевшою та дає змогу наймати робітників, які спеціалізуються не тільки на розробці клієнтської частини, а й серверної.

– Практично кожен веб-застосунок побудований із використанням трьох основних технологій – HTML (для розмітки елементів на сторінкі), CSS (для стилізіації елементів) та JavaScript («мозок» розробки, який робить сторінку «живою» та відповідає за її інтерактивність і взаємодію з користувачем).

Велика кількість бібліотек, з якими надає можливість працювати пакетний менеджер npm, значно полегшують розробку застосунку, використовуючи ресурси мови та її конкретні бібліотеки. Мова постійно оновлюється та має зрозумілий та зручний синтаксис, що полегшує роботу з підтримки застосунку та додавання нової функціональності в подальшому.

Візуальна частина застосунку написана за допомогою фреймворку мови JavaScript – React. React – це найпопулярніша бібліотека для створення

інтерфейсів користувача, яка використовується при розробці односторінкових застосунків SPA (Single Page Application) [12].

Серверна частина застосунку реалізована за допомогою Node.js – платформи з відкритим кодом для виконання високопродуктивних мережевих застосунків, які використовують браузерний «двигун» V8 з Google Chrome, написаних мовою JavaScript [13]. Запити до бази даних реалізовуються за допомогою фреймворку Express.js – це швидкий, гнучкий, мінімалістичний фреймворк для веб-застосунків, побудованих на Node.js [14].

В якості середовища розробки було обрано WebStorm – це інтегроване середовище розробки для JavaScript і пов’язаних з нею технологіями. Він дозволяє автоматизувати рутинну роботу і легко виконувати складні задачі, концентруючись виключно на важливому. Можна виділити декілька переваг WebStorm [15]:

– розумний редактор коду, який підказує кращі рішення з точки зору пришвидшення виконання програмного коду;

– вбудовані інструменти для розробників (можливість встановлювати npm-пакети через консоль самого середовища розробки, дуже зручний дебагер, покращене середовище виконання unit-тестів та інше);

– швидка навігація та пошук (по всьому проекту, в певній директорії в проекті тощо);

– ефективна команда робота (вбудована можливість роботи з git-ом – розподіленою системою контролю версій, яка дозволяє відстежувати історію розробки ПЗ і спільно працювати над складними проектами з будь-якої точки світу [16]);

– кастомізація (можливість встановлення різноманітних розширень, які дозволяють користувачу повністю налаштувати середовище розробки під себе – починаючи з кольору самого WebStorm до заміни іконок всіх файлів проекту).

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Мінімальні вимоги до технічного забезпечення представимо у вигляді переліку необхідної конфігурації як для комп'ютера-клієнта, так і для комп'ютера-сервера.

Системні вимоги до конфігурації комп'ютера-клієнта наведені в таблиці 4.1.

Таблиця 4.1 – Системні вимоги до комп'ютера-клієнта

Процесор	2 GHz та більше
Оперативна пам'ять	4 GB та більше
Вільне місце на диску	500MB мінімум
Оперативна система	Windows 8.1/10/11, macOS 10.14+
Пристрої взаємодії з користувачем	Клавіатура та миша (тачпад)

Системні вимоги до конфігурації комп'ютера-сервера наведені в таблиці 4.2.

Таблиця 4.2 – Системні вимоги до комп'ютера-сервера

Процесор	2 GHz та більше
Оперативна пам'ять	1GB та більше
Віртуальна пам’ять	Вдвічі більше оперативної
Вільне місце на диску	200 MB мінімум
Оперативна система	Windows 8.1/10/11, macOS 10.14+

4.3 Архітектура програмного забезпечення

В якості архітектури програмного забезпечення був обраний один з архітектурних шаблонів – дворівнева клієнт-серверна архітектура. В її основі лежить взаємодія двох програмних модулів – клієнтського та серверного.

На рисунках 4.1-4.2 наведена схема архітектури розробленого програмного забезпечення.

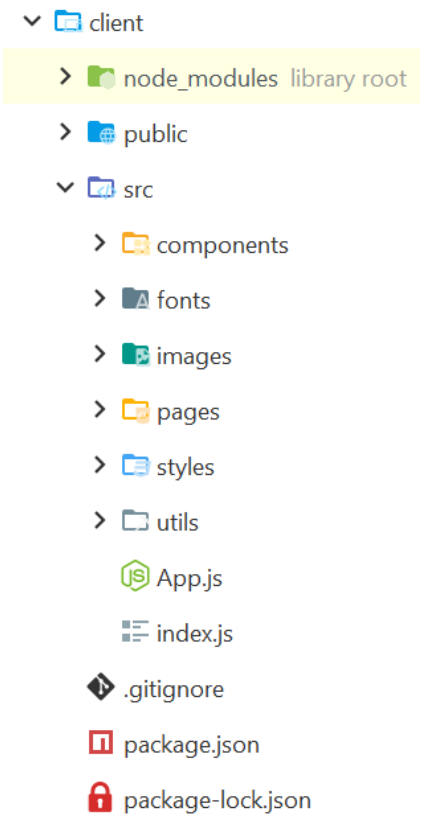

Рисунок 4.1 – Схема архітектури клієнтської частини розробленого програмного забезпечення

Опис компонентів клієнтської частини застосунку:

a) папка node_modules містить всі встановлені пакети та бібліотеки, потрібні для роботи та розробки клієнтської частини застосунку;

b) папка public містить статичні файли, такі як index.html файли JavaScript бібліотек, картинки та інші іконки;

c) папка src містить основні папки роботи програми:

1) components містить компоненти застосунку – частини програмного коду, які можуть перевикористовуватись у різних частинах застосунку. В даній папці містяться також CSS стилі компонентів – візуальне оформлення;
2) images містить усі картинки, іконки у різних форматах: .jpg, .png, .svg тощо;
3) fonts містить підключені шрифти, які використовуються в застосунку;

4) pages містить сторінки застосунку (в нашому випадку це сторінки для роботи з експертами, роботами, критеріями, оцінками, алгоритмами та головна сторінка);
5) styles містить усі CSS стилі застосунку;
6) utils містить допомоміжні функції застосунку;
7) файл App.js є точкою входу до застосунку;

d) файл .gitignore – це допоміжний файл, в якому вказуються всі файли та папки проекту, які не потрібно завантажувати в git;

e) файл package.json містить у собі інформацію про застосунок: назву, версію, залежності та інше;

f) файл package-lock.json містить дерево залежностей, що включає всі встановлені пакети та їх версії.

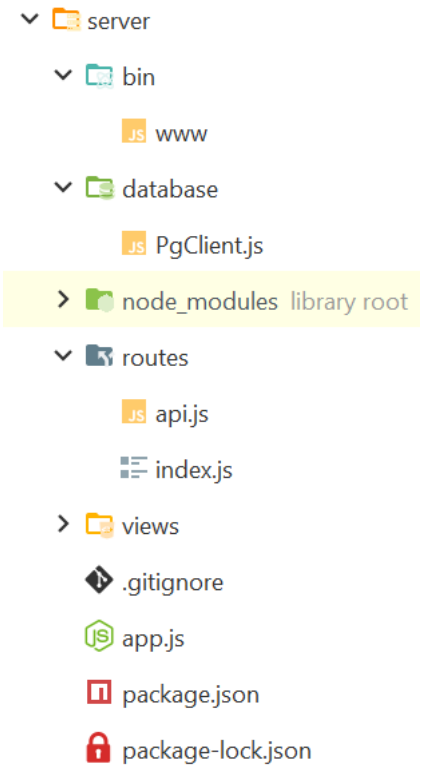

Рисунок 4.2 – Схема архітектури серверної частини розробленого програмного забезпечення

Опис компонентів клієнтської частини застосунку:

a) папка bin містить файл www, який створює сервер;

b) папка database містить файл PgClient, який містить в собі функції звернення до бази даних PostgresSQL;

c) папка node_modules містить всі встановлені пакети та бібліотеки, потрібні для роботи та розробки серверної частини застосунку;

d) папка routes містить в собі два файли:

1) index.js – є основним файлом серверної частини застосунку, який викликає всі інші функції;
2) app.js – містить опис роутів (назва роуту та функція, яка буде викликатись при його виклику);

e) папка views містить файли відображення відповіді серверу на коректний запит або такий, що відбувся з помилкою;

f) файл .gitignore – це допоміжний файл, в якому вказуються всі файли та папки проекту, які не потрібно завантажувати в git.

g) файл app.js запускає сервер;

h) файл package.json містить у собі інформацію про застосунок: назву, версію, залежності та інше;

i) файл package-lock.json містить дерево залежностей, що включає всі встановлені пакети та їх версії.

4.3.1 Діаграма класів (функціональних компонентів)

Сучасні застосунки, написані за допомогою фреймворку React.js, більше не використовують класові компоненти, натомість використовуються функціональні. Отже, в даному підрозділі буде наведений опис функціональних компонентів.

UML діаграма наведена у графічних матеріалах, лист 3 (Д3 - Діаграма функціональних компонентів), де зображене статичне представлення структури моделі, що відображає статичні елементи, такі як: функціональні компоненти, їх зміст та відношення між ними.

Таблиця 4.3 – Опис функціональних компонентів застосунку

Назва	Опис
App	Основний компонент застосунку, який визначає шлях до кожної із сторінок, а також обертає кожну з них хедером та футером
MainPage	Компонент головної сторінки застосунку, що містить опис самого веб-застосунку
CriteriaPage	Компонент сторінки, на якій можна додавати, редагувати чи видаляти критерії
ExpertPage	Компонент сторінки, на якій можна додавати, редагувати чи видаляти експертів
TaskPage	Компонент сторінки, на якій можна додавати, редагувати чи видаляти роботи
ValuePage	Компонент сторінки, на якій можна вносити інформацію по оцінкам об’єктивних та суб’єктивних критеріїв
TopsisPage	Компонент сторінка для виконання розрахунків методом TOPSIS, отримання результатів, а також виведення логів
AddModal	Компонент для створення експертів, критеріїв, робіт тощо
Alert	Компонент для створення сповіщень різного типу: про успішне додавання, видалення чи редагування даних тощо
DeleteModal	Компонент для підтвердження видалення даних з бази даних
Footer	Компонент відображення футеру сторінки
Header	Компонент відображення хедеру сторінки, що містить в собі компоненти NavBar та LogoTitle

Назва	Опис
LogoTitle	Компонент відображення логотипу застосунку
NavBar	Компонент для формування навігаційного меню застосунку
TaskCriteriaCreateForm	Компонент для створення записів про оцінки критеріїв
NonQualityCriteriaTable	Компонент для формування таблиці виведення інформації про суб’єктивні критерії
QualityCriteriaTable	Компонент для формування таблиці виведення інформації про об’єктивні критерії
Topsis	Компонент для виконання розрахунків методом TOPSIS та виведення результатів

4.3.2 Діаграма компонентів

Структурна діаграма компонентів наведена на рисунку 4.3.

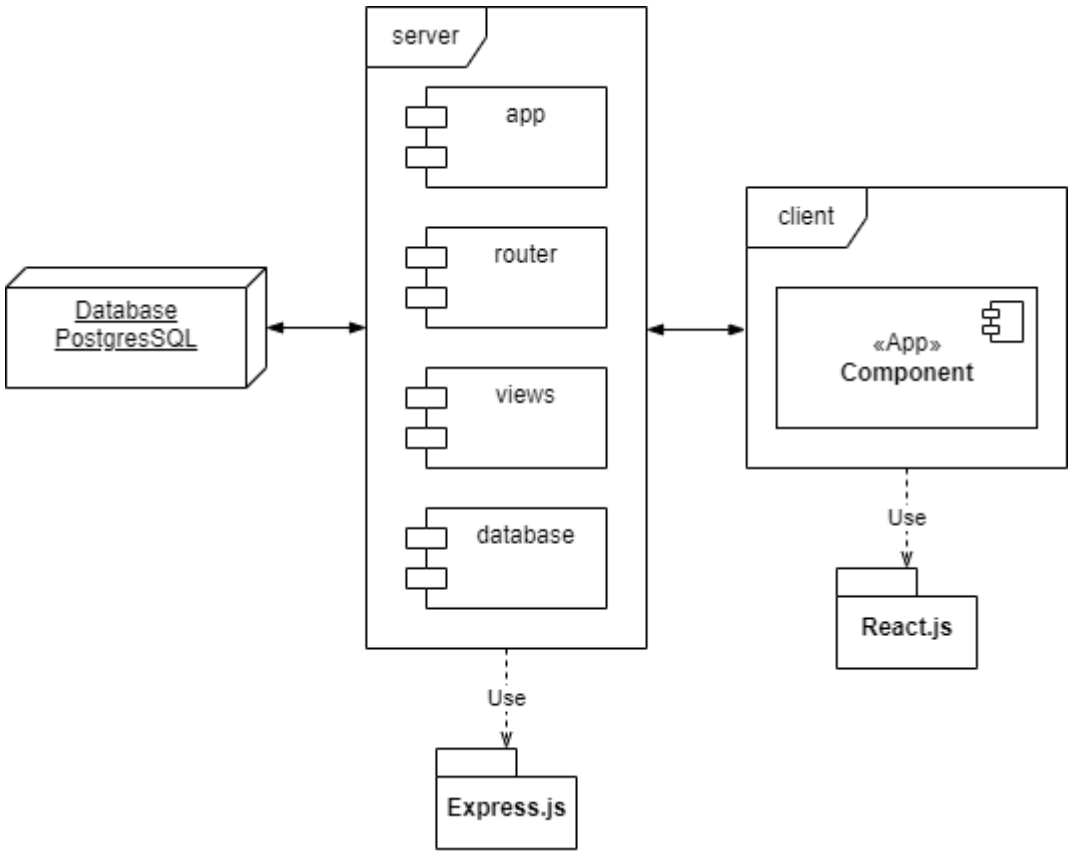

Рисунок 4.3 – Діаграма компонентів

Діаграма компонентів складається з двох шарів, а саме з частини клієнта та серверу. Клієнт має в собі один основний компонент – «App», який відповідає за створення інтерфейсу для користувача та працює безпосередньо з React.js, який відповідає за відображення компонентів інтерфейсу.

Серверна частина виконує роль посередника між базою даних та клієнтом, що здійснюється за допомогою вказаних модулів «app», «router», «views» та «database».

4.3.3 Специфікація функцій

Далі для детального опису специфікації функцій дані будуть надані по кожному з функціональних компонентів у табличному вигляді у форматі: назва функції, вхідні параметри, вихідні параметри та опис функції (таблиці 4.4 – 4.21). В таблицях 4.22-4.24 буде наведений опис допоміжних функцій.

Таблиця 4.4 – Опис специфікації функцій функціонального компоненту App.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.5 – Опис специфікації функцій функціонального компоненту MainPage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.6 – Опис специфікації функцій функціонального компоненту CriteriaPage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchCriteria	–	data – список усіх критеріїв	Функція для отримання списку критеріїв з бази даних.
onUpdate	name – назва критерію, id – унікальний ідентифікатор критерію	–	Функція для оновлення унікального ідентифікатору критерію та його імені.
onSubmit	event – подія, що відбулась при натисканні кнопки	–	Функція для додавання критерію до бази даних.
onCancel	–	–	Функція для видалення назви критерію, що вводилась при його редагуванні.
onInput	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження назви критерію, що вводиться при її редагуванні.
onChange	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження типу критерію, що змінюється при його редагуванні.
onDelete	–	–	Функція для видалення критерію з бази даних.

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
onInputTask	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження назви критерію, що вводиться при її додаванні.
onSubmitUpdate	name – назва критерію, currType – поточний тип критерію	–	Функція для оновлення критерію в базі даних.
onOpenDeleteModal	id – унікальний ідентифікатор критерію	–	Функція для відображення модального вікна, що підтверджує видалення та збереження унікального ідентифікатору критерію.
onCloseAlert	–	–	Функція для закриття сповіщення.
onHideDelete	–	–	Функція для закриття модального вікна, що підтверджує видалення критерію.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.7 – Опис специфікації функцій функціонального компоненту ExpertPage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchExperts	–	data – список усіх експертів	Функція для отримання списку експертів з бази даних.
onUpdate	name – ім’я експерта, id – унікальний ідентифікатор експерта	–	Функція для оновлення унікального ідентифікатору експерта та його імені.
onSubmit	event – подія, що відбулась при натисканні кнопки	–	Функція для додавання експерта до бази даних.
onDelete	–	–	Функція для видалення експерта з бази даних.
onHideAdd	–	–	Функція для закриття модального вікна з формою для додавання експерта до бази даних.
onCloseAlert	–	–	Функція для закриття сповіщення.
onSubmitUpdate	name – ім’я експерта	–	Функція для оновлення експерта в базі даних.
onOpenAddModal	id – унікальний ідентифікатор експерта	–	Функція для відображення модального вікна з формою для додавання експерта до бази даних.

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
onHideDelete	–	–	Функція для закриття модального вікна, що підтверджує видалення експерта.
onOpenDelete-Modal	id – унікальний ідентифікатор експерта	–	Функція для відображення модального вікна, що підтверджує видалення та збереження унікального ідентифікатору експерта.
onCancel	–	–	Функція для закриття модального вікна з формою для додавання експерта до бази даних.
onInputExpert	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження імені експерта, що вводиться при його додаванні.
onInput	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження імені експерта, що вводиться при його редагуванні.
onCancelExpert	–	–	Функція для видалення імені експерта, що вводилось при його редагуванні.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.8 – Опис специфікації функцій функціонального компоненту TaskPage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchTasks	–	data – список усіх робіт	Функція для отримання списку робіт з бази даних.
onUpdate	name – назва роботи, id – унікальний ідентифікатор роботи	–	Функція для оновлення унікального ідентифікатору роботи та її назви.
onSubmit	event – подія, що відбулась при натисканні кнопки	–	Функція для додавання роботи до бази даних.
onCancel	–	–	Функція для видалення назви роботи, що вводилась при її редагуванні.
onInput	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження назви роботи, що вводилась при її редагуванні.
onInputTask	event – подія, що відбулась при натисканні кнопки	–	Функція для збереження назви роботи, що вводилась при її додаванні.
onDelete	–	–	Функція для видалення роботи з бази даних.

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
onCancelTask	–	–	Функція для видалення назви роботи, що вводилась при його редагуванні.
onHideDelete	–	–	Функція для закриття модального вікна, що підтверджує видалення роботи.
onHideAdd	–	–	Функція для закриття модального вікна з формою для додавання роботи до бази даних.
onOpenDeleteModal	id – унікальний ідентифікатор роботи	–	Функція для відображення модального вікна, що підтверджує видалення та збереження ідентифікатору роботи.
onSubmitUpdate	name – назва роботи	–	Функція для оновлення роботи в базі даних.
onOpenAddModal	id – унікальний ідентифікатор роботи	–	Функція для відображення модального вікна з формою для додавання роботи до бази даних.
onCloseAlert	–	–	Функція для закриття сповіщення.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.9 – Опис специфікації функцій функціонального компоненту ValuePage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchTasksCriteria	–	data – список усіх оцінок критеріїв	Функція для отримання списку оцінок критеріїв з бази даних.
onOpenAddModal	–	–	Функція для відкриття модального вікна з формою для додавання оцінок критеріїв.
onHide	–	–	Функція для закриття модального вікна з формою для додавання оцінок критеріїв.
onCloseAlert	–	–	Функція для закриття сповіщення.
onSubmit	qualityCriteria – масив об’єктивних оцінок критеріїв, nonQualityCriteria – масив суб’єктивних оцінок критеріїв, taskId – унікальний ідентифікатор оцінюваної роботи	–	Функція для додавання оцінок за усіма критеріями (об’єктивними та суб’єктивними) по усім роботам до бази даних.

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
getEmptyNon-QualityDataObject	object – об’єкт	–	Функція для створення пустого об’єкту для подальшого збереження інформації за певним патерном.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.10 – Опис специфікації функцій функціонального компоненту ValuePage.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchTasks-Criteria	–	data – список усіх оцінок критеріїв	Функція для отримання списку оцінок критеріїв з бази даних.
calculateExpert-ValuesFunction	tasks – масив з оцінками суб'єктивних критерів	–	Функція для розрахунку кількісних оцінок суб’єктивних критеріїв.
getEmptyNon-QualityData-Object	object – об’єкт	–	Функція для створення пустого об’єкту для подальшого збереження інформації за певним патерном.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.11 – Опис специфікації функцій функціонального компоненту AddModal.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
renderName	–	–	Функція для відображення коректної назви у формі.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.12 – Опис специфікації функцій функціонального компоненту Alert.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
renderHeader	–	–	Функція для відображення коректної назви сповіщення.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.13 – Опис специфікації функцій функціонального компоненту DeleteModal.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.14 – Опис специфікації функцій функціонального компоненту Footer.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.15 – Опис специфікації функцій функціонального компоненту Header.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.16 – Опис специфікації функцій функціонального компоненту LogoTitle.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.17 – Опис специфікації функцій функціонального компоненту NavBar.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.18 – Опис специфікації функцій функціонального компоненту TaskCriteriaCreateForm.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
fetchTasks	–	data – список робіт	Функція для отримання списку робіт з бази даних.
fetchExperts	–	data – список експертів	Функція для отримання списку експертів з бази даних.
fetchCriteria	–	data – список критеріїв	Функція для отримання списку критеріїв з бази даних.
onChangeCriteria	array – масив критеріїв	–	Функція для формування списку унікальних ідентифікаторів критеріїв.
onChangeTask	event – подія, що виникає при виборі роботи у селекті	–	Функція для вибору роботи.
onClick	–	–	Функція для формування форми введення оцінок об’єктивних та суб’єктивних критеріїв.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.19 – Опис специфікації функцій функціонального компоненту NonQualityCriteriaTable.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
handleInputChange	event – подія, що виникає при введенні оцінки, expertId – унікальний ідентифікатор експерта, criteriaId – унікальний ідентифікатор критерію	–	Функція для збереження інформації про оцінки суб’єктивних критеріїв.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.20 – Опис специфікації функцій функціонального компоненту QualityCriteriaTable.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
handleInputChange	event – подія, що виникає при введенні оцінки, id – унікальний ідентифікатор критерію	–	Функція для збереження інформації про оцінки об’єктивних критеріїв.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.21 – Опис специфікації функцій функціонального компоненту Topsis.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
handleInputChange	event – подія, що виникає при введенні оцінки, ctiteriaId – унікальний ідентифікатор критерію	–	Функція для збереження інформації про оцінки всіх критеріїв (об’єктивних та суб’єктивних).
topsis	–	–	Функція для проведення розрахунків та обчислення задачі з визначення першочерговості виконання робіт методом TOPSIS.
onChangeCriteria	event – подія, що виникає при виборі значення з селекту, criteriaId – унікальний ідентифікатор критерію	–	Функція для збереження значення критерію (на максимум чи на мінімум) для проведення подальших розрахунків методом TOPSIS.
render	–	–	Функція для відображення компонентів на сторінці веб-застосунку.

Таблиця 4.22 – Опис специфікації функцій допоміжного класу ArrayUtil.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
setEmptyArrayOnEmpty	object – об’єкт, field – поле об’єкту	–	Функція для створення пустого об’єкту з заданим ключем та значенням пустого масиву.

Таблиця 4.23 – Опис специфікації функцій допоміжного класу ObjectUtil.js

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
setEmptyObjectTo	object – об’єкт, field – поле об’єкту	–	Функція для створення пустого об’єкту з заданим ключем та значенням пустого об’єкту.
getClone	object – об’єкт	object – склонований об’єкт	Клонування об'єкту.
isEmpty	object – об’єкт	Boolean (true/false) – пустий чи ні	Перевірка на те, чи пустий об’єкт.

Таблиця 4.24 – Опис специфікації інших допоміжних функцій

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
calculate-ExpertValues	C – матриця оцінок альтернатив експертами, expertsAmount – кількість експертів, servicesAmount – кількість робіт	sumRates – загальний поітераційний результат виконання алгоритму, allCoefsMatrixNorm – нормована матриця коефіцієнтів	Функція виконання методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів.
parseFloatTo-Fixed	value – кількість знаків після коми	Нормоване число	Функція для округлення числа до заданої кількості знаків після коми.
getExpertsRate	minRate – мінімальна оцінка, maxRate – максимальна оцінка	Випадкова оцінка	Функція для генерації випадкових оцінок експертів.
getIteration	iteration – номер ітерації	Поточна ітерація	Функція отримання поточної ітерації.
transformData-ToMatrix	data – вхідна матриця	Транспонована матриця	Функція транспонування матриці.

Назва функції	Вхідні параметри	Вихідні параметри	Опис функції
calculateTopsis	alternatives – матриця альтернатив, weights – вагові коефіцієнти оцінок критеріїв, tasks – масив робіт, criteria – масив критеріїв	normalizedAlternatives – матриця нормалізованих альтернатив, normalizedWeighed-Alternatives – матриця зважених нормалізованих альтернатив, PIS – вектор позитивних ідеальних точок, NIS – вектор негативних ідеальних точок, distances – вектор відстаней до позитивної ідеальної точки, order – вектор робіт, упорядкованих за зменшенням пріоритетності реалізації	Функція для визначення першочерговості виконання робіт методом TOPSIS.

4.4 Опис звітів

Звітом щодо виконання першого алгоритму (метод узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів) є матриця кількісних оцінок критеріїв, сформована з суб’єктивних оцінок критеріїв експертами (рис. 4.4).

Назва роботи	Симпатичність	Потрібність	Складність
Робота №1	0.06265	0.06138	0.07597
Робота №2	0.03789	0.08434	0.07777
Робота №3	0.04142	0.06666	0.09191
Робота №4	0.04791	0.06519	0.0869

Рисунок 4.4 – Результати виконання методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів

Звітом щодо виконання другого алгоритму (визначення першочерговості виконання робіт методом TOPSIS) є таблиця із зазначеним порядком реалізації робіт у порядку спадання їх пріоритетності (важливості) із вказанням числового значення важливості (рис. 4.5).

Порядок виконання робіт:

Порядок виконання	1	2	3	4
Назви робіт	Робота №2	Робота №4	Робота №1	Робота №3
Ваги	0.99715	0.37649	0.11766	0.00549

Рисунок 4.5 – Результати виконання методу TOPSIS

Висновок до розділу

В даному розділі було описане програмне та технічне забезпечення системи, була наведена структура самої системи та вказана специфікація кожного функціонального компоненту та функцій, що реалізовані у системі. А також наведений приклад звітів, які є результатом виконання програмного застосунку.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Для користування веб-застосунком користувачу достатньо базових знань на рівні користувача ПК. Застосунок має інтуїтивно зрозумілий інтерфейс. Для докладного та структурованого опису роботи з програмою надано покрокове керівництво користувача, яке містить опис роботи з програмним продуктом.

При переході на веб-сторінку користувач побачить стартову головну сторінку з описом самого веб-застосунку та меню навігації з назвами всіх основних сторінок:

1. «TOPSIS»;
2. «Експерти»;
3. «Критерії»;
4. «Роботи»;
5. «Оцінки».

В цьому розділі буде описана кожна зі сторінок більш детально.

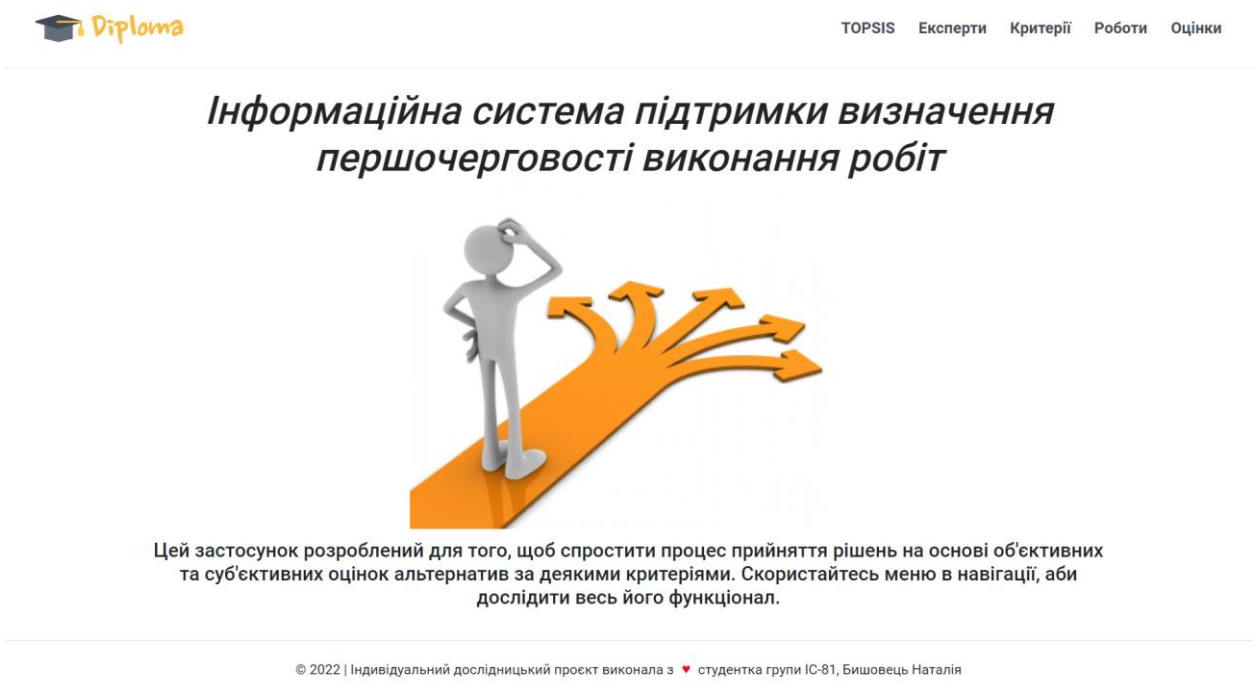

Рисунок 5.1 – Головна стартова сторінка веб-застосунку

1. Опис функціоналу сторінки «Експерти»

На рисунку 5.2 можна побачити базове відображення інформації про усіх експертів з бази даних: їхнє ім’я та порядковий номер у списку. Для кожного експерта є можливість зміни імені, а також видалення його з бази даних, створення чи додавання. Кожна функціональність буде описана окремо.

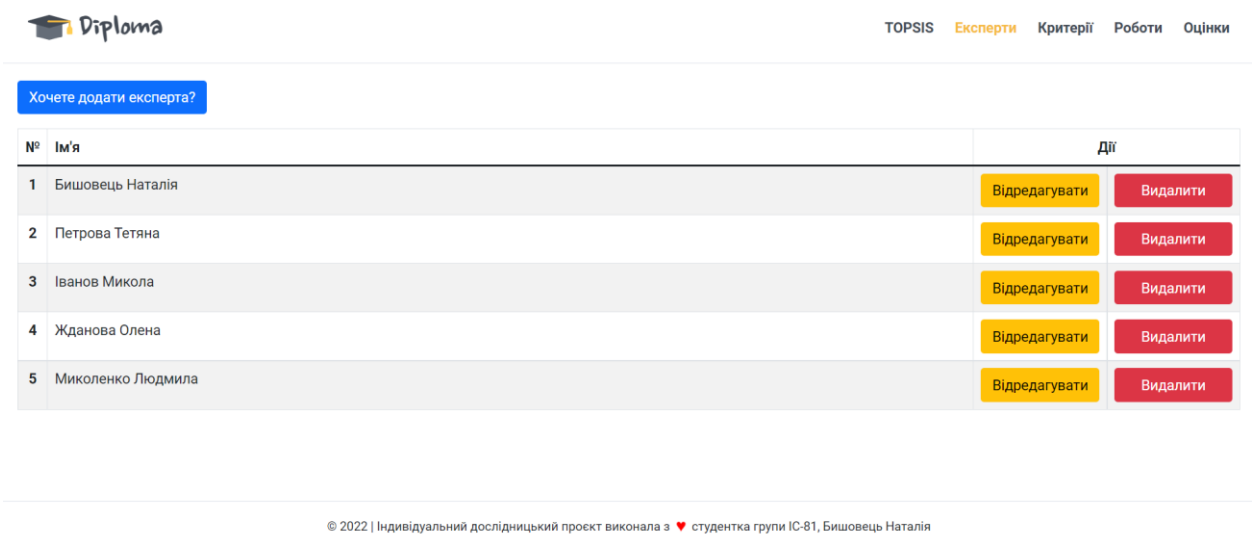

Рисунок 5.2 – Сторінка «Експерти»

При кліку на кнопку «Хочете додати експерта?» з’являється модальне вікно з формою для створення експерта, в якому користувач може ввести ім’я нового експерта та підтвердити його додавання до бази даних (рис. 5.3).

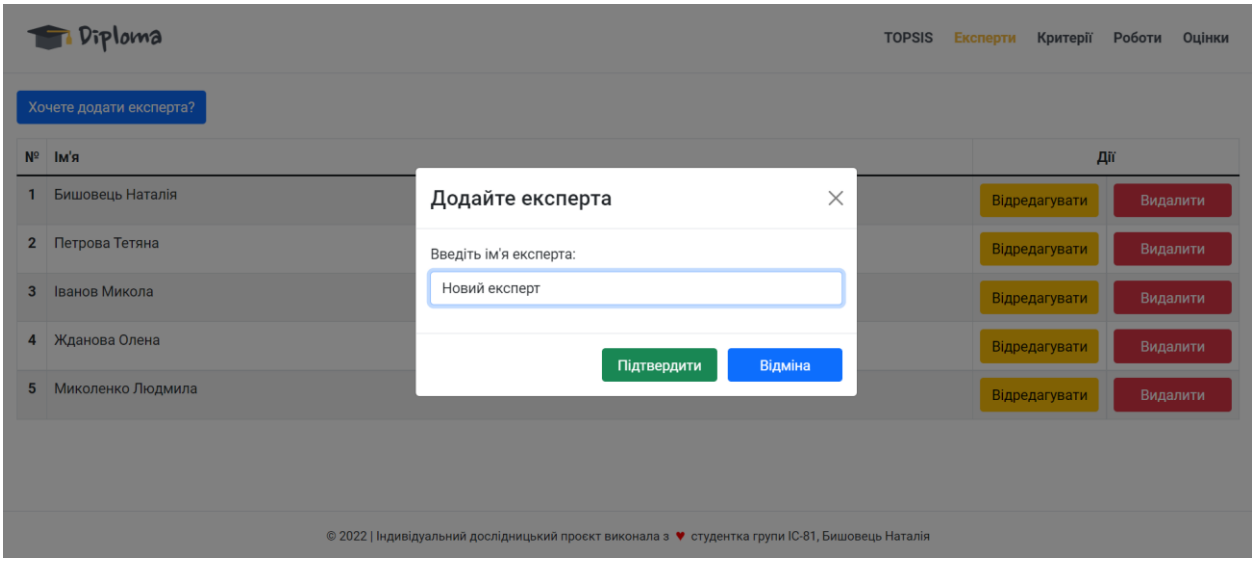

Рисунок 5.3 – Модальне вікно для додавання експерта до бази даних

Після введення імені (нехай буде «Новий експерт») та кліку на кнопку «Підтвердити», з’являється сповіщення про успішне додавання експерта до бази даних (рис. 5.4). Також список усіх експертів оновлюється і можна побачити новоствореного експерта останнім у списку (рис. 5.5).

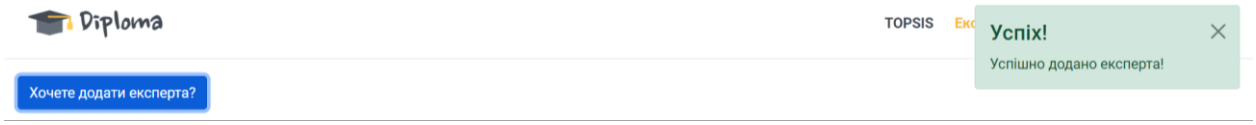

Рисунок 5.4 – Сповіщення про успішне додавання експерта до бази даних

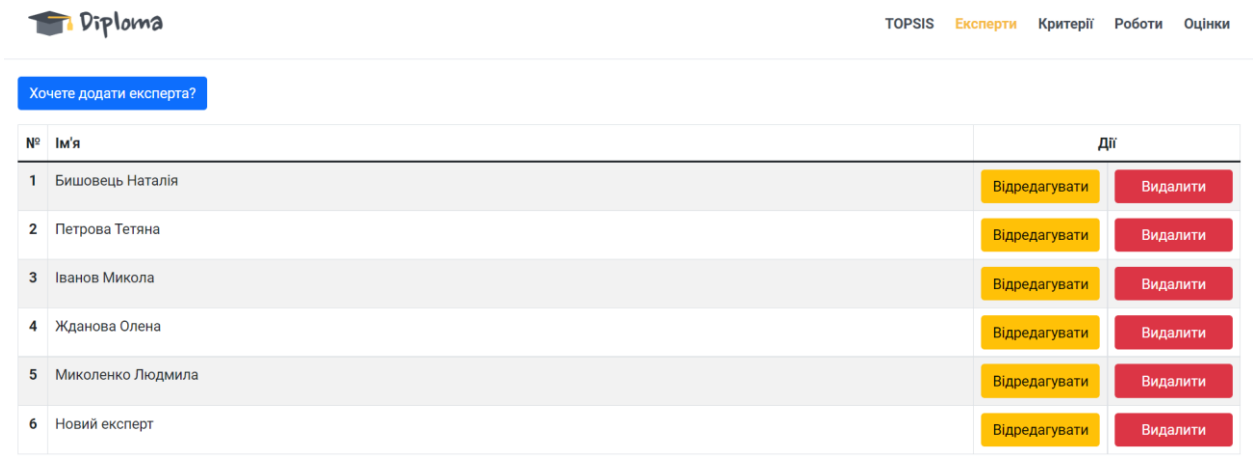

Рисунок 5.5 – Оновлений список усіх експертів

У випадку, якщо на стороні сервера виникла помилка, та запит не обробився, з’являється сповіщення, яке говорить про те, що щось пішло не так і користувачу краще повторити спробу пізніше (рис. 5.6).

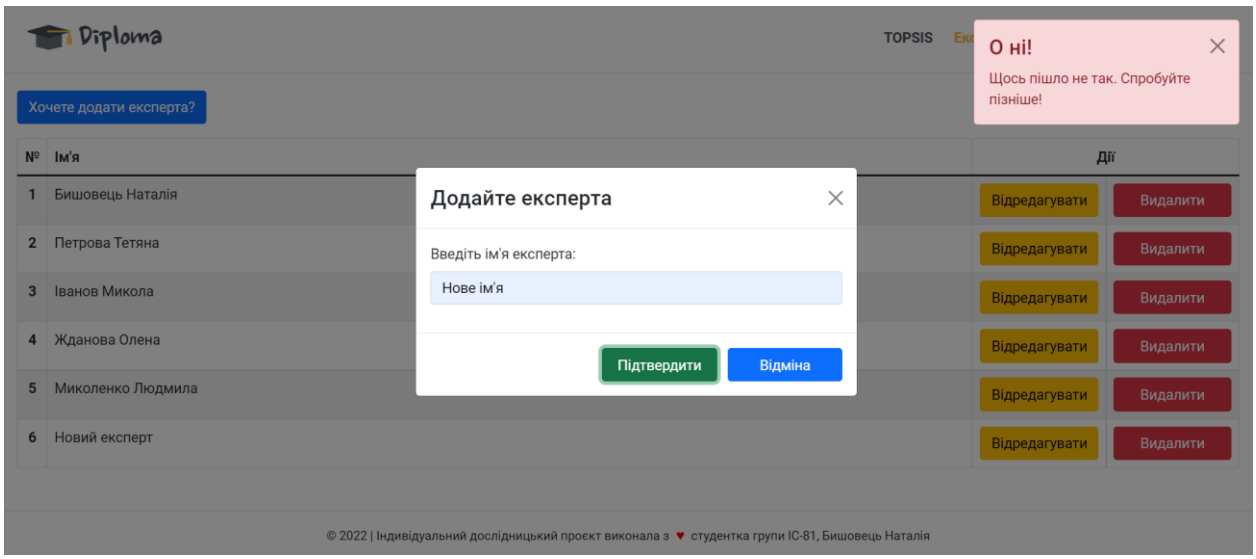

Рисунок 5.6 – Сповіщення про проблему на стороні серверу

Користувач має змогу видалити експерта з бази даних. При кліку на кнопку «Видалити» з’являється модальне вікно, яке просить користувача підтвердити його намір видалення (рис. 5.7).

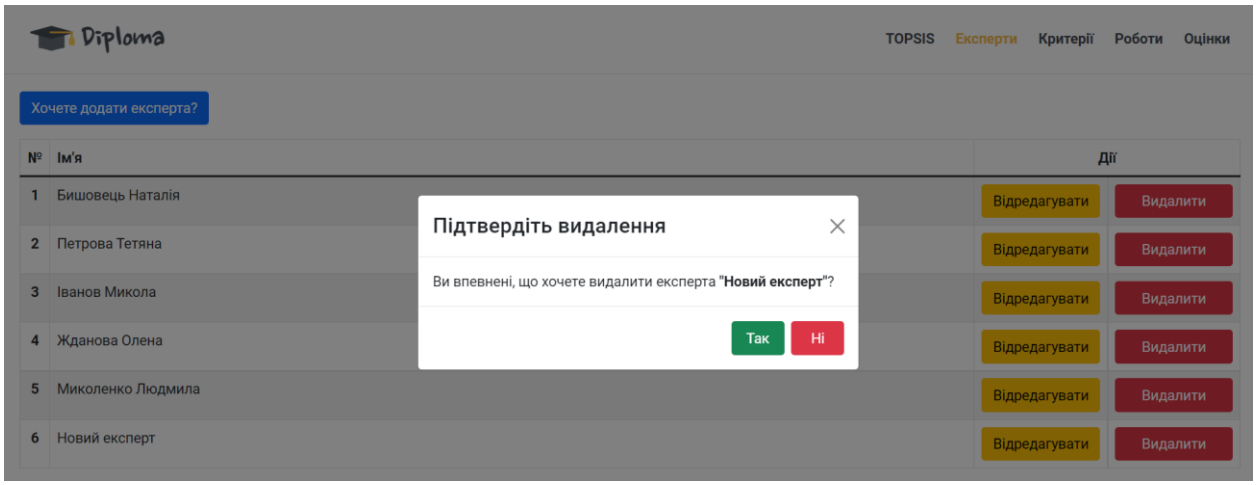

Рисунок 5.7 – Модальне вікно, що просить підтвердити намір користувача видалити експерта з бази даних

При кліку на «Так» у модальному вікні, з’являється сповіщення про успішне видалення експерта з бази даних (рис. 5.8), а також оновлюється список усіх експертів, і «Нового експерта» у загальному списку вже немає (рис. 5.9).

Рисунок 5.8 – Сповіщення про успішне видалення експерта з бази даних

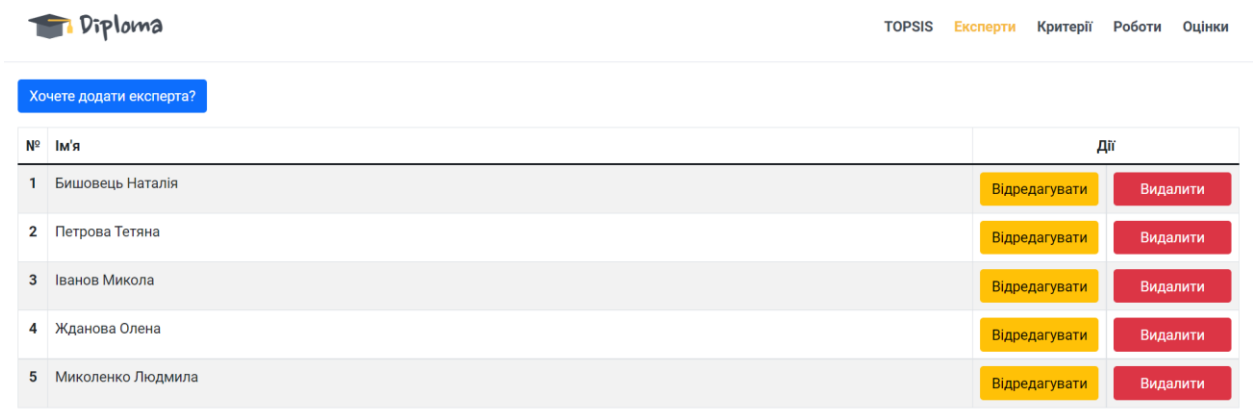

Рисунок 5.9 – Оновлений список усіх експертів

Користувач має змогу відредагувати ім’я експерта. Для цього потрібно натиснути на кнопку «Відредагувати» і замість імені з’являється поле для редагування (рис 5.10), в якому користувач може ввести нове ім’я або залишити старе, якщо передумав щось змінювати (натиснувши кнопку «Відміна»).

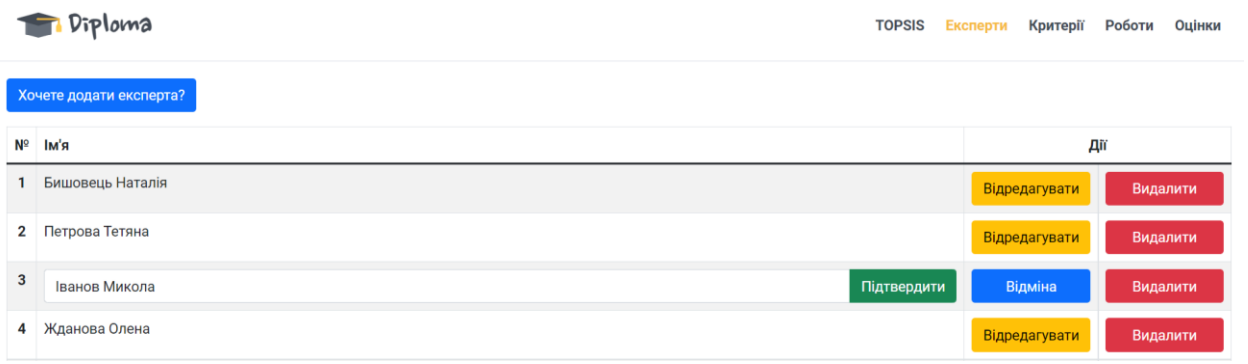

Рисунок 5.10 – Редагування імені експерта

Після введення нового імені (нехай буде «Іванов Іван») користувач побачить сповіщення про успішне оновлення експерта у базі даних (рис. 5.11), а також побачить оновлений список усіх експертів уже з оновленим ім’ям. На рисунку 5.12 можна побачити, що замість «Іванов Микола» тепер відображається «Іванов Іван».

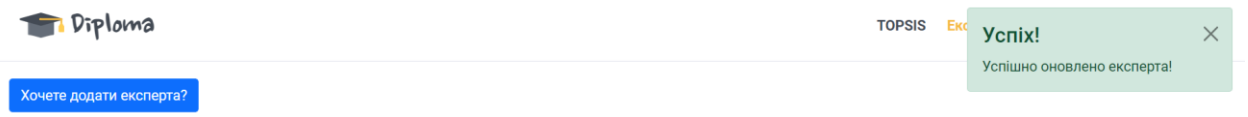

Рисунок 5.11 – Сповіщення про успішне оновлення експерта у базі даних

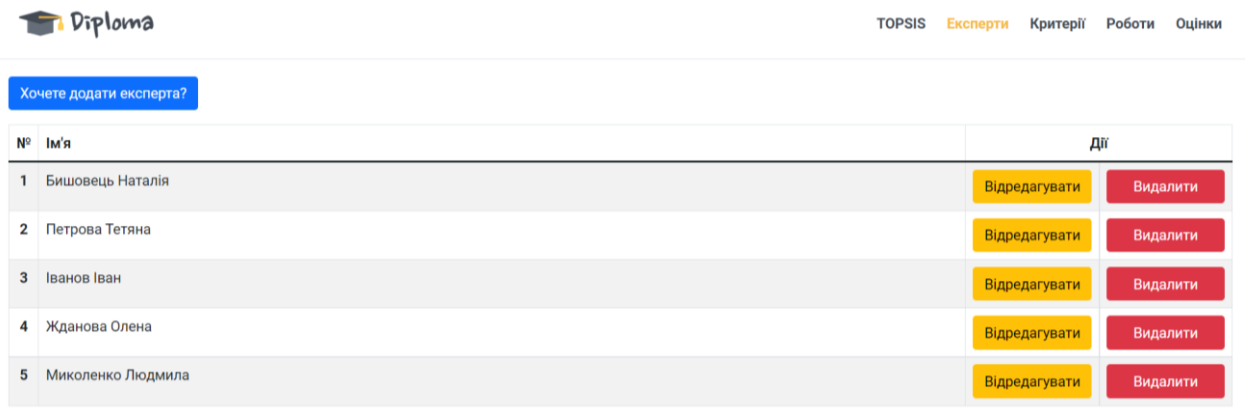

Рисунок 5.12 – Оновлений список усіх експертів з відредагованим іменем експерта

2. Опис функціоналу сторінки «Роботи»

На рисунку 5.13 можна побачити базове відображення інформації про усі роботи з бази даних: їхня назва та порядковий номер у списку. Для кожної роботи є можливість зміни назви, а також видалення її з бази даних, створення чи додавання. Кожна функціональність буде описана окремо.

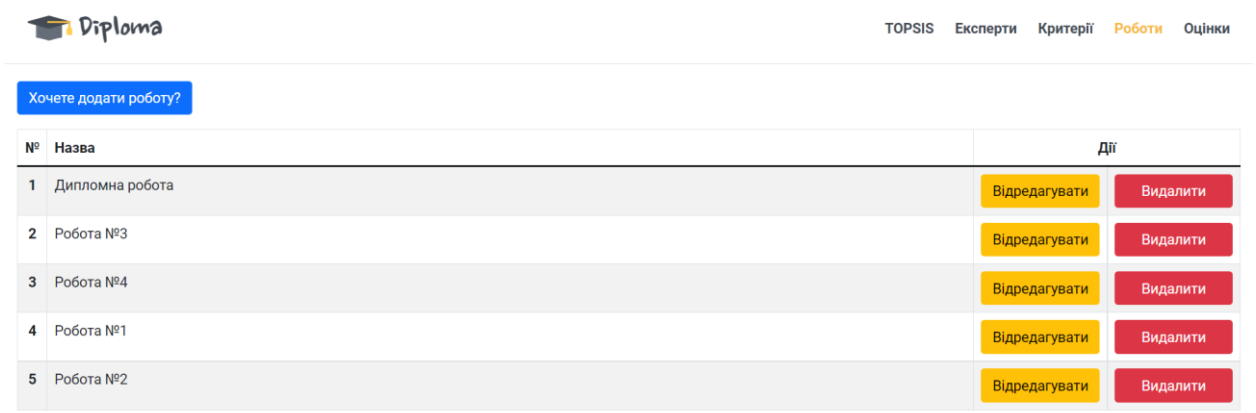

Рисунок 5.13 – Сторінка «Роботи»

При кліку на кнопку «Хочете додати роботу?» з’являється модальне вікно з формою для створення роботи, в якому користувач може ввести назву нової роботи та підтвердити її додавання до бази даних (рис. 5.14).

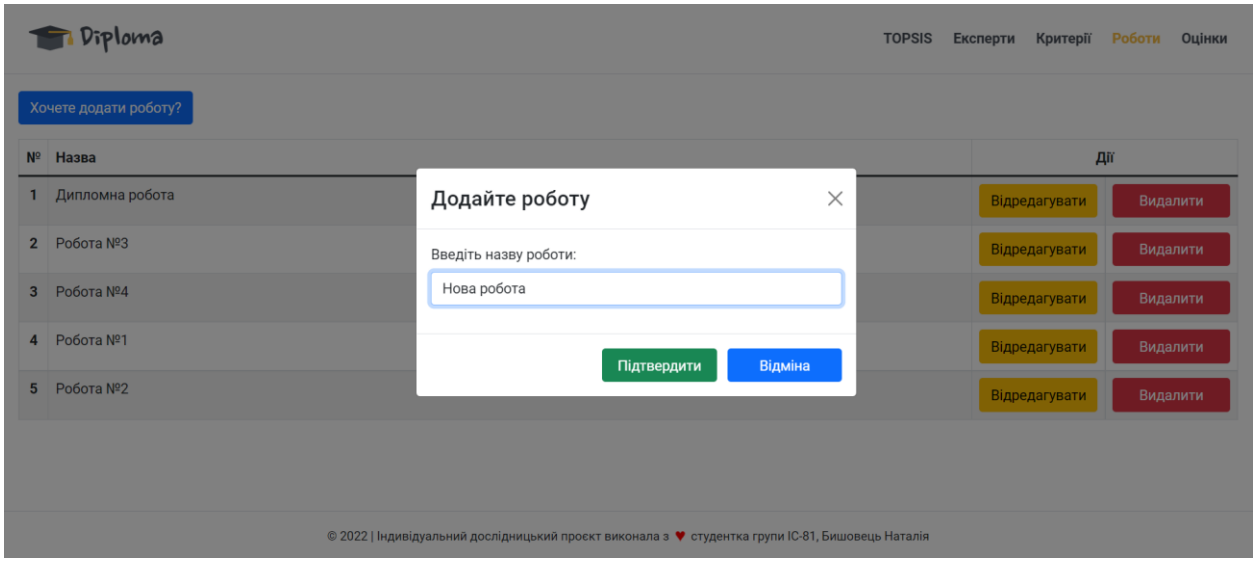

Рисунок 5.14 – Модальне вікно для додавання роботи до бази даних

Після введення назви (нехай буде «Нова робота») та кліку на кнопку «Підтвердити», з’являється сповіщення про успішне додавання роботи до бази

даних (рис. 5.15). Також список усіх робіт оновлюється і можна побачити новостворену роботу останньою у списку (рис. 5.16).

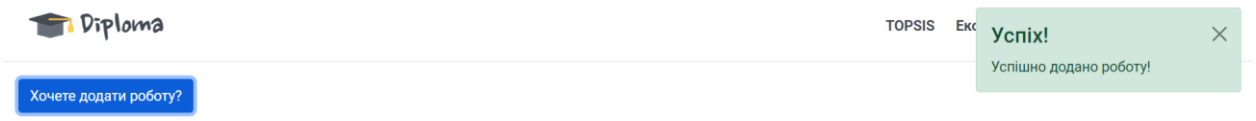

Рисунок 5.15 – Сповіщення про успішне додавання роботи до бази даних

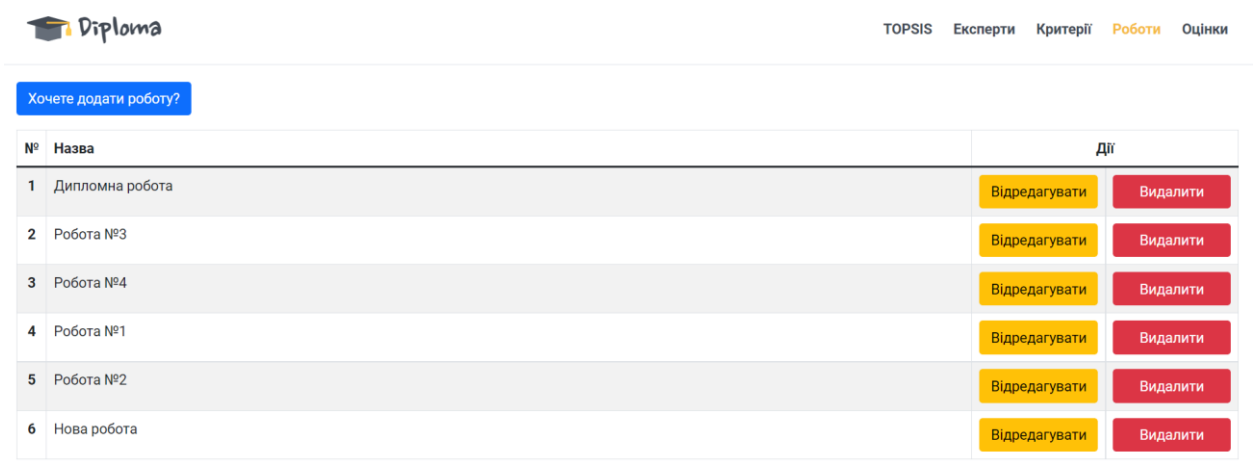

Рисунок 5.16 – Оновлений список усіх робіт

У випадку, якщо на стороні сервера виникла помилка, та запит не обробився, з'являється сповіщення, яке говорить про те, що щось пішло не так і користувачу краще повторити спробу пізніше (рис. 5.17).

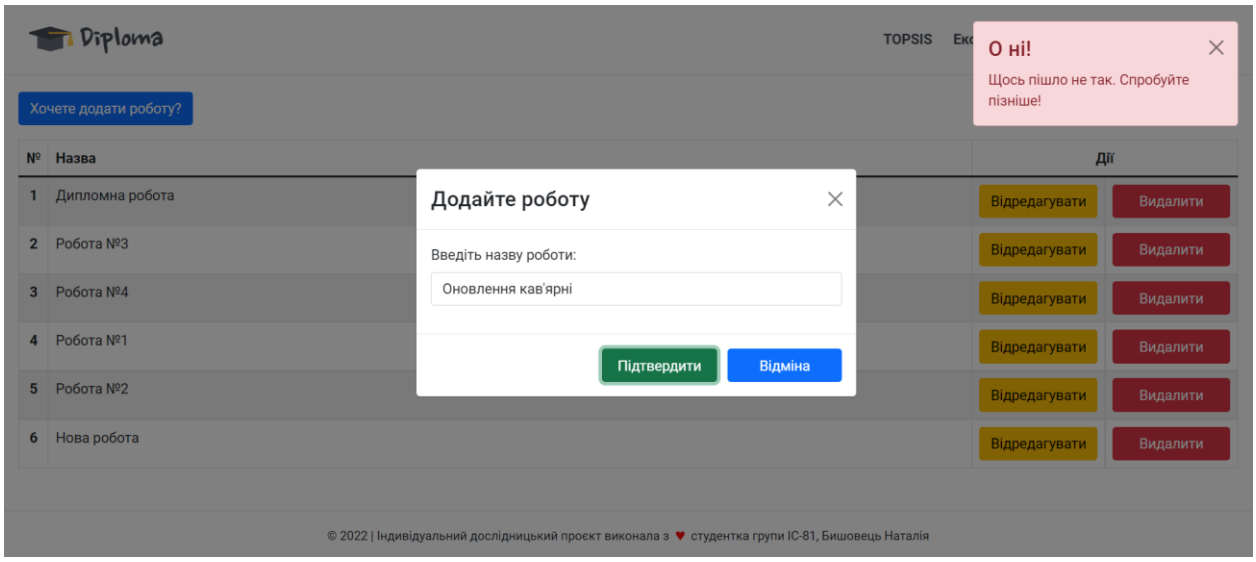

Рисунок 5.17 – Сповіщення про проблему на стороні серверу

Користувач має змогу видалити роботу з бази даних. При кліку на кнопку «Видалити» з’являється модальне вікно, яке просить користувача підтвердити його намір видалення (рис. 5.18).

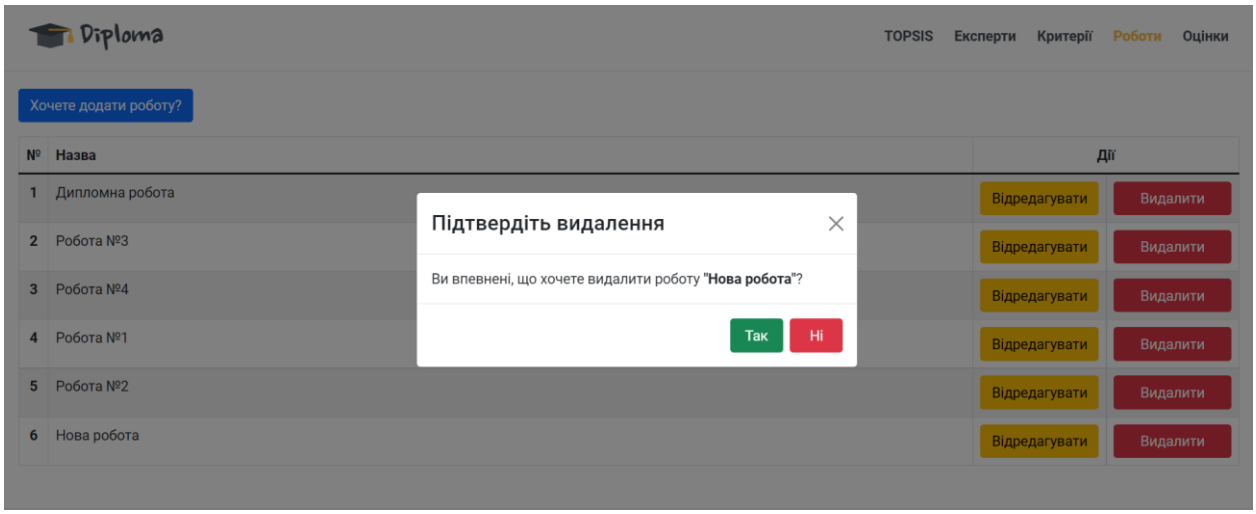

Рисунок 5.18 – Модальне вікно, що просить підтвердити намір користувача видалити експерта з бази даних

При кліку на «Так» у модальному вікні, з’являється сповіщення про успішне видалення роботи з бази даних (рис. 5.19), а також оновлюється список усіх робіт, і «Нова робота» у загальному списку більше не відображається (рис. 5.20).

Рисунок 5.19 – Сповіщення про успішне видалення роботи з бази даних

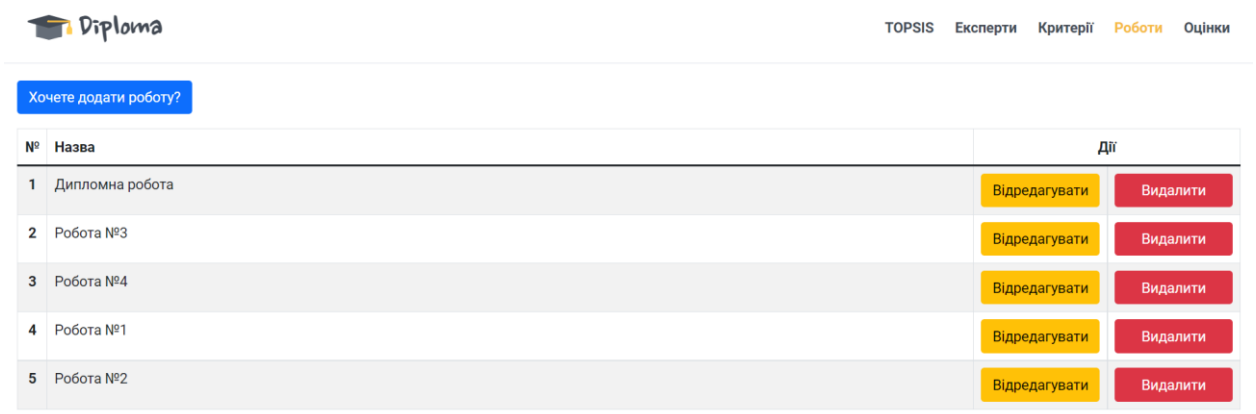

Рисунок 5.20 – Оновлений список усіх робіт

Користувач має змогу відредагувати назву роботи. Для цього потрібно натиснути на кнопку «Відредагувати» і замість назви з’являється поле для редагування (рис 5.21), в якому користувач може ввести нову назву або залишити стару, якщо передумав щось змінювати (натиснувши кнопку «Відміна»).

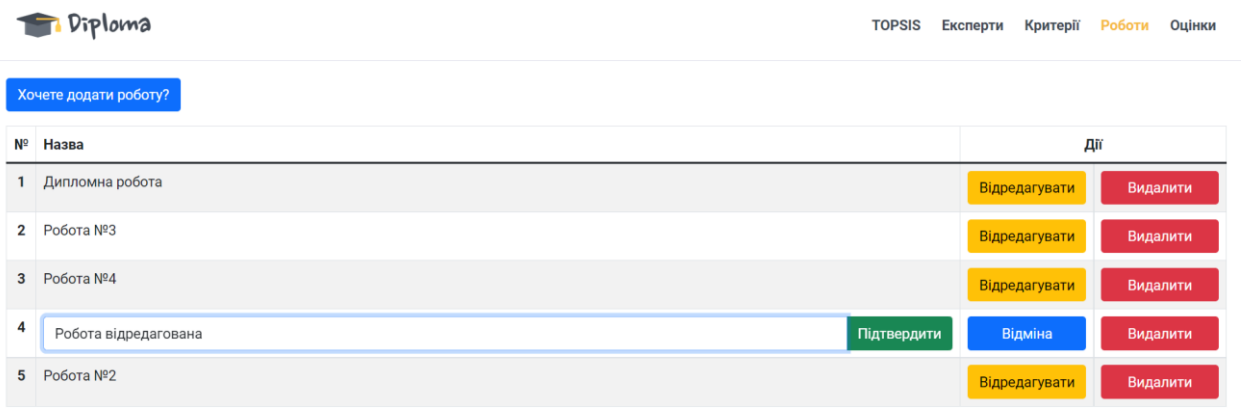

Рисунок 5.21 – Редагування назви роботи

Після введення нової назви (нехай буде «Робота відредагована») користувач побачить сповіщення про успішне оновлення роботи у базі даних (рис. 5.22), а також побачить оновлений список усіх робіт уже з оновленою назвою. На рисунку 5.23 можна побачити, що замість «Робота №1» відображається «Робота відредагована».

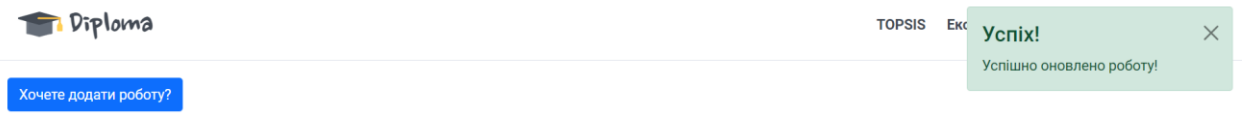

Рисунок 5.22 – Сповіщення про успішне оновлення роботи у базі даних

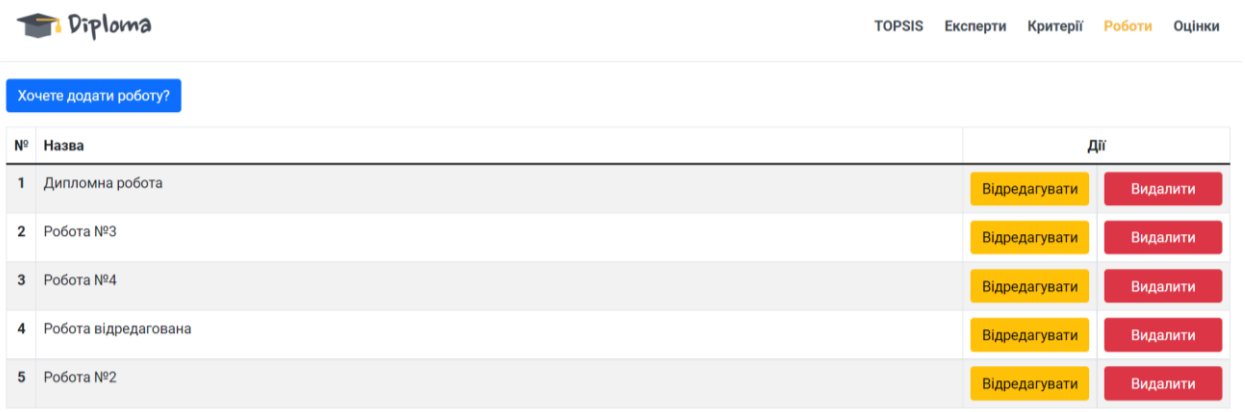

Рисунок 5.23 – Оновлений список усіх робіт з відредагованою назвою роботи

3. Опис функціоналу сторінки «Критерії»

На рисунку 5.24 можна побачити базове відображення інформації про усі критерії з бази даних: порядковий номер у списку, назва та тип (об’єктивний чи суб’єктивний). Для кожного критерію є можливість зміни її назви та типу, а також видалення з бази даних, створення чи додаванна. Кожна функціональність буде описана окремо.

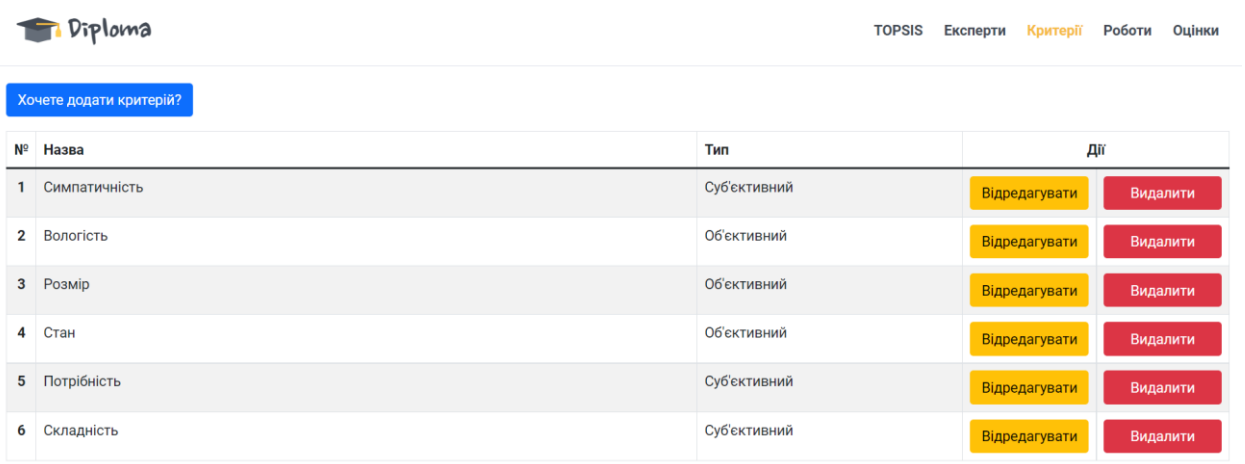

Рисунок 5.24 – Сторінка «Критерії»

При кліку на кнопку «Хочете додати критерій?» з’являється модальне вікно з формою для створення критерію, в якому користувач може ввести назву нового критерію, його тип та підтвердити його додавання до бази даних (рис. 5.25). Тип кожного нового критерію за замовчуванням – об’єктивний.

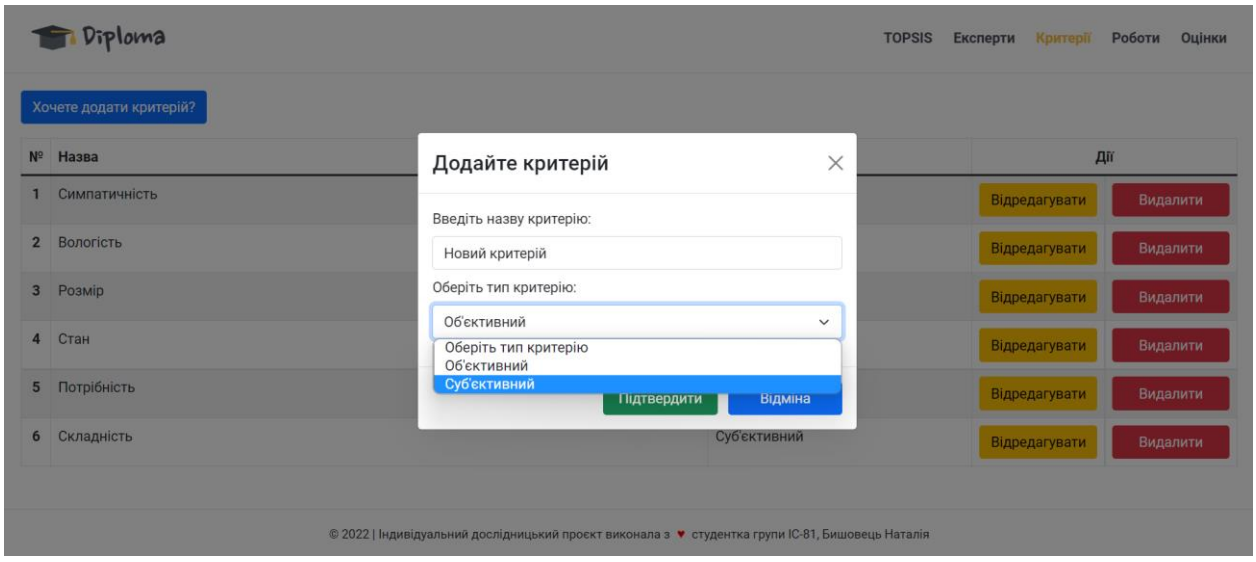

Рисунок 5.25 – Модальне вікно для додавання критерію до бази даних

Після введення назви (нехай буде «Новий критерій») та після вибору типу критерію (нехай буде «Суб’єктивний»), користувач натискає на кнопку «Підтвердити», після чого з’являється сповіщення про успішне додавання критерію до бази даних (рис. 5.26). Також список усіх критеріїв оновлюється і можна побачити новостворений критерій останнім у списку (рис. 5.27).

Рисунок 5.26 – Сповіщення про успішне додавання критерію до бази даних

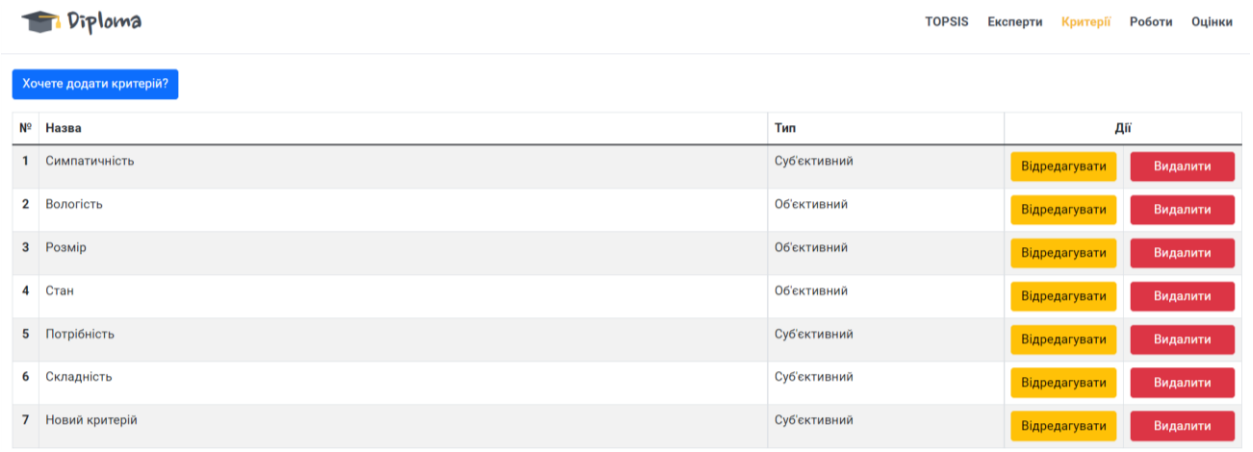

Рисунок 5.27 – Оновлений список усіх критеріїв

У випадку, якщо на стороні сервера виникла помилка, та запит не обробився, з’являється сповіщення, яке говорить про те, що щось пішло не так і користувачу краще повторити спробу пізніше (рис. 5.28).

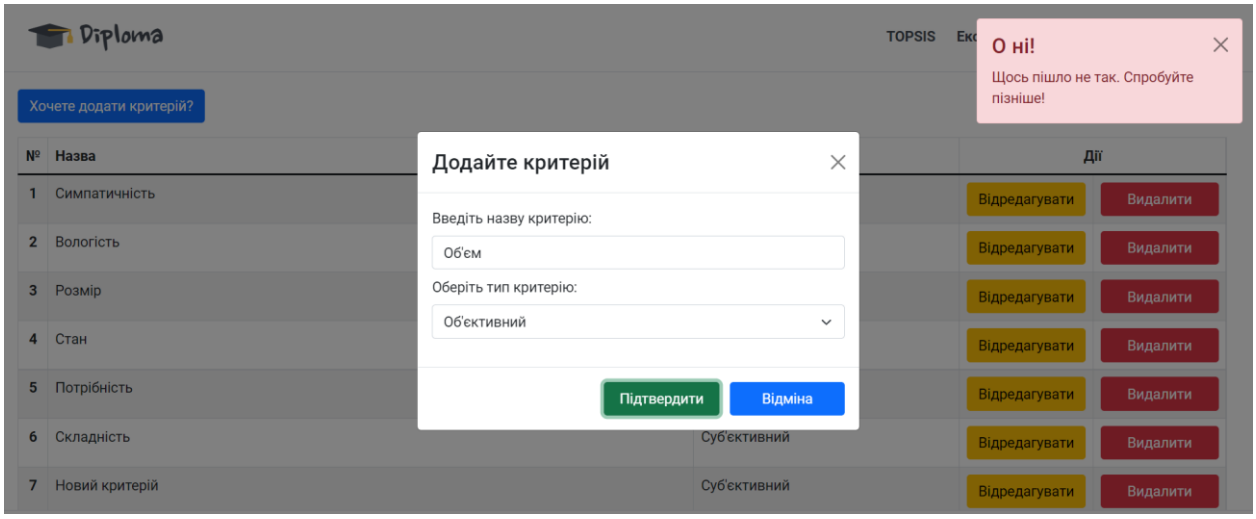

Рисунок 5.28 – Сповіщення про проблему на стороні серверу

Користувач має змогу видалити критерій з бази даних. При кліку на кнопку «Видалити» з’являється модальне вікно, яке просить користувача підтвердити його намір видалення (рис. 5.29).

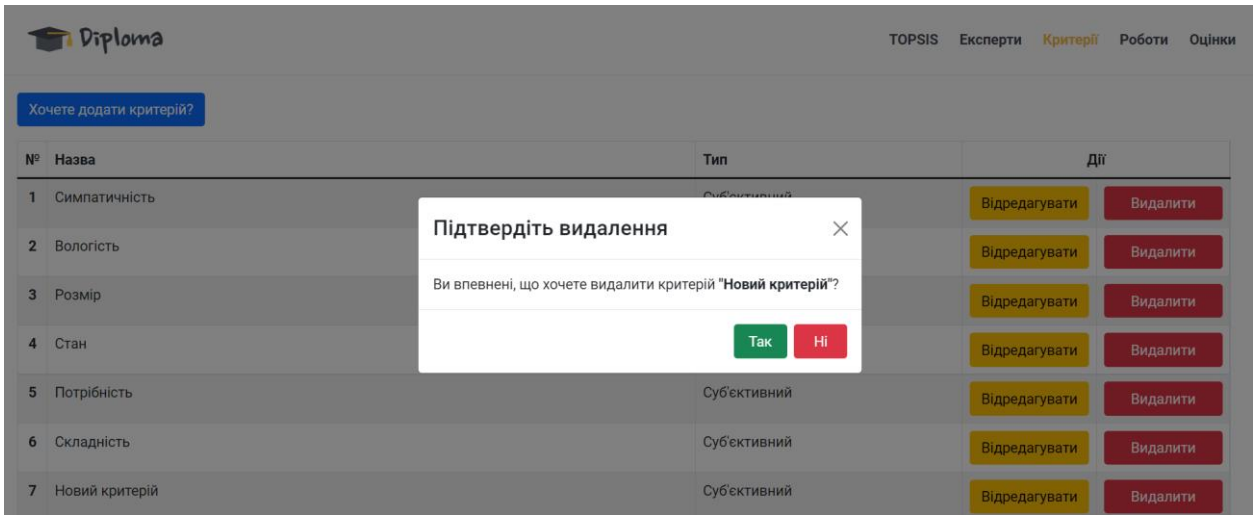

Рисунок 5.29 – Модальне вікно, що просить підтвердити намір користувача видалити критерій з бази даних

При кліку на «Так» у модальному вікні, з’являється сповіщення про успішне видалення критерію з бази даних (рис. 5.30), а також оновлюється список усіх критеріїв, і «Нового критерію» у загальному списку вже немає (рис. 5.31).

Рисунок 5.30 – Сповіщення про успішне видалення критерію з бази даних

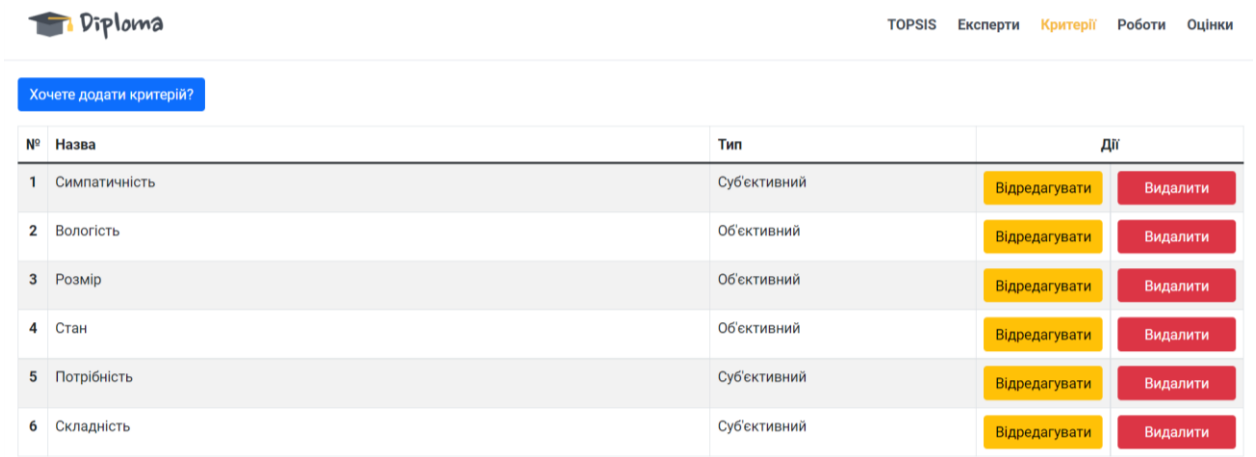

Рисунок 5.31 – Оновлений список усіх критеріїв

Користувач має змогу відредагувати назву та тип критерію. Для цього потрібно натиснути на кнопку «Відредагувати» і замість назви з’явиться поле для редагування, а замість типу – поле для вибору типу (рис 5.32), в якому користувач може ввести нову назву та обрати новий тип або залишити все, як є, якщо передумав щось змінювати (натиснувши кнопку «Відміна»).

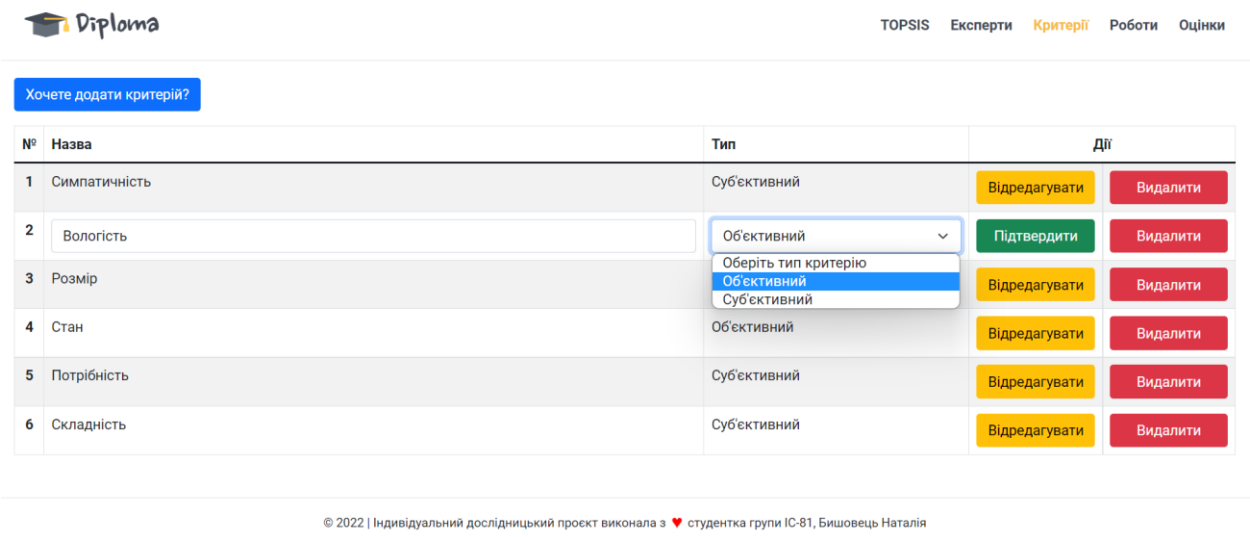

Рисунок 5.32 – Редагування назви та типу критерію

Після введення нової назви (нехай буде «Відредагований критерій») та нового типу (нехай буде «Суб’єктивний») користувач побачить сповіщення про успішне оновлення критерію у базі даних (рис. 5.33), а також побачить оновлений список усіх критеріїв уже з оновленою назвою та типом. На рисунку 5.34 можна побачити, що замість «Вологість» відображається «Відредагований критерій».

Рисунок 5.33 – Сповіщення про успішне оновлення критерію у базі даних

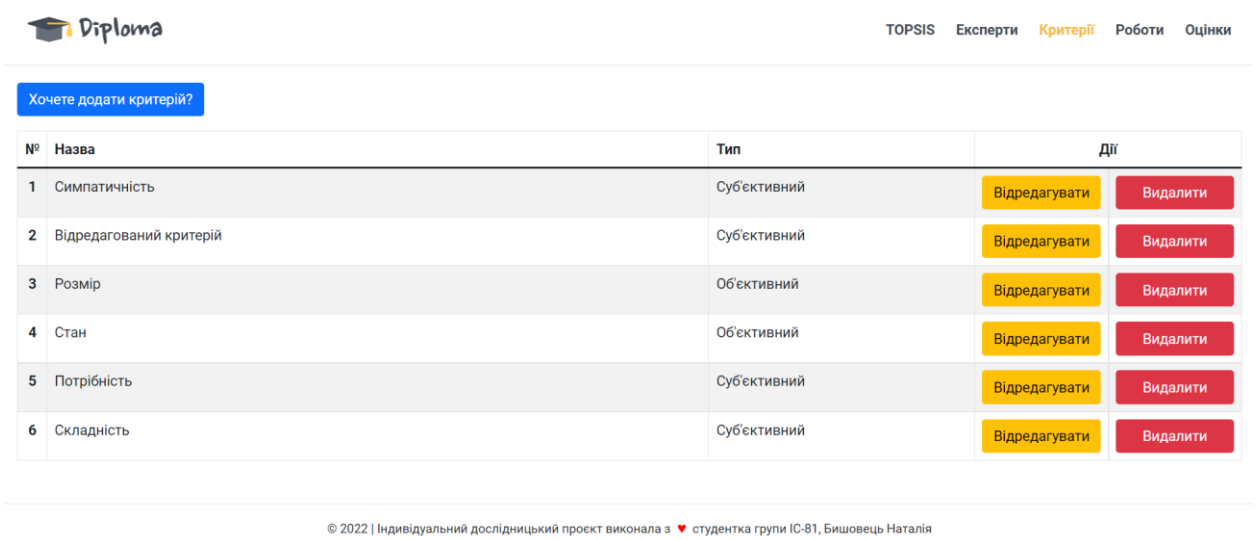

Рисунок 5.34 – Оновлений список усіх критеріїв з відредагованою назвою та типом критерію

4. Опис функціоналу сторінки «Оцінки»

На рисунку 5.35 можна побачити відображення інформації, яка надійшла з бази даних, за двома типами критеріїв: об’єктивним і суб’єктивним.

За першим типом критеріїв відображається лише назва роботи та оцінка кожного з критеріїв, якими ця робота характеризується.

За другим типом необхідно вказати більше інформації. По кожній з робіт необхідно вивести усі суб’єктивні критерії, якими ця робота характеризується та всі оцінки експертів, які оцінюють кожен з критеріїв. В подальшому ця інформація буде використана для знаходження кількісних оцінок за кожним з критеріїв методом узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів.

З усіх оцінок (об’єктивних та суб’єктивних, переведених у кількісну форму) буде сформована матриця з вхідними даними для методу визначення першочерговості виконання робіт методом TOPSIS.

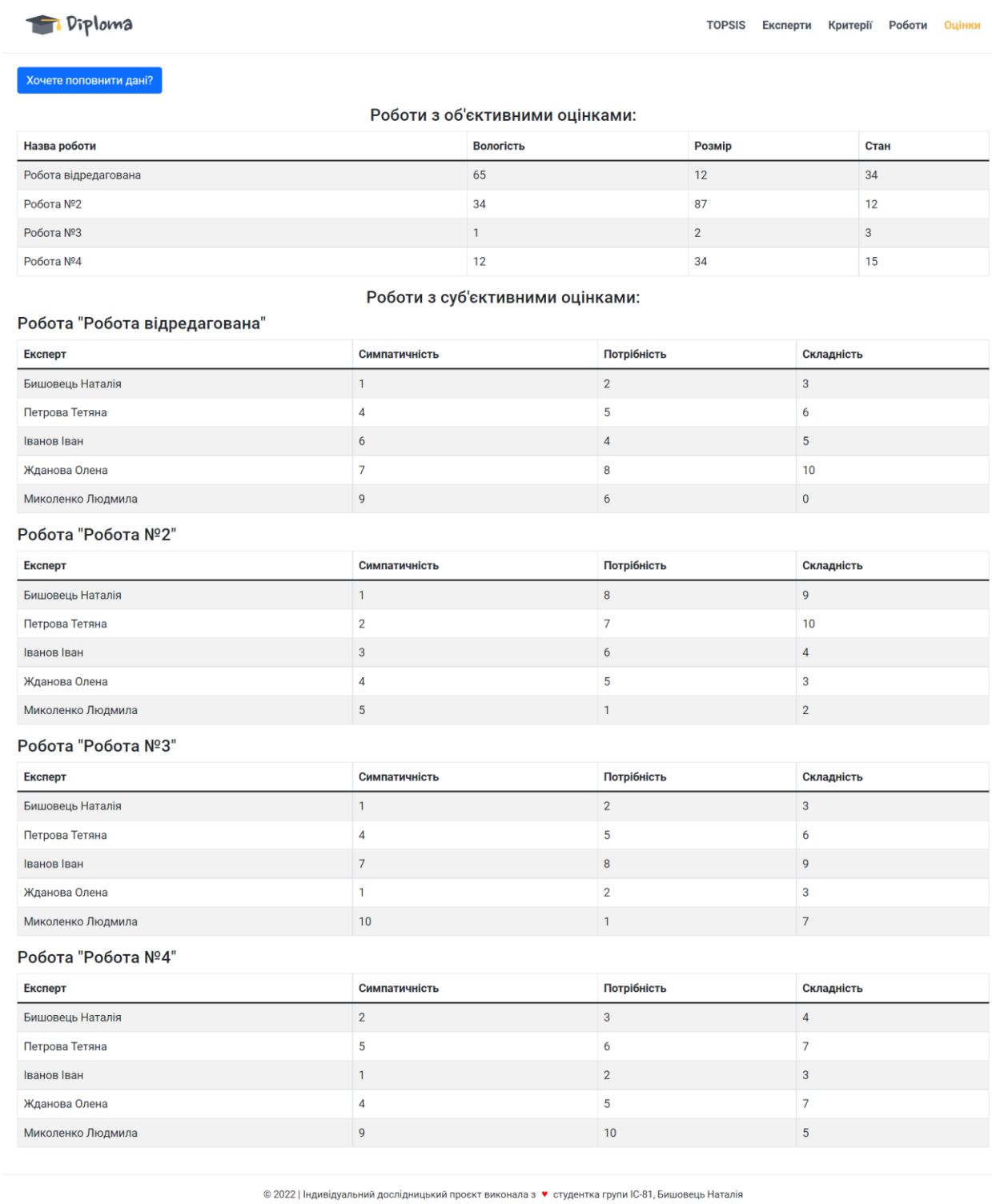

Рисунок 5.35 – Сторінка «Оцінки»

У користувача є можливість додавати оцінки за кожним з критеріїв для будь-якої з робіт. Для цього необхідно натиснути на кнопку «Хочете поповнити дані?» (рисунок 5.36). З’явиться модальне вікно, в якому необхідно обрати бажану роботу та критерії, за якими він хоче її оцінити.

Рисунок 5.36 – Модальне вікно вибору роботи та критеріїв, за якими вона оцінюється

Після натискання на кнопку «Підтвердити» з’явиться дві таблиці для введення інформації про оцінки об’єктивних та суб’єктивних критеріїв, а також сповіщення про те, що необхідно заповнити усі поля (рис. 5.37).

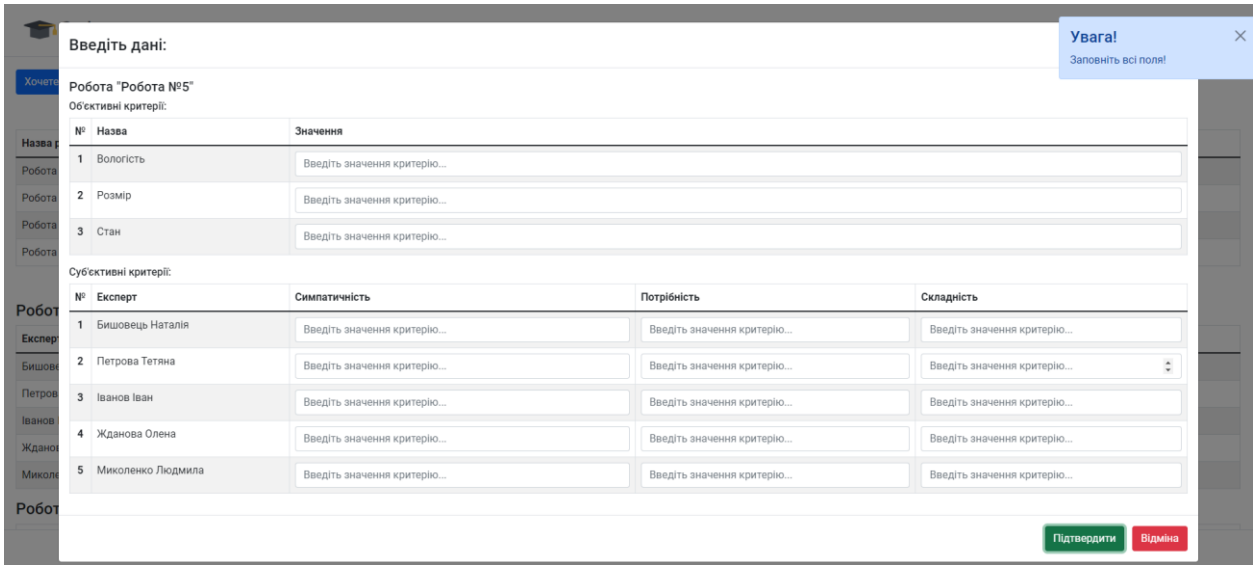

Рисунок 5.37 – Модальне вікно для заповнення інформації про оцінки об’єктивних та суб’єктивних критеріїв для обраної роботи

Після заповнення всіх даних натискаємо на кнопку «Підтвердити» для збереження інформації та для додавання всіх оцінок до бази даних. З’явиться сповіщення про успішне додавання до бази даних оцінок (рис. 5.38), а також оновиться список усіх оцінок на сторінці (рис. 5.39).

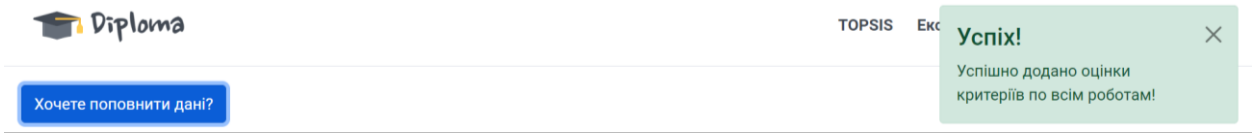

Рисунок 5.38 – Сповіщення про успішне додавання усіх оцінок за всіма критеріям до бази даних

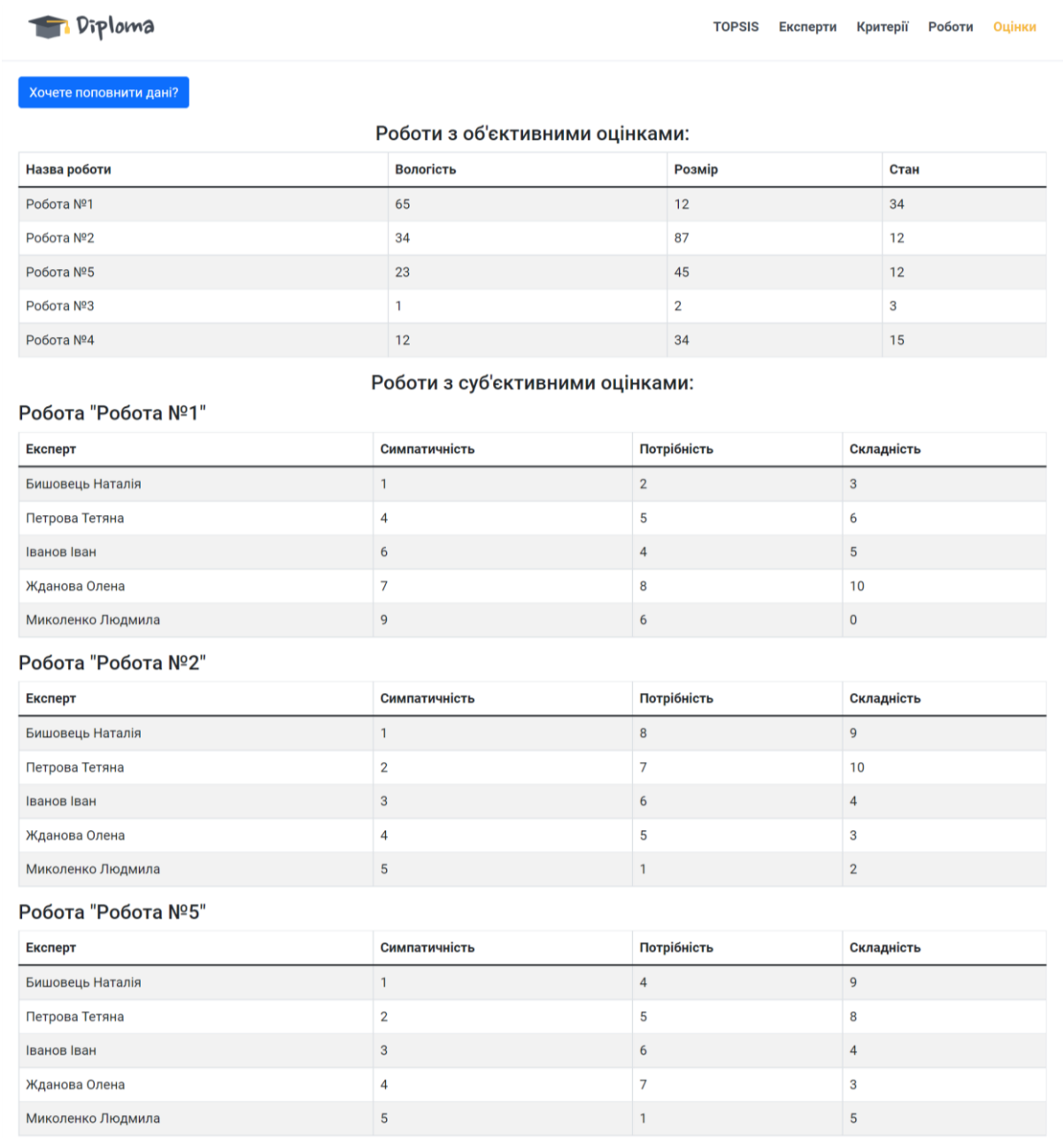

Рисунок 5.39 – Оновлена сторінка з новою роботою

5. Опис функціоналу сторінки «TOPSIS»

На рисунку 5.40 можна побачити відображення інформації, яка надійшла з бази даних, за двома типами критеріїв: об’єктивним і суб’єктивним. При чому, інформація про суб’єктивні критерії надійшла уже в кількісному вигляді після виконання методу узагальненого оцінювання важливості критеріїв з паралельним уточненням компетентності експертів.

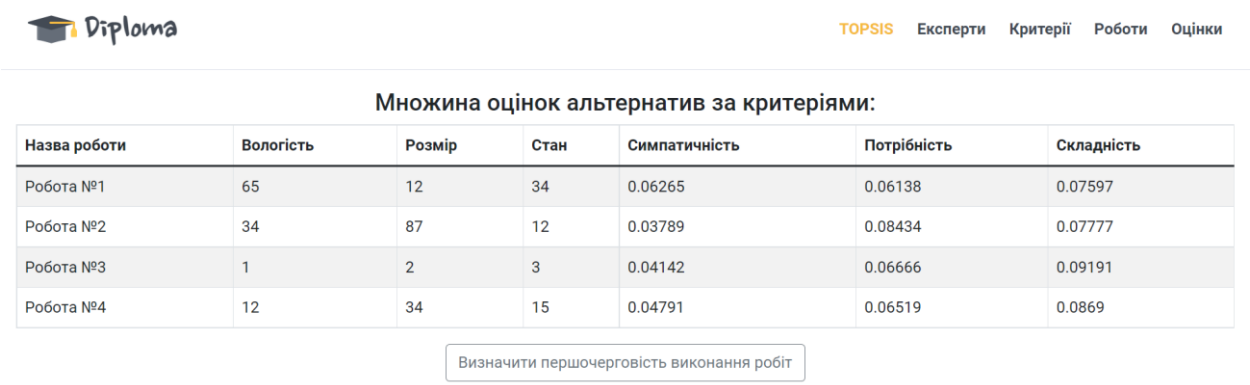

Рисунок 5.40 – Сторінка «TOPSIS»

У користувача є можливість визначити, в якому порядку йому краще реалізовувати вказані роботи. Також він може запустити виконання алгоритму TOPSIS для визначення першочерговості виконання робіт. Для цього необхідно натиснути кнопку «Визначити першочерговість виконання робіт». Після натискання з’явиться форма для введення вагових коефіцієнтів для кожного з критерію, який позначає їхню важливість (за замовчуванням – 1), а також вибору типу критерію (на мінімум чи на максимум) (рис. 5.41).

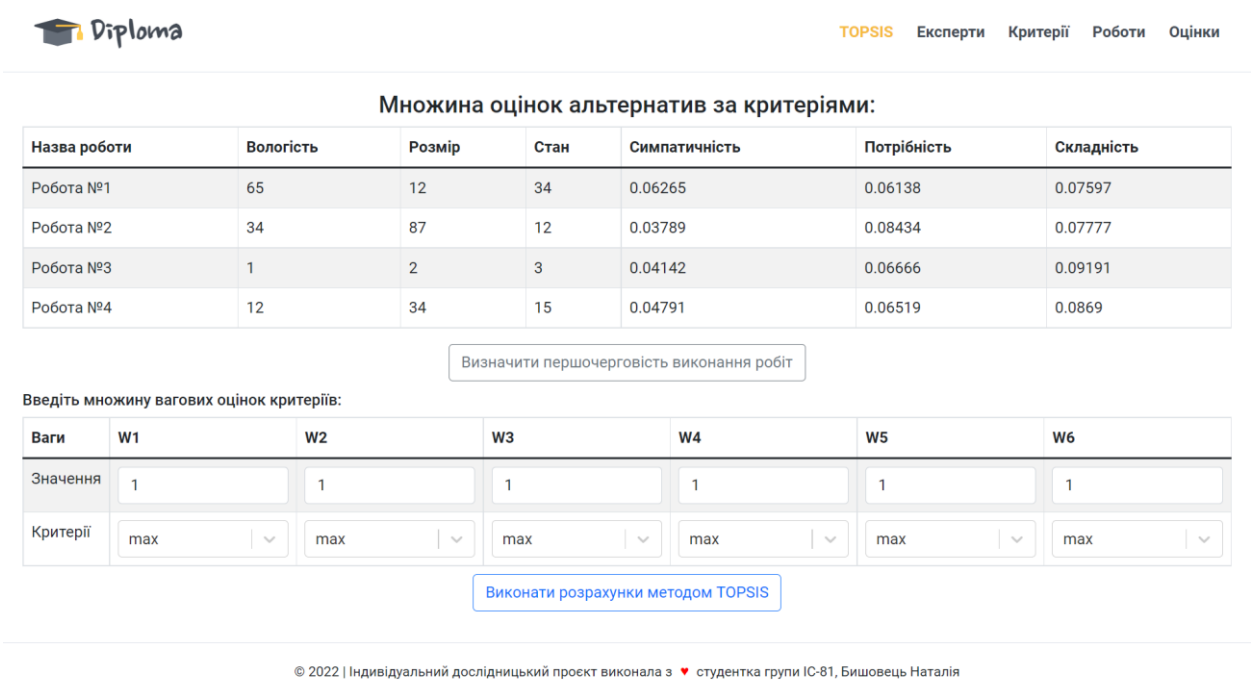

Рисунок 5.41 – Форма для вибору типу критерію (на максимум чи на мінімум), а також введення вагових коефіцієнтів, що позначають важливість кожного з них

Після введення всіх необхідних даних необхідно натиснути на кнопку «Виконати розрахунки методом TOPSIS» та отримати результат: порядок виконання робіт та ваги (важливість) (рис. 5.42).

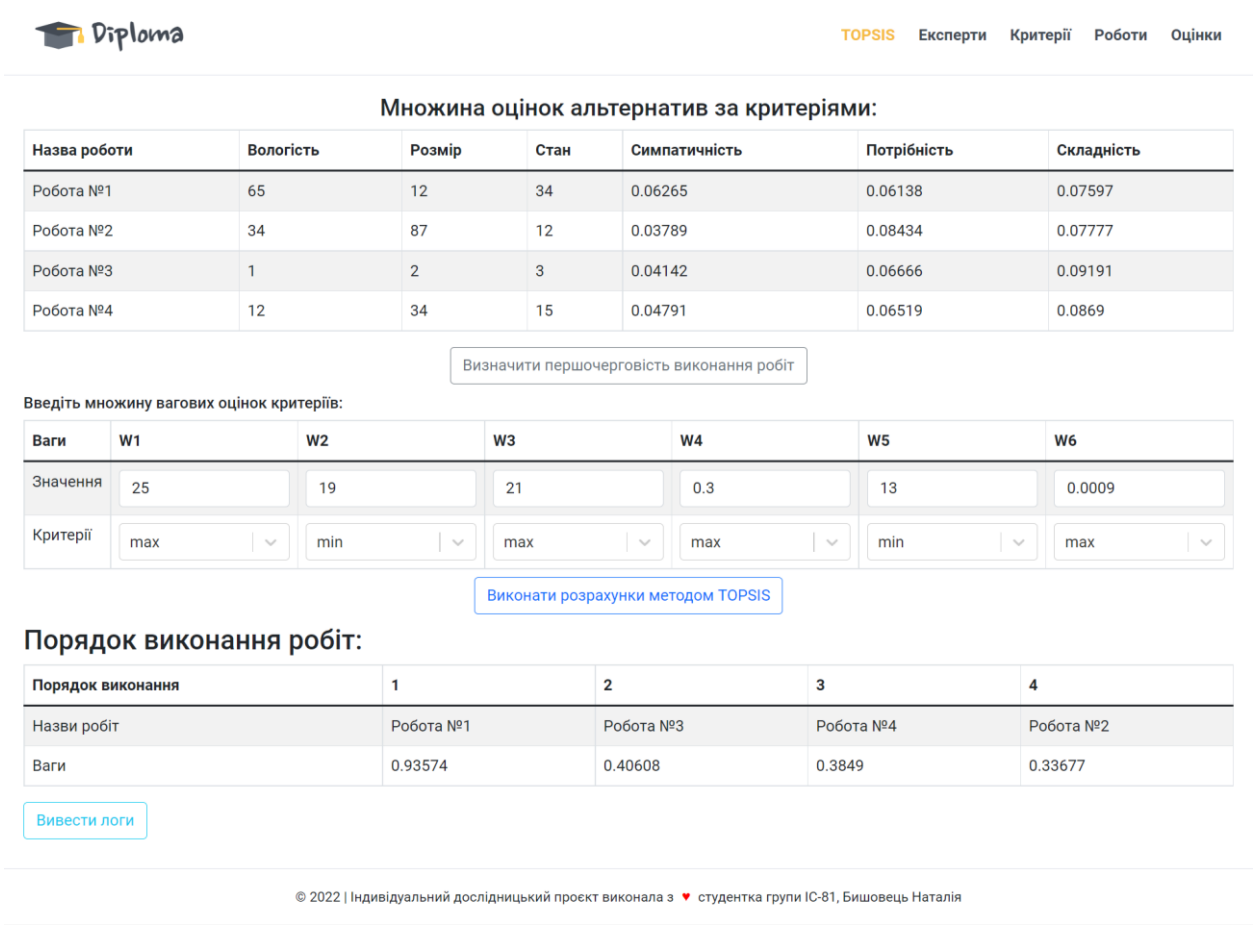

Рисунок 5.42 – Результат виконання алгоритму «TOPSIS»

Для дослідника є можливість побачити процес виконання алгоритму, якщо він натисне кнопку «Вивести логи». Тоді він побачить всі етапи алгоритму: матрицю нормалізованих оцінок альтернатив, матрицю зважених нормалізованих оцінок альтернатив, віддаленість альтернатив від позитивної ідеальної точки (PIS), віддаленість альтернатив від негативної ідеальної точки (NIS), а також наближеність альтернатив до позитивної ідеальної точки (PIS), що і є кінцевим результатом, за яким формується відповідь (рис. 5.43).

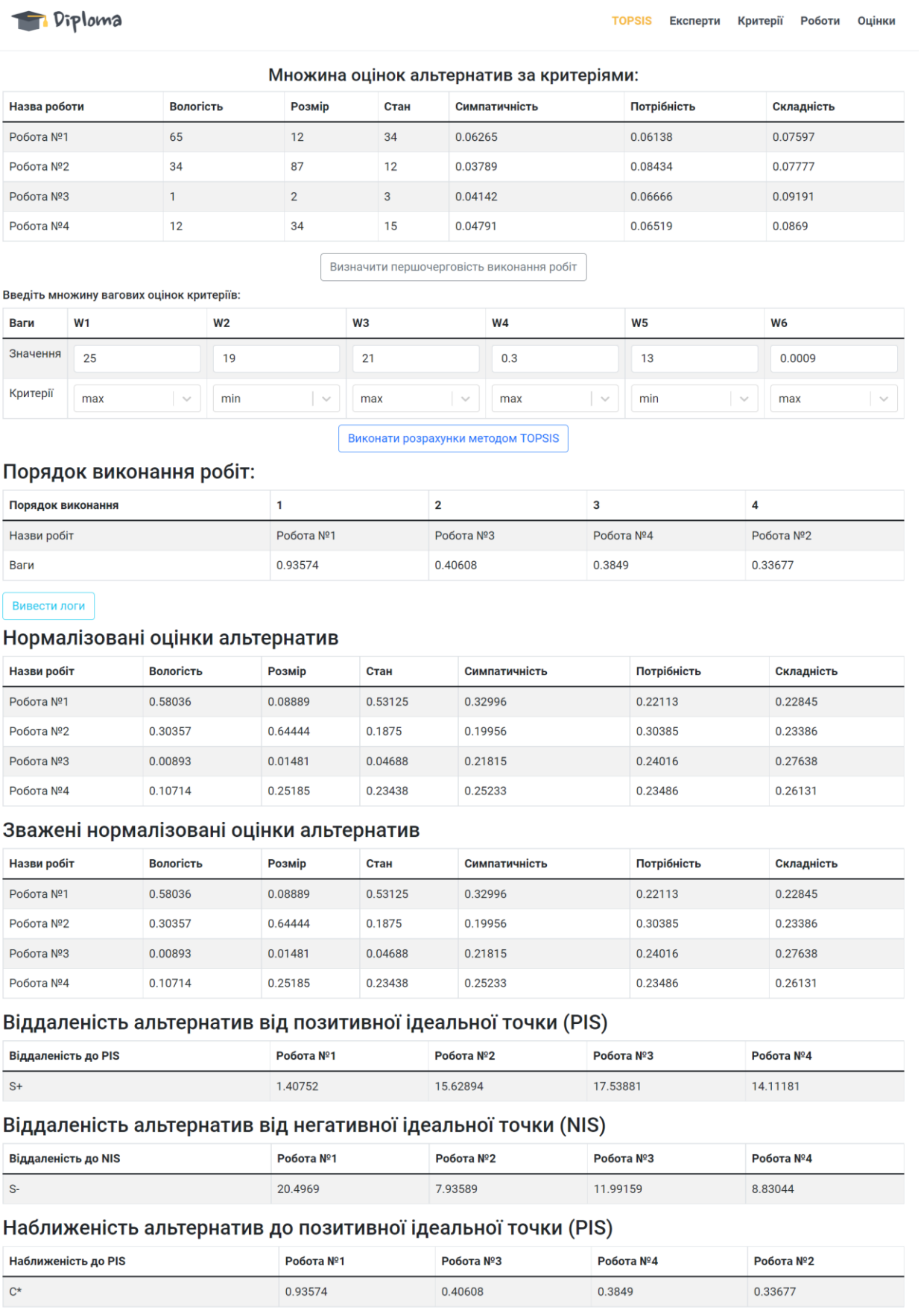

Рисунок 5.43 – Логи виконання алгоритму «TOPSIS»

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Перевірка відповідності функцій інформаційної системи підтримки визначення першочерговості виконання робіт вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603-92. Інформаційна технологія. Види випробувань автоматизованих систем;

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В процесі тестування була перевірена функціональність системи. Для перевірки працездатності розробленої інформаційної системи застосовувалось функціональне тестування. У наступних таблицях наведено перелік випробувань основних функціональних можливостей ПЗ (табл. 5.1-5.13).

Таблиця 5.1 – Створення експерта

Мета тесту	Перевірка функції «Створення експерта».
Початковий стан системи	Відкрита сторінка «Експерти».
Вхідні дані	Ім’я експерта.
Схема проведення тесту	Натиснути кнопку «Хочете додати експерта?» та ввести ім’я до форми створення експерта у модальному вікні, що з’явилось після натискання кнопки.
Очікуваний результат	Поява сповіщення про успішне додавання експерта з таким іменем до бази даних, перезавантаження сторінки та відображення нового експерта на сторінці.

Стан системи після проведення випробування	Сповіщення про успішне додавання експерта з’явилось, сторінка перезавантажилась, новий експерт з’явився у загальному списку.

Таблиця 5.2 – Редагування експерта

Мета тесту	Перевірка функції «Редагування експерта».
Початковий стан системи	Відкрита сторінка «Експерти».
Вхідні дані	Ім’я експерта.
Схема проведення тесту	Натиснути кнопку «Відредагувати» та ввести ім’я до форми редагування експерта, що з’явилась після натискання кнопки. Підтвердити зміни.
Очікуваний результат	Поява сповіщення про успішне оновлення імені експерта у базі даних, перезавантаження сторінки та відображення оновленого експерта замість старого на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне оновлення експерта з’явилось, сторінка перезавантажилась, оновлений експерт з’явився у загальному списку замість старого.

Таблиця 5.3 – Видалення експерта

Мета тесту	Перевірка функції «Видалення експерта».
Початковий стан системи	Відкрита сторінка «Експерти».
Вхідні дані	Унікальний ідентифікатор експерта (неявні вхідні дані, передаються при натисканні на кнопку «Видалити»).

Схема проведення тесту	Натиснути кнопку «Видалити» та підтвердити намір видалення, натиснувши кнопку «Так» у модальному вікні, що з’явилось, після натискання кнопки «Видалити».
Очікуваний результат	Поява сповіщення про успішне видалення експерта з бази даних, перезавантаження сторінки та відображення оновленого списку експертів на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне видалення експерта з’явилось, сторінка перезавантажилась, видалений експерт зник із загального списку експертів.

Таблиця 5.4 – Створення роботи

Мета тесту	Перевірка функції «Створення роботи».
Початковий стан системи	Відкрита сторінка «Роботи».
Вхідні дані	Назва роботи.
Схема проведення тесту	Натиснути кнопку «Хочете додати роботу?» та ввести назву до форми створення роботи у модальному вікні, що з’явилось після натискання кнопки.
Очікуваний результат	Поява сповіщення про успішне додавання роботи з такою назвою до бази даних, перезавантаження сторінки та відображення нової роботи на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне додавання роботи з’явилось, сторінка перезавантажилась, нова робота з’явилась у загальному списку.

Таблиця 5.5 – Редагування роботи

Мета тесту	Перевірка функції «Редагування роботи».
Початковий стан системи	Відкрита сторінка «Роботи».
Вхідні дані	Назва роботи.
Схема проведення тесту	Натиснути кнопку «Відредагувати» та ввести назву до форми редагування роботи, що з’явилась після натискання кнопки. Підтвердити зміни.
Очікуваний результат	Поява сповіщення про успішне оновлення назви роботи у базі даних, перезавантаження сторінки та відображення оновленої роботи замість старої на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне оновлення роботи з’явилось, сторінка перезавантажилась, оновлена робота з’явилась у загальному списку замість старого.

Таблиця 5.6 – Видалення роботи

Мета тесту	Перевірка функції «Видалення роботи».
Початковий стан системи	Відкрита сторінка «Роботи».
Вхідні дані	Унікальний ідентифікатор роботи (неявні вхідні дані, передаються при натисканні на кнопку «Видалити»).
Схема проведення тесту	Натиснути кнопку «Видалити» та підтвердити намір видалення, натиснувши кнопку «Так» у модальному вікні, що з’явилось, після натискання кнопки «Видалити».
Очікуваний результат	Поява сповіщення про успішне видалення роботи з бази даних, перезавантаження сторінки та відображення оновленого списку робіт на сторінці.

Стан системи після проведення випробування	Сповіщення про успішне видалення роботи з’явилось, сторінка перезавантажилась, видалена робота зникла із загального списку експертів.

Таблиця 5.7 – Створення критерію

Мета тесту	Перевірка функції «Створення критерію».
Початковий стан системи	Відкрита сторінка «Критерії».
Вхідні дані	Назва та тип критерію.
Схема проведення тесту	Натиснути кнопку «Хочете додати критерій?», ввести назву та обрати тип критерію у форму створення критерію в модальному вікні, що з’явилось після натискання кнопки.
Очікуваний результат	Поява сповіщення про успішне додавання критерію до бази даних, перезавантаження сторінки та відображення нового критерію на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне додавання критерію з’явилось, сторінка перезавантажилась, новий критерій з’явився у загальному списку.

Таблиця 5.8 – Редагування критерію

Мета тесту	Перевірка функції «Редагування критерію».
Початковий стан системи	Відкрита сторінка «Критерії».
Вхідні дані	Назва та тип критерію.
Схема проведення тесту	Натиснути кнопку «Відредагувати», ввести назву та обрати тип критерію у формі редагування критерію, що з’явилась після натискання кнопки. Підтвердити зміни.

Очікуваний результат	Поява сповіщення про успішне оновлення критерію у базі даних, перезавантаження сторінки та відображення оновленого критерію замість старого на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне оновлення критерію з’явилось, сторінка перезавантажилась, оновлений критерій з’явився у загальному списку замість старого.

Таблиця 5.9 – Видалення критерію

Мета тесту	Перевірка функції «Видалення критерію».
Початковий стан системи	Відкрита сторінка «Критерії».
Вхідні дані	Унікальний ідентифікатор критерію (неявні вхідні дані, передаються при натисканні на кнопку «Видалити»).
Схема проведення тесту	Натиснути кнопку «Видалити» та підтвердити намір видалення, натиснувши кнопку «Так» у модальному вікні, що з’явилось, після натискання кнопки «Видалити».
Очікуваний результат	Поява сповіщення про успішне видалення критерію з бази даних, перезавантаження сторінки та відображення оновленого списку критеріїв на сторінці.
Стан системи після проведення випробування	Сповіщення про успішне видалення критерію з’явилось, сторінка перезавантажилась, видалений критерій зник із загального списку критеріїв.

Таблиця 5.10 – Додавання оцінок для об’єктивних критеріїв

Мета тесту	Перевірка функції «Додавання оцінок» для об’єктивних критеріїв.
Початковий стан системи	Відкрите модальне вікно додавання оцінок, що з’явилось після натискання кнопки «Хочете поповнити дані?» на сторінці «Оцінки».
Вхідні дані	Назва роботи та масив об’єктивних критеріїв.
Схема проведення тесту	Обрати назву роботи з випадаючого списку робіт та обрати об’єктивні критерії з випадаючого списку критеріїв.
Очікуваний результат	Поява форми для внесення оцінок для об’єктивних критеріїв (відображається лише назва критерію та форма для введення оцінки). Після введення оцінок та натискання на кнопку «Підтвердити» з’являється сповіщення про успішне додавання оцінок за всіма об’єктивними критеріями для конкретної обраної роботи до бази даних. Перезавантаження сторінки. Відображення оцінок у загальній таблиці робіт з об’єктивними критеріями.
Стан системи після проведення випробування	Після вибору роботи та об’єктивних критеріїв, якими вона характеризується, з’явилось модальне вікно з формою для введення оцінок. Після натискання на кнопку «Підтвердити» з’явилось сповіщення про успішне додавання оцінок за всіма об’єктивними критеріями до бази даних. Сторінка перезавантажилась. Оцінки за суб’єктивними критерями додались до загальної таблиці.

Таблиця 5.11 – Додавання оцінок для суб’єктивних критеріїв

Мета тесту	Перевірка функції «Додавання оцінок» для суб’єктивних критеріїв.
Початковий стан системи	Відкрите модальне вікно додавання оцінок, що з’явилось після натискання кнопки «Хочете поповнити дані?» на сторінці «Оцінки».
Вхідні дані	Назва роботи та масив суб’єктивних критеріїв.
Схема проведення тесту	Обрати назву роботи з випадаючого списку робіт та обрати суб’єктивні критерії з випадаючого списку критеріїв.
Очікуваний результат	Поява форми для внесення оцінок для суб’єктивних критеріїв (відображається назва критерію та форма для введення оцінки для кожного з існуючих у базі даних експертів). Після введення оцінок та натискання на кнопку «Підтвердити» з’являється сповіщення про успішне додавання оцінок за всіма суб’єктивними критеріями для конкретної обраної роботи до бази даних. Перезавантаження сторінки. Відображення оцінок у загальній таблиці з суб’єктивними критеріями по конкретній роботі.
Стан системи після проведення випробування	Після вибору роботи та суб’єктивних критеріїв, якими вона характеризується, з’явилось модальне вікно з формою для введення оцінок. Після натискання на кнопку «Підтвердити» з’явилось сповіщення про успішне додавання оцінок за всіма суб’єктивними критеріями до бази даних. Сторінка перезавантажилась. Оцінки за суб’єктивними критерями додались до загальної таблиці по конкретній роботі.

Таблиця 5.12 – Робота алгоритму TOPSIS

Мета тесту	Перевірка функції «Визначити першочерговість виконання робіт».
Початковий стан системи	Відкрита сторінка «TOPSIS» та натиснута кнопка «Визначити першочерговість виконання робіт».
Вхідні дані	Матриця альтернатив, вагові коефіцієнти критеріїв, що позначають їх важливість, та тип критерію (на максимум чи на мінімум).
Схема проведення тесту	Ввести вагові коефіцієнти за кожним з критерієм та обрати його тип. Натиснути кнопку «Визначити першочерговість виконання робіт».
Очікуваний результат	Після натискання на кнопку відображається таблиця з зазначеним порядковим номером виконання робіт у порядку спадання їх важливості.
Стан системи після проведення випробування	Відобразилась таблиця з зазначеним порядковим номером виконання робіт у порядку спадання їх важливості.

Таблиця 5.13 – Виведення логів виконання алгоритму TOPSIS

Мета тесту	Перевірка функції «Виведення логів».
Початковий стан системи	Відкрита сторінка «TOPSIS» та здійснені всі кроки зі схеми проведення тесту, описані в таблиці 5.12.
Вхідні дані	Результат виконання алгоритму TOPSIS: матриця нормалізованих оцінок альтернатив, матриця зважених нормалізованих оцінок альтернатив,

	віддаленість альтернатив від позитивної ідеальної точки (PIS), віддаленість альтернатив від негативної ідеальної точки (NIS), наближеність альтернатив до позитивної ідеальної точки (PIS).
Схема проведення тесту	Натиснути на кнопку «Вивести логи». Змінити дані у формі (вагові коефіцієнти та тип). Натиснути на кнопку «Визначити першочерговість виконання робіт» ще раз.
Очікуваний результат	Після натискання на кнопку відображаються таблиці з даними, зазначеними у вхідних даних. Після зміни інформації у формі для введення та натискання кнопки ще раз, дані змінюються.
Стан системи після проведення випробування	Відобразились таблиці з усією інформацію, зазначеною у вхідних даних. Після зміни інформації у формі для введення та повторного натискання кнопки «Визначити першочерговість виконання робіт» дані у таблицях змінились.

Висновок до розділу

В даному розділі було наведене керівництво користувача, були проведені випробування програмного продукту, розглянувши мету випробувань та навівши основні сценарії роботи програми. Всі випробування пройшли успішно.

ЗАГАЛЬНІ ВИСНОВКИ

В індивідуальному дослідницькому проєкті була реалізована інформаційна система підтримки визначення першочерговості виконання робіт. Вона надає користувачам можливість зберігати інформацію про роботи, експертів, критерії, за якими оцінюється пріоритетність виконання роботи, а також інформацію про оцінки об’єктивних чи суб’єктивних критеріїв.

Нижче наведені висновки за кожним розділом пояснювльної записки.

У першому розділі був розроблений опис процесу діяльності та функціональної моделі системи, було розглянуто вже існуючі аналоги, описано призначення розробки інформаційної системи, її цілі та задачі.

У другому розділі було детально описано вхідні дані, що подаються на вхід до системи користувачем, вихідні дані, що є візуальним відображенням результату роботи алгоритмів. Був розроблений детальний опис структури бази даних, що використовувалась при розробці даної системи для збереження інформації про експертів, роботи, критерії та їх оцінки.

У третьому розділі була описана змістовна та математична постановка двох задач, а також методи їх розв’язання. Зазначено, що перший метод використовується для визначення кількісних оцінок суб’єктивних критеріїв, а другий, на основі кількісних оцінок усіх критеріїв, формує першочерговість виконання робіт, сортуючи їх у порядку спадання пріоритетності виконання.

У четвертому розділі було детально описане програмне та технічне забезпечення, а саме: засоби розробки ПЗ, які використовувалися у процесі вирішення задач, архітектура програмного забезпечення, до опису якої входить діаграма класів (функціональних компонентів), діаграма компонентів та детально описана специфікація функцій. В кінці розділу надався приклад оформлення звітів, що є результатами роботи обох алгоритмів.

У п’ятому розділі було наведене керівництво користувача, детально описана вся функціональність інформаційної системи, наведені знімки екрану, які демонструють можливості роботи з системою, також були проведені

випробування програмного продукту. Всі випробування пройшли успішно, що означає, що ПЗ є точним і здатним вирішувати задачі, необхідні користувачеві, воно відповідає функціональним можливостям програмного набору, який є необхідним користувачу.

В розробленому інтерфейсі індивідуального дослідницького проєкту були реалізовані функції створення, редагування та видалення експертів, робіт, критеріїв, а також внесення інформації про оцінки об’єктивних та суб’єктивних критеріїв. Також реалізований перегляд результатів виконання двох алгоритмів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Артим І. І. КІЛЬКІСНІ МЕТОДИ ЕКСПЕРТНОГО ОЦІНЮВАННЯ [Електронний ресурс] / І. І. Артим, Р. Г. Селіверстов, В. П. Новосад // Науково-методична розробка. – 2009. – Режим доступу до ресурсу: http://heqes.kubg.edu.ua/wp-content/uploads/2017/10/Кількісні-методи-експертного-оцінювання.pdf.

2. Ching-Lai Hwang, Kwangsun Yoon. Multiple Attribute Decision Making: Methods and Applications. – Springer-Verlag. Berlin Heidelberg New York, 1981.

3. Нечеткие множества в моделях управления и искусственного интеллекта/Под ред. Д.А.Поспелова. – М.: Наука. Гл. ред. физ.-мат. лит., 1986.–312с.

4. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.). Addison Wesley Longman Inc. ISBN 0-201-30998-X.

5. Що таке PostgreSQL? [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/5154595-what-is-postgresql.

6. Гайна Г.А. Основи проектування баз даних [Текст] / Г. Гайна // Навчальний посібник. − К.: КНУБА, 2005. – 204 с.

7. Saaty, Thomas L.; Peniwati, Kirti (2008). Group Decision Making: Drawing out and Reconciling Differences. Pittsburgh, Pennsylvania: RWS Publications. ISBN 978-1-888603-08-8.

8. Методичні вказівки до виконання домашніх контрольних робіт з курсу КОМП'ЮТЕРНІ СИСТЕМИ [Електронний ресурс] // Методичні вказівки. – 2012. – Режим доступу до ресурсу: https://comsys.kpi.ua/katalog/files/metodichni-vkazivki-domashnoyi-kontrolnoyi-roboti-ks.pdf.

9. Орловский С.А. Проблемы принятия решений при нечеткой исходной информации. – М.: Наука. – 1981. – 194 с.

10. The complexity calculation for group decision making using TOPSIS algorithm [Електронний ресурс] // AIP Conference Proceedings 1755. – 2016. – Режим доступу до ресурсу: https://aip.scitation.org/doi/pdf/10.1063/1.4958502.

11. Чому JavaScript — перспективна мова програмування? [Електронний ресурс] // dou.ua. – 2021. – Режим доступу до ресурсу: https://dou.ua/forums/topic/35184/.

12. React: Making faster, smoother UIs for data-driven Web apps. InfoWorld. Архів оригіналу за 15 грудня 2015. Процитовано 3 березня 2015.

13. Что на самом деле является Node.js? [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/post/420123/.

14. What is Express.js? [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/uk/.

15. What is WebStorm? [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/.

16. Що таке git? [Електронний ресурс] – Режим доступу до ресурсу: https://qagroup.com.ua/publications/shcho-take-git/.

Додаток А

Тексти програмного коду

Інформаційна система підтримки визначення першочерговості виконання робіт

(Найменування програми (документа))

CD-RW

(Вид носія даних)

50 арк, 800 Кб

(Обсяг програми (документа) , арк.,

Київ – 2022 року

AddModal.js

```
import { Button, Modal, Form } from "react-bootstrap";
import styles from "./AddModal.module.scss";

const AddModal = ({
  showModal,
  onHide,
  onSubmit,
  onInput,
  onCancel,
  value,
  name,
  onChange,
}) => {
  const renderName = () => {
    if (name === "роботу") {
      return "назву роботи";
    }
    if (name === "експерта") {
      return "ім'я експерта";
    }
    if (name === "критерій") {
      return "назву критерію";
    }
  };
  return (
    <Modal centered show={showModal} onHide={onHide}>
      <Modal.Header closeButton>
        <Modal.Title>Додайте {name}</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form className="mb-3">
          <Form.Group className="mb-2" controlId="formBasicEmail">
            <Form.Label>Введіть {renderName()}:</Form.Label>
            <Form.Control
              type="text"
              placeholder="Почніть друкувати..."
              onChange={onInput}
              value={value}
            />
          </Form.Group>
          {name === "критерій" && (
            <>
              <Form.Label>Оберіть тип критерію:</Form.Label>
              <Form.Select defaultValue="yes" onChange={onChange}>
                <option>Оберіть тип критерію</option>
                <option value="yes">Об'єктивний</option>
                <option value="no">Суб'єктивний</option>
              </Form.Select>
            </>
          )}
        </Form>
      </Modal.Body>
      <Modal.Footer className={styles.buttons}>
        <Button
          variant="success"
          type="submit"
          className="me-2"
          onClick={onSubmit}
        >
```

```
          Підтвердити
        </Button>
        <Button variant="primary" type="submit" onClick={onCancel}>
          Відміна
        </Button>
      </Modal.Footer>
    </Modal>
  );
};

export default AddModal;
```

Alert.js

```
import { Alert } from "react-bootstrap";
import styles from "./Alert.module.scss";
import { useEffect } from "react";

const CustomAlert = ({ show, variant, message, onClose, setShow }) => {
  useEffect(() => {
    const timeoutId = setTimeout(
      () => setShow({ show: false, variant: "", message: "" }),
      1500
    );
    return () => {
      clearTimeout(timeoutId);
    };
  }, [show]);

  const renderHeader = () => {
    if (variant === "success") {
      return <Alert.Heading>Успіх!</Alert.Heading>;
    }
    if (variant === "danger") {
      return <Alert.Heading>О ні!</Alert.Heading>;
    }
    if (variant === "primary") {
      return <Alert.Heading>Увага!</Alert.Heading>;
    }
  };

  return (
    <Alert
      variant={variant}
      show={show}
      onClose={onClose}
      className={styles.alert}
      dismissible
    >
      {renderHeader()}
      <p>{message}</p>
    </Alert>
  );
};
export default CustomAlert;
```

DeleteModal.js

```
import { Button, Modal } from "react-bootstrap";
import { ObjectUtil } from "../../utils/ObjectUtil";
import styles from "./DeleteModal.module.scss";
```

```
const DeleteModal = ({ showModal, onHide, onDelete, array, id, name }) => {
  return (
    <Modal centered show={showModal} onHide={onHide}>
      <Modal.Header closeButton>
        <Modal.Title>Підтвердіть видалення</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <p>
          Ви впевнені, що хочете видалити {name}
          <span className={styles.text}>
            "
            {ObjectUtil.isEmpty(array) &&
              id &&
              array.find((task) => task.id === id).Name}
            "
          </span>
          ?
        </p>
      </Modal.Body>
      <Modal.Footer className={styles.buttons}>
        <Button variant="success" onClick={onDelete}>
          Так
        </Button>
        <Button variant="danger" onClick={onHide}>
          Ні
        </Button>
      </Modal.Footer>
    </Modal>
  );
};

export default DeleteModal;
```

Footer.js

```
import styles from "./Footer.module.scss";

export default function Footer() {
  return (
    <section className={styles.footer}>
      <p>
        © 2022 | Індивідуальний дослідницький проєкт виконала з{" "}
        <span className={styles.footerHeart}> ♥ </span>
        студентка групи ІС-81, Бишовець Наталія
      </p>
    </section>
  );
}
```

Header.js

```
import LogoTitle from './LogoTitle';
import styles from './Header.module.scss';
import NavBar from '../NavBar';

export default function Header() {
  return (
    <header className={styles.header}>
      <LogoTitle />
      <NavBar />
    </header>
  );
```

```
}
```

LogoTitle.js

```
import Logo from "../../../images/graduate.svg";
import styles from "./LogoTitle.module.scss";
import { NavLink } from "react-router-dom";

export default function LogoTitle() {
  return (
    <NavLink
      className={(navData) =>
        navData.isActive ? styles.activeName : styles.name
      }
      to="/"
      exact="true"
    >
      <img src={Logo} className={styles.logo} alt="Diploma logo" />
      Diploma
    </NavLink>
  );
}
```

NavBar.js

```
import { NavLink } from "react-router-dom";
import styles from "./NavBar.module.scss";

export default function NavBar() {
  return (
    <nav className={styles.container}>
      <NavLink
        to="/topsis"
        exact="true"
        className={(navData) =>
          navData.isActive ? styles.activeLink : styles.link
        }
      >
        TOPSIS
      </NavLink>
      <NavLink
        to="/experts"
        exact="true"
        className={(navData) =>
          navData.isActive ? styles.activeLink : styles.link
        }
      >
        Експерти
      </NavLink>
      <NavLink
        to="/criteria"
        exact="true"
        className={(navData) =>
          navData.isActive ? styles.activeLink : styles.link
        }
      >
        Критерії
      </NavLink>
      <NavLink
        to="/tasks"
        exact="true"
        className={(navData) =>
```

```
          navData.isActive ? styles.activeLink : styles.link
        }
      >
        Роботи
      </NavLink>
      <NavLink
        to="/values"
        exact="true"
        className={(navData) =>
          navData.isActive ? styles.activeLink : styles.link
        }
      >
        Оцінки
      </NavLink>
    </nav>
  );
}
```

TaskCriteriaCreateForm.js

```
import { Button, Modal, Form } from "react-bootstrap";
import styles from "./TaskCriteriaCreateForm.module.scss";
import axios from "axios";
import Constants from "../../utils/Constants";
import { useState, useEffect } from "react";
import { ObjectUtil } from "../../utils/ObjectUtil";
import Select from "react-select";
import QualityCriteriaTable from "../QualityCriteriaTable";
import NonQualityCriteriaTable from "../NonQualityCriteriaTable";
import Alert from "../../components/Alert";

const TaskCriteriaCreateForm = ({ showModal, onHide, onSubmit }) => {
  const [showAlert, setShowAlert] = useState({
    show: false,
    variant: "",
    message: "",
  });
  const [tasks, setTasks] = useState({});
  const [criteria, setCriteria] = useState({});
  const [experts, setExperts] = useState({});
  const [taskId, setTaskId] = useState();
  const [criteriaId, setCriteriaId] = useState([]);
  const [add, setAdd] = useState(false);
  const [inputQualityValue, setInputQualityValue] = useState({});
  const [inputNonQualityValue, setInputNonQualityValue] = useState({});

  useEffect(() => {
    const fetchTasks = async () => {
      const { data } = await axios.get(`${Constants.BASE_URI}getTasks`);

      setTasks(
        data.map((task) => {
          return {
            value: { id: task.id, name: task.Name },
            label: task.Name,
          };
        })
      );
    };
    if (!Object.keys(tasks).length) {
      fetchTasks();
```

```
  }
  const fetchExperts = async () => {
    const { data } = await axios.get(`${Constants.BASE_URI}getExperts`);
    setExperts(
      data.map((expert) => {
        return {
          value: { id: expert.id, name: expert.Name },
          label: expert.Name,
        };
      })
    );
  };
  if (!Object.keys(experts).length) {
    fetchExperts();
  }
  const fetchCriteria = async () => {
    const { data } = await axios.get(`${Constants.BASE_URI}getCriteria`);
    setCriteria(
      data.map((criteria) => {
        return {
          value: {
            id: criteria.id,
            name: criteria.Name,
            isQuality: criteria.IsQualityCriteria,
          },
          label: criteria.Name,
        };
      })
    );
  };
  if (!Object.keys(criteria).length) {
    fetchCriteria();
  }
}, []);

const onChangeCriteria = (array) => {
  setCriteriaId(array.map(({ value }) => value));
};

const onChangeTask = ({ value }) => {
  setTaskId(value);
};

const onCloseAlert = () => {
  setShowAlert(false);
};

const onClick = () => {
  if (taskId && criteriaId.length) {
    setAdd(true);

    const checkQualities =
      Object.keys(inputQualityValue).length ===
      criteriaId.filter((crit) => crit.isQuality).length;

    const checkNonQualities =
      Object.keys(inputNonQualityValue).length === experts.length &&
      Object.entries(inputNonQualityValue).every(
        (value) => Object.entries(value)[1].length === 2
      );
```

```
    if (checkQualities && checkNonQualities) {
      onSubmit(inputQualityValue, inputNonQualityValue, taskId);
    } else {
      setShowAlert({
        show: true,
        variant: "primary",
        message: "Заповніть всі поля!",
      });
    }
  }

  if (!taskId || !criteriaId.length) {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Заповнено не всі поля!",
    });
  }
};

return (
  <>
    <Modal
      centered
      show={showModal}
      onHide={onHide}
      dialogClassName={styles.modal}
    >
      <Modal.Header closeButton>
        <Modal.Title>Введіть дані:</Modal.Title>
      </Modal.Header>
      {!add ? (
        <Modal.Body>
          <Form className="mb-3">
            <Form.Group className="mb-2">
              <Form.Label>Оберіть роботу:</Form.Label>
              <Select
                options={ObjectUtil.isEmpty(tasks) && tasks}
                className="basic-select"
                classNamePrefix="select"
                placeholder="Оберіть роботу.."
                onChange={onChangeTask}
              />
            </Form.Group>

            <Form.Group className="mb-3">
              <Form.Label>Оберіть критерії:</Form.Label>
              <Select
                options={ObjectUtil.isEmpty(criteria) && criteria}
                className="basic-multi-select"
                classNamePrefix="select"
                placeholder="Оберіть критерії.."
                onChange={onChangeCriteria}
                isMulti
                closeMenuOnSelect={false}
              />
            </Form.Group>
          </Form>
        </Modal.Body>
```

```
      ) : (
        <Modal.Body styles={styles.modal}>
          <h5>Робота "{taskId.name}"</h5>
          <QualityCriteriaTable
            values={criteriaId.filter((crit) => crit.isQuality)}
            inputValue={inputQualityValue}
            setInputValue={setInputQualityValue}
          />
          <NonQualityCriteriaTable
            values={criteriaId.filter((crit) => !crit.isQuality)}
            experts={experts}
            inputValue={inputNonQualityValue}
            setInputValue={setInputNonQualityValue}
          />
        </Modal.Body>
      )}
      <Modal.Footer className={styles.buttons}>
        <Button
          variant="success"
          type="submit"
          className="me-2"
          onClick={onClick}
        >
          Підтвердити
        </Button>
        <Button variant="danger" type="submit" onClick={onHide}>
          Відміна
        </Button>
      </Modal.Footer>
    </Modal>
    <Alert
      show={showAlert.show}
      variant={showAlert.variant}
      message={showAlert.message}
      onClose={onCloseAlert}
      setShow={setShowAlert}
    />
  </>
 );
};

export default TaskCriteriaCreateForm;
```

NonQualityCriteriaTable.js

```
import { Table, InputGroup, Form } from "react-bootstrap";
import styles from "../styles/modules/Table.module.scss";

export default function NonQualityCriteriaTable({
 values,
 experts,
 inputValue,
 setInputValue,
}) {
 const handleInputChange = ({ target: { value } }, expertId, criteriaId) => {
  setInputValue({
   ...inputValue,
   [expertId]: { ...inputValue[expertId], [criteriaId]: value },
  });
 };
```

```
  return (
    <>
      {values.length ? (
        <>
          <h6>Суб'єктивні критерії:</h6>
          <Table striped bordered>
            <thead>
              <tr>
                <th>№</th>
                <th>Експерт</th>
                {values.map((criteria) => (
                  <th key={criteria.id}>{criteria.name}</th>
                ))}
              </tr>
            </thead>
            <tbody>
              {experts.map((expert, idx) => (
                <tr key={idx}>
                  <td className={styles.sequence}>{idx + 1}</td>
                  <td className={styles.type}>{expert.value.name}</td>
                  {values.map((criteria) => (
                    <td>
                      <InputGroup>
                        <Form.Control
                          type="number"
                          min={0}
                          max={10}
                          placeholder="Введіть значення критерію..."
                          value={inputValue.value}
                          onChange={(event) =>
                            handleInputChange(
                              event,
                              expert.value.id,
                              criteria.id
                            )
                          }
                        />
                      </InputGroup>
                    </td>
                  ))}
                </tr>
              ))}
            </tbody>
          </Table>
        </>
      ) : (
        <p>Жодного суб'єктивного критерію не обрано</p>
      )}
    </>
  );
}
```

QualityCriteriaTable.js

```
import { Table, InputGroup, Form } from "react-bootstrap";
import styles from "../styles/modules/Table.module.scss";

export default function QualityCriteriaTable({
  values,
  inputValue,
  setInputValue,
```

```
}) {
  const handleInputChange = ({ target: { value } }, id) => {
    setInputValue({ ...inputValue, [id]: value });
  };

  return (
    <>
      {values.length ? (
        <>
          <h6 className="">Об'єктивні критерії:</h6>
          <Table striped bordered>
            <thead>
              <tr>
                <th>№</th>
                <th>Назва</th>
                <th>Значення</th>
              </tr>
            </thead>
            <tbody>
              {values.map((value, idx) => (
                <tr key={value.id}>
                  <td className={styles.sequence}>{idx + 1}</td>
                  <td className={styles.type}>{value.name}</td>
                  <td>
                    <InputGroup>
                      <Form.Control
                        type="number"
                        min={0}
                        placeholder="Введіть значення критерію..."
                        value={inputValue.value}
                        onChange={(event) => handleInputChange(event, value.id)}
                      />
                    </InputGroup>
                  </td>
                </tr>
              ))}
            </tbody>
          </Table>
        </>
      ) : (
        <p className="mb-3">Жодного об'єктивного критерію не обрано.</p>
      )}
    </>
  );
}
```

Topsis.js

```
import { Table, Form, Button } from "react-bootstrap";
import { useState } from "react";
import Select from "react-select";
import calculateTopsis from "../utils/CalculateTopsis";
import styles from "../styles/modules/TopsisPage.module.scss";

export default function Topsis({ values, formattedCriteria }) {
  const [show, setShow] = useState(false);
  const [info, setInfo] = useState();
  const [showLogs, setShowLogs] = useState(false);

  const alternatives = [];
  Object.entries(values).forEach((value) => alternatives.push(value[1]));
```

```
const tasks = Object.entries(values).reduce((acc, curr, i) => {
  acc[`task-${i}`] = curr[0];
  return acc;
}, {});

const [weights, setWeights] = useState(
  alternatives[0].reduce((acc, curr, i) => {
    acc[i] = 1;
    return acc;
  }, {})
);

const [criteria, setCriteria] = useState(
  alternatives[0].reduce((acc, curr, i) => {
    acc[i] = "max";
    return acc;
  }, {})
);

const handleInputChange = ({ target: { value } }, criteriaId) => {
  setWeights({
    ...weights,
    [criteriaId]: Number(value),
  });
};

const topsis = () => {
  const dtoOut = calculateTopsis(alternatives, weights, tasks, criteria);
  setShow(true);
  setInfo(dtoOut);
};

const onChangeCriteria = ({ value }, criteriaId) => {
  setCriteria({
    ...criteria,
    [criteriaId]: value,
  });
};

return (
  <>
    <h6>Введіть множину вагових оцінок критеріїв:</h6>
    <Table striped bordered className="mb-2">
      <thead>
        <tr>
          <th>Ваги</th>
          {alternatives[0].map((arr, i) => (
            <th key={i}>W{i + 1}</th>
          ))}
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>Значення</td>
          {alternatives[0].map((arr, i) => {
            return (
              <td key={i}>
                <Form.Control
                  type="number"
                  min={1}
```

```
                value={weights[i]}
                onChange={(event) => handleInputChange(event, i)}
              />
            </td>
          );
        })}
      </tr>
      <tr>
        <td>Критерії</td>
        {alternatives[0].map((arr, i) => {
          return (
            <td key={i}>
              <Select
                options={[
                  { value: "max", label: "max" },
                  { value: "min", label: "min" },
                ]}
                className="basic-select"
                classNamePrefix="select"
                placeholder="Оберіть критерій.."
                onChange={(event) => onChangeCriteria(event, i)}
                defaultValue={{ label: "max", value: "max" }}
              />
            </td>
          );
        })}
      </tr>
    </tbody>
  </Table>
  <Button
    onClick={topsis}
    className={styles.button}
    variant="outline-primary"
  >
    Виконати розрахунки методом TOPSIS
  </Button>
  {show && (
    <>
      <h3 className="mb-2">Порядок виконання робіт:</h3>
      <Table striped bordered>
        <thead>
          <tr>
            <th>Порядок виконання</th>
            {Object.values(info.order).map((or, i) => {
              return <th key={i}>{i + 1}</th>;
            })}
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Назви робіт</td>
            {Object.values(info.order).map((task, i) => {
              return <td key={i}>{task}</td>;
            })}
          </tr>
          <tr>
            <td>Ваги</td>
            {Object.values(info.distances).map((distance, i) => {
              return <td key={i}>{distance}</td>;
            })}
```

```
        </tr>
      </tbody>
    </Table>
    <Button
      onClick={() => setShowLogs(!showLogs)}
      className="mb-2"
      variant="outline-info"
    >
      Вивести логи
    </Button>
  </>
)}

{showLogs && (
  <>
    <h3 className="mb-2">Нормалізовані оцінки альтернатив</h3>
    <Table striped bordered>
      <thead>
        <tr>
          <th>Назви робіт</th>
          {formattedCriteria.map((key) => {
            return <th key={key}>{key}</th>;
          })}
        </tr>
      </thead>
      <tbody>
        {Object.keys(values).map((task, i) => {
          return (
            <tr key={i}>
              <td key={i}>{task}</td>
              {info.normalizedAlternatives[i].map((alternatives) => (
                <td>{alternatives}</td>
              ))}
            </tr>
          );
        })}
      </tbody>
    </Table>

    <h3 className="mb-2">Зважені нормалізовані оцінки альтернатив</h3>
    <Table striped bordered>
      <thead>
        <tr>
          <th>Назви робіт</th>
          {formattedCriteria.map((key) => {
            return <th key={key}>{key}</th>;
          })}
        </tr>
      </thead>
      <tbody>
        {Object.keys(values).map((task, i) => {
          return (
            <tr key={i}>
              <td key={i}>{task}</td>
              {info.normalizedAlternatives[i].map((alternatives) => (
                <td>{alternatives}</td>
              ))}
            </tr>
          );
        })}
```

```
  </tbody>
</Table>

<h3 className="mb-2">
  Віддаленість альтернатив від позитивної ідеальної точки (PIS)
</h3>
<Table striped bordered>
  <thead>
    <tr>
      <th>Віддаленість до PIS</th>
      {Object.keys(values).map((task) => {
        return <th key={task}>{task}</th>;
      })}
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>S+</td>
      {info.PIS.map((alternatives) => (
        <td>{alternatives}</td>
      ))}
    </tr>
  </tbody>
</Table>

<h3 className="mb-2">
  Віддаленість альтернатив від негативної ідеальної точки (NIS)
</h3>
<Table striped bordered>
  <thead>
    <tr>
      <th>Віддаленість до NIS</th>
      {Object.keys(values).map((task) => {
        return <th key={task}>{task}</th>;
      })}
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>S-</td>
      {info.NIS.map((alternatives) => (
        <td>{alternatives}</td>
      ))}
    </tr>
  </tbody>
</Table>

<h3 className="mb-2">
  Наближеність альтернатив до позитивної ідеальної точки (PIS)
</h3>
<Table striped bordered>
  <thead>
    <tr>
      <th>Наближеність до PIS</th>
      {Object.values(info.order).map((task) => {
        return <th key={task}>{task}</th>;
      })}
    </tr>
  </thead>
  <tbody>
```

```
            <tr>
              <td>C*</td>
              {Object.values(info.distances).map((distance) => (
                <td>{distance}</td>
              ))}
            </tr>
          </tbody>
        </Table>
      </>
    )}
  </>
 );
}
```

CriteriaPage.js

```
import axios from "axios";
import { useState, useEffect } from "react";
import { Table, Button, Form } from "react-bootstrap";
import { ObjectUtil } from "../utils/ObjectUtil";
import Constants from "../utils/Constants";
import styles from "../styles/modules/Table.module.scss";
import AddModal from "../components/AddModal";
import DeleteModal from "../components/DeleteModal";
import Alert from "../components/Alert";

export default function CriteriaPage() {
  const [criteria, setCriteria] = useState({});
  const [edit, setEdit] = useState(false);
  const [value, setValue] = useState("");
  const [type, setType] = useState(true);
  const [valueCriteria, setValueCriteria] = useState("");
  const [id, setId] = useState();
  const [showAddModal, setShowAddModal] = useState(false);
  const [showDeleteModal, setShowDeleteModal] = useState(false);
  const [showAlert, setShowAlert] = useState({
    show: false,
    variant: "",
    message: "",
  });

  useEffect(() => {
    const fetchCallback = async () => {
      const { data } = await axios.get(`${Constants.BASE_URI}getCriteria`);

      setCriteria(data);
    };
    if (!Object.keys(criteria).length) {
      fetchCallback();
    }
  }, []);

  const onUpdate = async (name, id) => {
    setEdit(true);
    setValueCriteria(name);
    setId(id);
  };

  const onSubmit = async (event) => {
    event.preventDefault();
```

```
  if (value !== "") {
    try {
      await axios.post(`${Constants.BASE_URI}addCriteria`, {
        name: value,
        isQuality: type,
      });
      setShowAddModal(false);
      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно додано критерій!",
      });
      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
    setValue("");
  } else {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onCancel = () => {
  setValue("");
  setShowAddModal(false);
};

const onInput = ({ target: { value } }) => {
  setValue(value);
};

const onChange = ({ target: { value } }) => {
  setType(value === "yes");
};

const onInputTask = ({ target: { value } }) => {
  setValueCriteria(value);
};

const onDelete = async () => {
  try {
    await axios.post(`${Constants.BASE_URI}deleteCriteria`, { id });
    setShowDeleteModal(false);
    setEdit(false);
    setShowAlert({
      show: true,
      variant: "success",
      message: "Успішно видалено критерій!",
    });
    setTimeout(() => window.location.reload(), 500);
  } catch (e) {
```

```
    setShowAlert({
      show: true,
      variant: "danger",
      message: "Щось пішло не так. Спробуйте пізніше!",
    });
    throw e;
  }
};

const onSubmitUpdate = async (name, currType) => {
  if ((name !== valueCriteria && valueCriteria !== "") || currType !== type) {
    try {
      await axios.post(`${Constants.BASE_URI}updateCriteria`, {
        id,
        name: valueCriteria,
        isQuality: type,
      });
      setValueCriteria("");
      setEdit(false);

      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно оновлено критерій!",
      });

      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
  }
  if (name === valueCriteria) {
    setEdit(false);
  }
  if (valueCriteria === "") {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onOpenDeleteModal = (id) => {
  setShowDeleteModal(true);
  setId(id);
};

const onHideAdd = () => {
  setShowAddModal(false);
};

const onOpenAddModal = () => {
  setShowAddModal(true);
};
```

```
const onCloseAlert = () => {
  setShowAlert({ show: false, variant: "", message: "" });
};

const onHideDelete = () => {
  setShowDeleteModal(false);
};

return (
  <>
    <Button onClick={onOpenAddModal} className="mb-3">
      Хочете додати критерій?
    </Button>
    <Table striped bordered>
      <thead>
        <tr>
          <th>№</th>
          <th>Назва</th>
          <th className={styles.type}>Тип</th>
          <th colSpan={2} className={styles.actions}>
            Дії
          </th>
        </tr>
      </thead>
      <tbody>
        {ObjectUtil.isEmpty(criteria) &&
          criteria.map((crit, i) => {
            return (
              <tr key={crit.id}>
                <td className={styles.sequence}>{i + 1}</td>

                {edit && id === crit.id && (
                  <td>
                    <Form.Group controlId="formBasicEmail">
                      <Form.Control
                        type="text"
                        placeholder="Введіть нову назву..."
                        value={valueCriteria}
                        onChange={onInputTask}
                      />
                    </Form.Group>
                  </td>
                )}
                {!edit && <td> {crit.Name}</td>}
                {edit && id !== crit.id && <td> {crit.Name}</td>}

                {!edit && (
                  <td>
                    {crit.IsQualityCriteria ? "Об'єктивний" : "Суб'єктивний"}
                  </td>
                )}
                {edit && id !== crit.id && (
                  <td>
                    {crit.IsQualityCriteria ? "Об'єктивний" : "Суб'єктивний"}
                  </td>
                )}
                {edit && id === crit.id && (
                  <td>
                    <Form.Select
```

```
                defaultValue={crit.IsQualityCriteria ? "yes" : "no"}
                onChange={onChange}
              >
                <option>Оберіть тип критерію</option>
                <option value="yes">Об'єктивний</option>
                <option value="no">Суб'єктивний</option>
              </Form.Select>
            </td>
          )}

          <td className={styles.actionBtn}>
            {edit && id !== crit.id && (
              <Button
                variant="warning"
                onClick={() => onUpdate(crit.Name, crit.id)}
              >
                Відредагувати
              </Button>
            )}
            {!edit && (
              <Button
                variant="warning"
                onClick={() => onUpdate(crit.Name, crit.id)}
              >
                Відредагувати
              </Button>
            )}

            {edit && id === crit.id && (
              <Button
                variant="success"
                onClick={() =>
                  onSubmitUpdate(crit.Name, crit.IsQualityCriteria)
                }
              >
                Підтвердити
              </Button>
            )}
          </td>

          <td className={styles.actionBtn}>
            <Button
              variant="danger"
              onClick={() => onOpenDeleteModal(crit.id)}
            >
              Видалити
            </Button>
          </td>
        </tr>
      );
    })}
  </tbody>
</Table>
<AddModal
  showModal={showAddModal}
  onHide={onHideAdd}
  onSubmit={onSubmit}
  onInput={onInput}
  onCancel={onCancel}
  value={value}
```

```
        name={"критерій"}
        onChange={onChange}
      />
      <DeleteModal
        showModal={showDeleteModal}
        onHide={onHideDelete}
        onDelete={onDelete}
        array={criteria}
        id={id}
        name={"критерій "}
      />
      <Alert
        show={showAlert.show}
        variant={showAlert.variant}
        message={showAlert.message}
        onClose={onCloseAlert}
        setShow={setShowAlert}
      />
    </>
  );
}
```

ExpertPage.js

```
import axios from "axios";
import { useState, useEffect } from "react";
import { Table, Button, Form, InputGroup } from "react-bootstrap";
import { ObjectUtil } from "../utils/ObjectUtil";
import Constants from "../utils/Constants";
import styles from "../styles/modules/Table.module.scss";
import DeleteModal from "../components/DeleteModal";
import AddModal from "../components/AddModal";
import Alert from "../components/Alert";

export default function ExpertPage() {
  const [experts, setExperts] = useState({});
  const [edit, setEdit] = useState(false);
  const [value, setValue] = useState("");
  const [valueExpert, setValueExpert] = useState("");
  const [id, setId] = useState();
  const [showDeleteModal, setShowDeleteModal] = useState(false);
  const [showAddModal, setShowAddModal] = useState(false);
  const [showAlert, setShowAlert] = useState({
    show: false,
    variant: "",
    message: "",
  });

  useEffect(() => {
    const fetchCallback = async () => {
      const { data } = await axios.get(`${Constants.BASE_URI}getExperts`);

      setExperts(data);
    };
    if (!Object.keys(experts).length) {
      fetchCallback();
    }
  }, []);

  const onUpdate = async (name, id) => {
```

```
  setEdit(true);
  setValueExpert(name);
  setId(id);
};

const onSubmit = async (event) => {
  event.preventDefault();

  if (value !== "") {
    try {
      await axios.post(`${Constants.BASE_URI}addExpert`, { name: value });
      setShowAddModal(false);
      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно додано експерта!",
      });
      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
    setValue("");
  } else {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onCancel = () => {
  setShowAddModal(false);
};

const onCancelExpert = () => {
  setEdit(false);
  setValueExpert("");
};

const onInput = ({ target: { value } }) => {
  setValue(value);
};

const onInputExpert = ({ target: { value } }) => {
  setValueExpert(String(value));
};

const onOpenDeleteModal = (id) => {
  setShowDeleteModal(true);
  setId(id);
};

const onOpenAddModal = () => {
  setShowAddModal(true);
```

```
};

const onHideDelete = () => {
  setShowDeleteModal(false);
};

const onHideAdd = () => {
  setShowAddModal(false);
};

const onDelete = async () => {
  try {
    await axios.post(`${Constants.BASE_URI}deleteExpert`, { id });
    setShowDeleteModal(false);
    setEdit(false);
    setShowAlert({
      show: true,
      variant: "success",
      message: "Успішно видалено експерта!",
    });
    setTimeout(() => window.location.reload(), 500);
  } catch (e) {
    setShowAlert({
      show: true,
      variant: "danger",
      message: "Щось пішло не так. Спробуйте пізніше!",
    });
    throw e;
  }
};

const onSubmitUpdate = async (name) => {
  if (name !== valueExpert && valueExpert !== "") {
    try {
      await axios.post(`${Constants.BASE_URI}updateExpert`, {
        id,
        name: valueExpert,
      });
      setValueExpert("");
      setEdit(false);

      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно оновлено експерта!",
      });

      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
  }
  if (name === valueExpert) {
    setEdit(false);
  }
```

```
  if (valueExpert === "") {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onCloseAlert = () => {
  setShowAlert({ show: false, variant: "", message: "" });
};

return (
  <>
    <Button onClick={onOpenAddModal} className="mb-3">
      Хочете додати експерта?
    </Button>
    <Table striped bordered>
      <thead>
        <tr>
          <th>№</th>
          <th>Ім'я</th>
          <th colSpan={2} className={styles.actions}>
            Дії
          </th>
        </tr>
      </thead>
      <tbody>
        {ObjectUtil.isEmpty(experts) &&
          experts.map((expert, i) => {
            return (
              <tr key={expert.id}>
                <td className={styles.sequence}>{i + 1}</td>
                {edit && id === expert.id && (
                  <td>
                    <InputGroup>
                      <Form.Control
                        type="text"
                        placeholder="Введіть нову назву..."
                        value={valueExpert}
                        onChange={onInputExpert}
                      />
                      <Button
                        variant="success"
                        onClick={() => onSubmitUpdate(expert.Name)}
                      >
                        Підтвердити
                      </Button>
                    </InputGroup>
                  </td>
                )}
                {!edit && <td> {expert.Name}</td>}
                {edit && id !== expert.id && <td> {expert.Name}</td>}
                <td className={styles.actionBtn}>
                  {edit && id !== expert.id && (
                    <Button
                      variant="warning"
                      onClick={() => onUpdate(expert.Name, expert.id)}
                    >
```

```
                    Відредагувати
                  </Button>
                )}
                {!edit && (
                  <Button
                    variant="warning"
                    onClick={() => onUpdate(expert.Name, expert.id)}
                  >
                    Відредагувати
                  </Button>
                )}
                {edit && id === expert.id && (
                  <Button
                    variant="primary"
                    type="submit"
                    onClick={onCancelExpert}
                  >
                    Відміна
                  </Button>
                )}
              </td>
              <td className={styles.actionBtn}>
                <Button
                  variant="danger"
                  onClick={() => onOpenDeleteModal(expert.id)}
                >
                  Видалити
                </Button>
              </td>
            </tr>
          );
        })}
      </tbody>
    </Table>
    <DeleteModal
      showModal={showDeleteModal}
      onHide={onHideDelete}
      onDelete={onDelete}
      array={experts}
      id={id}
      name={"експерта "}
    />
    <AddModal
      showModal={showAddModal}
      onHide={onHideAdd}
      onSubmit={onSubmit}
      onInput={onInput}
      onCancel={onCancel}
      value={value}
      name={"експерта"}
    />
    <Alert
      show={showAlert.show}
      variant={showAlert.variant}
      message={showAlert.message}
      onClose={onCloseAlert}
      setShow={setShowAlert}
    />
  </>
);
```

```
}
```

MainPage.js

```
import mainImg from "../images/make-decision.jpg";
import styles from "../styles/modules/MainPage.module.scss";

export default function MainPage() {
  return (
    <>
      <h1 className={styles.header}>
        Інформаційна система підтримки визначення першочерговості виконання
        робіт
      </h1>
      <img src={mainImg} alt="Main image" className={styles.image} />
      <h5 className={styles.text}>
        Цей застосунок розроблений для того, щоб спростити процес прийняття
        рішень на основі об'єктивних та суб'єктивних оцінок альтернатив за
        деякими критеріями. Скористайтесь меню в навігації, аби дослідити весь
        його функціонал.
      </h5>
    </>
  );
}
```

TaskPage.js

```
import axios from "axios";
import { useState, useEffect } from "react";
import { Table, Button, Form, InputGroup } from "react-bootstrap";
import { ObjectUtil } from "../utils/ObjectUtil";
import Constants from "../utils/Constants";
import styles from "../styles/modules/Table.module.scss";
import DeleteModal from "../components/DeleteModal";
import Alert from "../components/Alert";
import AddModal from "../components/AddModal";

export default function TaskPage() {
  const [tasks, setTasks] = useState({});
  const [edit, setEdit] = useState(false);
  const [value, setValue] = useState("");
  const [valueTask, setValueTask] = useState("");
  const [id, setId] = useState();
  const [showDeleteModal, setShowDeleteModal] = useState(false);
  const [showAddModal, setShowAddModal] = useState(false);
  const [showAlert, setShowAlert] = useState({
    show: false,
    variant: "",
    message: "",
  });

  useEffect(() => {
    const fetchCallback = async () => {
      const { data } = await axios.get(`${Constants.BASE_URI}getTasks`);

      setTasks(data);
    };
    if (!Object.keys(tasks).length) {
      fetchCallback();
    }
  }, []);
```

```
const onUpdate = async (name, id) => {
  setEdit(true);
  setValueTask(name);
  setId(id);
};

const onSubmit = async (event) => {
  event.preventDefault();

  if (value !== "") {
    try {
      await axios.post(`${Constants.BASE_URI}addTask`, { name: value });
      setShowAddModal(false);
      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно додано роботу!",
      });
      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
    setValue("");
  } else {
    setShowAlert({
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onCancel = () => {
  setValue("");
  setShowAddModal(false);
};

const onInput = ({ target: { value } }) => {
  setValue(value);
};

const onInputTask = ({ target: { value } }) => {
  setValueTask(value);
};

const onOpenDeleteModal = (id) => {
  setShowDeleteModal(true);
  setId(id);
};

const onCancelTask = () => {
  setEdit(false);
  setValueTask("");
};
```

```
const onHideDelete = () => {
  setShowDeleteModal(false);
};

const onHideAdd = () => {
  setShowAddModal(false);
};

const onDelete = async () => {
  try {
    await axios.post(`${Constants.BASE_URI}deleteTask`, { id });
    setShowDeleteModal(false);
    setEdit(false);
    setShowAlert({
      show: true,
      variant: "success",
      message: "Успішно видалено роботу!",
    });
    setTimeout(() => window.location.reload(), 500);
  } catch (e) {
    setShowAlert({
      show: true,
      variant: "danger",
      message: "Щось пішло не так. Спробуйте пізніше!",
    });
    throw e;
  }
};

const onSubmitUpdate = async (name) => {
  if (name !== valueTask && valueTask !== "") {
    try {
      await axios.post(`${Constants.BASE_URI}updateTask`, {
        id,
        name: valueTask,
      });
      setValueTask("");
      setEdit(false);

      setShowAlert({
        show: true,
        variant: "success",
        message: "Успішно оновлено роботу!",
      });

      setTimeout(() => window.location.reload(), 500);
    } catch (e) {
      setShowAlert({
        show: true,
        variant: "danger",
        message: "Щось пішло не так. Спробуйте пізніше!",
      });
      throw e;
    }
  }
  if (name === valueTask) {
    setEdit(false);
  }
  if (valueTask === "") {
    setShowAlert({
```

```
      show: true,
      variant: "primary",
      message: "Ви нічого не ввели.",
    });
  }
};

const onOpenAddModal = () => {
  setShowAddModal(true);
};

const onCloseAlert = () => {
  setShowAlert({ show: false, variant: "", message: "" });
};

return (
  <>
    <Button onClick={onOpenAddModal} className="mb-3">
      Хочете додати роботу?
    </Button>
    <Table striped bordered>
      <thead>
        <tr>
          <th>№</th>
          <th>Назва</th>
          <th colSpan={2} className={styles.actions}>
            Дії
          </th>
        </tr>
      </thead>
      <tbody>
        {ObjectUtil.isEmpty(tasks) &&
          tasks.map((task, i) => {
            return (
              <tr key={task.id}>
                <td className={styles.sequence}>{i + 1}</td>
                {edit && id === task.id && (
                  <td>
                    <InputGroup>
                      <Form.Control
                        type="text"
                        placeholder="Введіть нову назву..."
                        value={valueTask}
                        onChange={onInputTask}
                      />
                      <Button
                        variant="success"
                        onClick={() => onSubmitUpdate(task.Name)}
                      >
                        Підтвердити
                      </Button>
                    </InputGroup>
                  </td>
                )}
                {!edit && <td> {task.Name}</td>}
                {edit && id !== task.id && <td> {task.Name}</td>}
                <td className={styles.actionBtn}>
                  {edit && id !== task.id && (
                    <Button
                      variant="warning"
```

```
                onClick={() => onUpdate(task.Name, task.id)}
               >
                Відредагувати
               </Button>
              )}
              {!edit && (
               <Button
                variant="warning"
                onClick={() => onUpdate(task.Name, task.id)}
               >
                Відредагувати
               </Button>
              )}
              {edit && id === task.id && (
               <Button
                variant="primary"
                type="submit"
                onClick={onCancelTask}
               >
                Відміна
               </Button>
              )}
             </td>
             <td className={styles.actionBtn}>
              <Button
               variant="danger"
               onClick={() => onOpenDeleteModal(task.id)}
              >
               Видалити
              </Button>
             </td>
            </tr>
           );
          })}
        </tbody>
       </Table>
       <DeleteModal
        showModal={showDeleteModal}
        onHide={onHideDelete}
        onDelete={onDelete}
        array={tasks}
        id={id}
        name={"роботу "}
       />
       <AddModal
        showModal={showAddModal}
        onHide={onHideAdd}
        onSubmit={onSubmit}
        onInput={onInput}
        onCancel={onCancel}
        value={value}
        name={"роботу"}
       />
       <Alert
        show={showAlert.show}
        variant={showAlert.variant}
        message={showAlert.message}
        onClose={onCloseAlert}
        setShow={setShowAlert}
       />
```

```
    </>
  );
}
```

TopsisPage.js

```
import { Button, Form, Table } from "react-bootstrap";
import { useEffect, useState } from "react";
import axios from "axios";
import { ObjectUtil } from "../utils/ObjectUtil";
import { ArrayUtil } from "../utils/ArrayUtil";
import Constants from "../utils/Constants";
import calculateExpertValues from "../utils/experts-algorithm/CalculateExpertValues";
import Topsis from "../components/Topsis";
import styles from "../styles/modules/TopsisPage.module.scss";

export default function TopsisPage() {
  const [nonQualityTasks, setNonQualityTasks] = useState({});
  const [tasks, setTasks] = useState({});
  const [formattedData, setFormattedData] = useState({
    criteria: [],
    qualityTasks: {},
  });
  const [, setQualityTasks] = useState({});
  const [showAlgorithm, setShowAlgorithm] = useState(false);

  useEffect(() => {
    const fetchCallback = async () => {
      const preFormattedData = ObjectUtil.getClone(formattedData);
      const nonQualifiedCriteria = {};
      let temp1 = {};
      const rawNonQualifiedCriteria = [];

      const { data } = await axios.get(`${Constants.BASE_URI}`);
      const qualifiedCriteria = data.filter((item) => {
        if (!item.isqualitycriteria) {
          rawNonQualifiedCriteria.push(item);
        }
        return item.isqualitycriteria;
      });
      qualifiedCriteria.forEach((elem) => {
        if (preFormattedData.criteria.indexOf(elem.criterianame) === -1) {
          preFormattedData.criteria.push(elem.criterianame);
        }
      });
      setTasks(data);

      qualifiedCriteria.forEach((task) => {
        const { taskname, CriteriaValue } = task;
        ArrayUtil.setEmptyArrayOnEmpty(preFormattedData.qualityTasks, taskname);
        preFormattedData.qualityTasks[taskname].push(CriteriaValue);
      });

      rawNonQualifiedCriteria.forEach((task) => {
        const { taskname, criterianame, CriteriaValue, expertname } = task;
        ObjectUtil.setEmptyObjectTo(nonQualifiedCriteria, taskname);
        if (!nonQualifiedCriteria[taskname].criteria) {
          nonQualifiedCriteria[taskname] = getEmptyNonQualityDataObject(
            nonQualifiedCriteria[taskname]
          );
```

```
    }
    if (
      nonQualifiedCriteria[taskname].criteria.indexOf(criterianame) === -1
    ) {
      nonQualifiedCriteria[taskname].criteria.push(criterianame);
    }
    ArrayUtil.setEmptyArrayOnEmpty(
      nonQualifiedCriteria[taskname].experts,
      expertname
    );
    nonQualifiedCriteria[taskname].experts[expertname].push(CriteriaValue);
  });
  setQualityTasks(temp1);
  setNonQualityTasks(nonQualifiedCriteria);
  setFormattedData(preFormattedData);
};

const calculateExpertValuesFunction = (tasks) => {
  const C = Object.keys(tasks).map((task) =>
    Object.keys(tasks[task].experts).map((expert) =>
      tasks[task].experts[expert].map((criteriaValue) => criteriaValue)
    )
  );

  const expertsAmount = Object.keys(tasks).map(
    (task) =>
      Object.keys(tasks[task].experts).map((expert) => expert).length
  )[0];

  const servicesAmount = Object.keys(tasks).map((task) =>
    tasks[task].criteria.map((key) => key)
  )[0].length;

  let expertValues = [];
  C.forEach((array) => {
    const { sumRates } = calculateExpertValues(
      array,
      expertsAmount,
      servicesAmount
    );

    expertValues.push(
      Object.entries(sumRates)[Object.entries(sumRates).length - 1][1]
    );
  });

  const valuesToPush = Object.values(tasks)[0].criteria;

  formattedData.criteria = [...formattedData.criteria, ...valuesToPush];

  const keys = Object.keys(formattedData.qualityTasks);

  keys.forEach((key, i) => {
    formattedData.qualityTasks[key] = [
      ...formattedData.qualityTasks[key],
      ...expertValues[i],
    ];
  });

  setQualityTasks(formattedData);
```

```
  };

  if (!Object.keys(tasks).length && !Object.keys(nonQualityTasks).length) {
    fetchCallback();
  }

  if (ObjectUtil.isEmpty(nonQualityTasks)) {
    calculateExpertValuesFunction(nonQualityTasks);
  }
}, [nonQualityTasks]);

const getEmptyNonQualityDataObject = (object) => {
  object.criteria = [];
  object.experts = {};
  return object;
};

return (
  <>
    <Form.Group>
      <Form.Label className="text-center" as={"h4"}>
        Множина оцінок альтернатив за критеріями:
      </Form.Label>
      <Table striped bordered>
        <thead>
          <tr>
            <th>Назва роботи</th>
            {formattedData.criteria.map((key) => {
              return <th key={key}>{key}</th>;
            })}
          </tr>
        </thead>
        <tbody>
          {Object.keys(formattedData.qualityTasks).map((task, i) => {
            return (
              <tr key={i}>
                <td key={i}>{task}</td>
                {formattedData.qualityTasks[task].map((criteriaValue, i) => {
                  return <td key={i}>{criteriaValue}</td>;
                })}
              </tr>
            );
          })}
        </tbody>
      </Table>
    </Form.Group>
    <Button
      onClick={() => setShowAlgorithm(!showAlgorithm)}
      variant="outline-secondary"
      className={styles.button}
    >
      Визначити першочерговість виконання робіт
    </Button>
    {showAlgorithm && (
      <Topsis
        values={formattedData.qualityTasks}
        formattedCriteria={formattedData.criteria}
      />
    )}
  </>
```

```
  );
}
```

ValuePage.js

```
import { Button, Form, Table } from "react-bootstrap";
import { useEffect, useState } from "react";
import axios from "axios";
import { nanoid } from "nanoid";
import { ObjectUtil } from "../utils/ObjectUtil";
import { ArrayUtil } from "../utils/ArrayUtil";
import Constants from "../utils/Constants";
import TaskCriteriaCreateForm from "../components/TaskCriteriaCreateForm";
import Alert from "../components/Alert/Alert";

export default function ValuePage() {
  const [nonQualityTasks, setNonQualityTasks] = useState({});
  const [tasks, setTasks] = useState({});
  const [formattedData, setFormattedData] = useState({
    criteria: [],
    qualityTasks: {},
  });
  const [, setQualityTasks] = useState({});
  const [show, setShow] = useState(false);
  const [showAlert, setShowAlert] = useState({
    show: false,
    variant: "",
    message: "",
  });

  useEffect(() => {
    const fetchCallback = async () => {
      const preFormattedData = ObjectUtil.getClone(formattedData);
      const nonQualifiedCriteria = {};
      let temp1 = {};
      const rawNonQualifiedCriteria = [];

      const { data } = await axios.get(`${Constants.BASE_URI}`);
      const qualifiedCriteria = data.filter((item) => {
        if (!item.isqualitycriteria) {
          rawNonQualifiedCriteria.push(item);
        }
        return item.isqualitycriteria;
      });
      qualifiedCriteria.forEach((elem) => {
        if (preFormattedData.criteria.indexOf(elem.criterianame) === -1) {
          preFormattedData.criteria.push(elem.criterianame);
        }
      });
      setTasks(data);

      qualifiedCriteria.forEach((task) => {
        const { taskname, CriteriaValue } = task;
        ArrayUtil.setEmptyArrayOnEmpty(preFormattedData.qualityTasks, taskname);
        preFormattedData.qualityTasks[taskname].push(CriteriaValue);
      });

      rawNonQualifiedCriteria.forEach((task) => {
        const { taskname, criterianame, CriteriaValue, expertname } = task;
        ObjectUtil.setEmptyObjectTo(nonQualifiedCriteria, taskname);
        if (!nonQualifiedCriteria[taskname].criteria) {
```

```
        nonQualifiedCriteria[taskname] = getEmptyNonQualityDataObject(
          nonQualifiedCriteria[taskname]
        );
      }
      if (
        nonQualifiedCriteria[taskname].criteria.indexOf(criterianame) === -1
      ) {
        nonQualifiedCriteria[taskname].criteria.push(criterianame);
      }
      ArrayUtil.setEmptyArrayOnEmpty(
        nonQualifiedCriteria[taskname].experts,
        expertname
      );
      nonQualifiedCriteria[taskname].experts[expertname].push(CriteriaValue);
    });
    setQualityTasks(temp1);
    setNonQualityTasks(nonQualifiedCriteria);
    setFormattedData(preFormattedData);
  };
  if (!Object.keys(tasks).length) {
    fetchCallback();
  }
}, []);

const getEmptyNonQualityDataObject = (object) => {
  object.criteria = [];
  object.experts = {};
  return object;
};

const onOpenAddModal = () => {
  setShow(true);
};

const onHide = () => {
  setShow(false);
};

const onSubmit = async (qualityCriteria, nonQualityCriteria, taskId) => {
  const array = [];

  Object.entries(qualityCriteria).forEach(([criteriaId, criteriaValue]) =>
    array.push({
      Task_ID: Number(taskId.id),
      Criteria_ID: Number(criteriaId),
      Expert_ID: "null",
      CriteriaValue: Number(criteriaValue),
    })
  );

  Object.entries(nonQualityCriteria).forEach(([expertId, obj]) => {
    Object.entries(obj).forEach(([criteriaId, value]) =>
      array.push({
        Task_ID: Number(taskId.id),
        Criteria_ID: Number(criteriaId),
        Expert_ID: Number(expertId),
        CriteriaValue: Number(value),
      })
    );
  });
```

```
  try {
    await axios.post(`${Constants.BASE_URI}addTaskCriteria`, { array });
    setShow(false);
    setShowAlert({
      show: true,
      variant: "success",
      message: "Успішно додано оцінки критеріїв по всім роботам!",
    });
    setTimeout(() => window.location.reload(), 500);
  } catch (e) {
    setShowAlert({
      show: true,
      variant: "danger",
      message: "Щось пішло не так. Спробуйте пізніше!",
    });
    throw e;
  }
};

const onCloseAlert = () => {
  setShowAlert(false);
};

return (
  <>
    <Button onClick={onOpenAddModal} className="mb-3">
      Хочете поповнити дані?
    </Button>
    <TaskCriteriaCreateForm
      showModal={show}
      onHide={onHide}
      onSubmit={onSubmit}
      allTasks={Object.keys(formattedData.qualityTasks)}
    />
    <Form.Group>
      <Form.Label className="text-center" as={"h4"}>
        Роботи з об'єктивними оцінками:
      </Form.Label>
      <Table striped bordered>
        <thead>
          <tr>
            <th>Назва роботи</th>
            {formattedData.criteria.map((key) => {
              return <th key={key}>{key}</th>;
            })}
          </tr>
        </thead>
        <tbody>
          {Object.keys(formattedData.qualityTasks).map((task) => {
            return (
              <tr>
                <td>{task}</td>
                {formattedData.qualityTasks[task].map((criteriaValue) => {
                  return <td>{criteriaValue}</td>;
                })}
              </tr>
            );
          })}
        </tbody>
```

```
        </Table>
      </Form.Group>

      <Form.Group>
        <Form.Label className="text-center" as={"h4"}>
          Роботи з суб'єктивними оцінками:
        </Form.Label>
        {Object.keys(nonQualityTasks).map((task) => (
          <>
            <h4 className="mb-2">Робота "{task}"</h4>
            <Table striped bordered>
              <thead>
                <tr>
                  <th>Експерт</th>
                  {nonQualityTasks[task].criteria.map((key) => {
                    return <th key={key}>{key}</th>;
                  })}
                </tr>
              </thead>
              <tbody>
                {Object.keys(nonQualityTasks[task].experts).map((expert) => (
                  <tr>
                    <td>{expert}</td>
                    {nonQualityTasks[task].experts[expert].map(
                      (criteriaValue) => (
                        <td key={nanoid()}>{criteriaValue}</td>
                      )
                    )}
                  </tr>
                ))}
              </tbody>
            </Table>
          </>
        ))}
      </Form.Group>
      <Alert
        show={showAlert.show}
        variant={showAlert.variant}
        message={showAlert.message}
        onClose={onCloseAlert}
        setShow={setShowAlert}
      />
    </>
  );
}
```

CalculateExpertValues.js

```
import { parseFloatToFixed, getIteration } from "../functions.js";

// import { C, expertsAmount, servicesAmount } from "./constants.js";

export default function calculateExpertValues(
  C,
  expertsAmount,
  servicesAmount
) {
  const sumRates = {};
  const allCoefsMatrixNorm = {};

  // console.log("Матриця оцінок альтернатив експертами:");
```

```
// console.table(C);
//
const l = 0.001;
// console.log(
//   "Порогова величина сумарної зміни вагових коефіцієнтів експертів: ",
//   l
// );

let sumC = [];

//розрахунок суми по рядку матриці оцінок експертів
for (let i = 0; i < expertsAmount; i += 1) {
  let temp = 0;
  for (let j = 0; j < servicesAmount; j += 1) {
    temp += C[i][j];
  }
  sumC.push(temp);
}

// console.log("Сума оцінок експертів:");
// console.table(sumC);

//зважена матриця оцінок експертів
const weightedC = new Array(expertsAmount)
  .fill(0)
  .map(() => new Array(servicesAmount).fill(0));

for (let i = 0; i < expertsAmount; i += 1) {
  for (let j = 0; j < servicesAmount; j += 1) {
    weightedC[i][j] = parseFloatToFixed(C[i][j] / sumC[i]);
  }
}
//
// console.log("Зважена матриця оцінок альтернатив експертами:");
// console.table(weightedC);

//матриця компетенцій експертів
let coefsMatrix = new Array(expertsAmount)
  .fill(0)
  .map(() =>
    new Array(servicesAmount)
      .fill(0)
      .map(() => parseFloatToFixed(1 / expertsAmount))
  );

// console.log("Компетенції експертів: ");
// console.table(coefsMatrix);

let G = 9999;

const S = new Array(expertsAmount)
  .fill(0)
  .map(() => new Array(servicesAmount).fill(0));

let iteration = 1;
while (G > l) {
  // for (let step = 0; step < 2; step += 1) {

  // console.log(`Iteration - ${iteration}`);
```

```
//матриця відповідей експертів з урахуванням їх компетенції
for (let i = 0; i < expertsAmount; i += 1) {
  for (let j = 0; j < servicesAmount; j += 1) {
    S[i][j] = parseFloatToFixed(weightedC[i][j] * coefsMatrix[i][j]);
  }
}

// console.log("Відповіді експертів з урахуванням їх компетенцій: ");
// console.table(S);

let P = [];
//вектор сумарних оцінок об'єктів
for (let i = 0; i < servicesAmount; i++) {
  let temp = 0;
  for (let j = 0; j < expertsAmount; j++) {
    temp += S[j][i];
  }
  P.push(parseFloatToFixed(temp));
}

// console.log("Сумарні оцінки об'єктів:");
// console.table(P);

let U = [];

for (let i = 0; i < servicesAmount; i++) {
  let temp = 0;
  for (let j = 0; j < expertsAmount; j++) {
    temp += S[j][i];
  }
  U[i] = parseFloatToFixed(temp / expertsAmount);
}

// console.log(
//   "Середньоарифметичне відповідей експертів із врахуванням їх компетентності:"
// );
// console.table(U);

sumRates[`Iteration №${iteration}`] = U;

const Z = new Array(expertsAmount)
  .fill(0)
  .map(() => new Array(servicesAmount).fill(0));

for (let i = 0; i < expertsAmount; i += 1) {
  for (let j = 0; j < servicesAmount; j += 1) {
    Z[i][j] = parseFloatToFixed(Math.abs(S[i][j] - U[j]));
  }
}

// console.log("Матриця відхилень відповідей від середньозважених: ");
// console.table(Z);

let sumZ = [];

//розрахунок суми по рядку модулів відхилень
for (let i = 0; i < expertsAmount; i += 1) {
  let temp = 0;
  for (let j = 0; j < servicesAmount; j += 1) {
    temp += Z[i][j];
```

```
  }
  sumZ.push(temp);
}
// console.log("Сума модулів відхилень:");
// console.table(sumZ);

const sumZsum = sumZ.reduce((acc, curr) => acc + curr, 0);
const sumZnormalized = sumZ.map((elem) =>
  parseFloatToFixed(elem / sumZsum)
);

// console.log("Сума модулів відхилень (нормалізована):");
// console.table(sumZnormalized);

const coefsMatrixNew = new Array(expertsAmount)
  .fill(0)
  .map(() => new Array(servicesAmount).fill(0));

for (let i = 0; i < expertsAmount; i += 1) {
  for (let j = 0; j < servicesAmount; j += 1) {
    coefsMatrixNew[i][j] = parseFloatToFixed(
      coefsMatrix[i][j] / sumZnormalized[i]
    );
  }
}
// console.log("Нові значення компетенцій:");
// console.table(coefsMatrixNew);

let sumCoefsMatrixNew = [];

for (let i = 0; i < servicesAmount; i++) {
  let temp = 0;
  for (let j = 0; j < expertsAmount; j++) {
    temp += coefsMatrixNew[j][i];
  }
  sumCoefsMatrixNew.push(parseFloatToFixed(temp));
}

// console.log("Сума значень компетенцій для кожного експерта:");
// console.table(sumCoefsMatrixNew);

const coefsMatrixNewNorm = new Array(expertsAmount)
  .fill(0)
  .map(() => new Array(servicesAmount).fill(0));

for (let i = 0; i < expertsAmount; i += 1) {
  for (let j = 0; j < servicesAmount; j += 1) {
    coefsMatrixNewNorm[i][j] = parseFloatToFixed(
      coefsMatrixNew[i][j] / sumCoefsMatrixNew[j]
    );
  }
}
// console.log("Нормовані нові значення компетенцій:");
// console.table(coefsMatrixNewNorm);

coefsMatrix = JSON.parse(JSON.stringify(coefsMatrixNewNorm));

let sumCoefsMatrixNewNorm = [];

for (let i = 0; i < servicesAmount; i++) {
```

```
      let temp = 0;
      for (let j = 0; j < expertsAmount; j++) {
        temp += coefsMatrixNewNorm[j][i];
      }
      sumCoefsMatrixNewNorm.push(parseFloatToFixed(temp));
    }

    allCoefsMatrixNorm[`Iteration №${iteration}`] = coefsMatrixNewNorm;

    if (iteration !== 1) {
      const currSumRate = sumRates[getIteration(iteration)];
      const prevSumRate = sumRates[getIteration(iteration - 1)];

      G = currSumRate
        .map((elem, i) => Math.abs(elem - prevSumRate[i]))
        .reduce((curr, prev) => curr + prev, 0);
    }
    iteration += 1;
  }

  return { sumRates, allCoefsMatrixNorm };

  // export { sumRates, allCoefsMatrixNorm };
}
```

ArrayUtil.js

```
export class ArrayUtil {
  static setEmptyArrayOnEmpty(object, field) {
    if (!object[field]) {
      object[field] = [];
    }
  }
}
```

CalculateTopsis.js

```
import { parseFloatToFixed } from "./functions";

export default function calculateTopsis(
  alternatives,
  weights,
  tasks,
  criteria
) {
  const dtoOut = {};

  const normAlternatives = new Array(alternatives.length)
    .fill(0)
    .map(() => new Array(alternatives[0].length).fill(0));

  let sumOfColumns = alternatives.reduce(function (r, a) {
    a.forEach(function (b, i) {
      r[i] = (r[i] || 0) + b;
    });
    return r;
  }, []);

  for (let i = 0; i < alternatives.length; i += 1) {
    for (let j = 0; j < alternatives[0].length; j += 1) {
      normAlternatives[i][j] = parseFloatToFixed(
        alternatives[i][j] / sumOfColumns[j]
```

```
    );
  }
}

dtoOut.normalizedAlternatives = normAlternatives;

const normWeightAlternatives = new Array(alternatives.length)
  .fill(0)
  .map(() => new Array(alternatives[0].length).fill(0));

for (let i = 0; i < normAlternatives.length; i += 1) {
  for (let j = 0; j < normAlternatives[0].length; j += 1) {
    normWeightAlternatives[i][j] = parseFloatToFixed(
      normAlternatives[i][j] * weights[j]
    );
  }
}

dtoOut.normalizedWeighedAlternatives = normWeightAlternatives;

const maxMinMatrix = [];
for (let i = 0; i < normWeightAlternatives[0].length; i += 1) {
  let temp = [];
  for (let j = 0; j < normWeightAlternatives.length; j += 1) {
    temp.push(normWeightAlternatives[j][i]);
  }

  if (criteria[i] === "max") {
    maxMinMatrix.push({ min: Math.min(...temp), max: Math.max(...temp) });
  } else {
    maxMinMatrix.push({ min: Math.max(...temp), max: Math.min(...temp) });
  }
}

const PIS = [];
for (let i = 0; i < normWeightAlternatives.length; i += 1) {
  let temp = 0;
  for (let j = 0; j < normWeightAlternatives[0].length; j += 1) {
    temp += Math.pow(normWeightAlternatives[i][j] - maxMinMatrix[j].max, 2);
  }
  PIS.push(parseFloatToFixed(Math.sqrt(temp), 5));
}

dtoOut.PIS = PIS;

const NIS = [];
for (let i = 0; i < normWeightAlternatives.length; i += 1) {
  let temp = 0;
  for (let j = 0; j < normWeightAlternatives[0].length; j += 1) {
    temp += Math.pow(normWeightAlternatives[i][j] - maxMinMatrix[j].min, 2);
  }
  NIS.push(parseFloatToFixed(Math.sqrt(temp), 5));
}

dtoOut.NIS = NIS;

let distanceC = [];
for (let i = 0; i < PIS.length; i += 1) {
  distanceC.push(parseFloatToFixed(NIS[i] / (PIS[i] + NIS[i]), 5));
}
```

```
  distanceC = distanceC.reduce((acc, curr, i) => {
    acc[`task-${i}`] = curr;
    return acc;
  }, {});

  const sortedDistances = Object.entries(distanceC).sort((a, b) => b[1] - a[1]);

  const order = Object.fromEntries(sortedDistances);
  dtoOut.distances = order;

  const orderOfImplementation = {};
  Object.keys(order)
    .map((key) => ({ [key]: tasks[key] }))
    .forEach((elem) => {
      const objectKey = Object.keys(elem)[0];
      orderOfImplementation[objectKey] = elem[objectKey];
    });

  dtoOut.order = orderOfImplementation;

  return dtoOut;
}
```

Constants.js

```
export default {
  BASE_URI: "http://localhost:3000/api/",
};
```

functions.js

```
//функція нормування числа
export const parseFloatToFixed = (value) => parseFloat(value.toFixed(5));

//функція генерації випадкового числа
export const getExpertsRate = (minRate, maxRate) =>
  Math.floor(Math.random() * (maxRate - minRate) + minRate);

//функція отримання поточної ітерації
export const getIteration = (iteration) => `Iteration №${iteration}`;

//транспонування матриці
export const transformDataToMatrix = (data) => {
  const matrix = Object.keys(data).map((iteration) => data[iteration]);

  return matrix[0].map((_, colIndex) => matrix.map((row) => row[colIndex]));
};

//генерація випадкових кольорів
export const randomColor = () => {
  return Math.floor(Math.random() * 16777215).toString(16);
};
```

ObjectUtil.js

```
export class ObjectUtil {
  static setEmptyObjectTo(object, field) {
    if (!object[field]) {
      object[field] = {};
    }
  }
```

```
  static getClone(object) {
    return JSON.parse(JSON.stringify(object));
  }

  static isEmpty(object) {
    return Boolean(Object.keys(object).length);
  }
}
```

App.js

```
import React from "react";
import { Routes, Route } from "react-router-dom";
import Header from "./components/Header";
import MainPage from "./pages/MainPage";
import ExpertPage from "./pages/ExpertPage";
import CriteriaPage from "./pages/CriteriaPage";
import TaskPage from "./pages/TaskPage";
import Footer from "./components/Footer";
import ValuePage from "./pages/ValuePage";
import TopsisPage from "./pages/TopsisPage";

function App() {
  return (
    <>
      <Header />
      <Routes>
        <Route exact="true" path="/" element={<MainPage />} />
        <Route exact="true" path="/experts" element={<ExpertPage />} />
        <Route exact="true" path="/criteria" element={<CriteriaPage />} />
        <Route exact="true" path="/tasks" element={<TaskPage />} />
        <Route exact="true" path="/values" element={<ValuePage />} />
        <Route exact="true" path="/topsis" element={<TopsisPage />} />
      </Routes>
      <Footer />
    </>
  );
}

export default App;
```

www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('server:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
```

```
/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}
```

```
/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

PgClient.js

```
const { Pool } = require("pg");

class PgClient {
  constructor() {
    const credentials = {
      user: "postgres",
      host: "localhost",
      database: "postgres",
      password: "DBPASS",
      port: 5432,
    };
    this.pool = new Pool(credentials);
  }

  async getAllTasks() {
    const { rows } = await this.pool.query(
      `SELECT "Task_Criteria".id,
          "Task"."Name" as TaskName,
          "Criteria"."Name" as CriteriaName,
          "Criteria"."IsQualityCriteria" as IsQualityCriteria,
          "Experts"."Name" as ExpertName,
          "CriteriaValue"
          FROM public."Task_Criteria"
          JOIN public."Task" On "Task_ID" = public."Task".id
          JOIN public."Criteria" On "Criteria_ID" = public."Criteria".id
          LEFT JOIN public."Experts" On "Expert_ID" = public."Experts".id`
    );
    return rows;
  }

  async addTask(name) {
    await this.pool.query(
      `INSERT INTO public."Task"("Name")
           VALUES ('${name}')`
    );
  }

  async updateTask(id, name) {
    await this.pool.query(
      `UPDATE public."Task"
                  SET "Name"='${name}'
                  WHERE id='${id}'`
    );
  }

  async deleteTask(id) {
```

```
  await this.pool.query(
    `DELETE FROM public."Task_Criteria"
       WHERE "Task_ID"='${id}'`
  );

  await this.pool.query(
    `DELETE FROM public."Task"
                WHERE id='${id}'`
  );
}

async getTasks() {
  const { rows } = await this.pool.query(
    `SELECT * FROM public."Task"
    ORDER BY id ASC`
  );

  return rows;
}

async getExperts() {
  const { rows } = await this.pool.query(
    `SELECT * FROM public."Experts"
    ORDER BY id ASC`
  );

  return rows;
}

async addExpert(name) {
  await this.pool.query(
    `INSERT INTO public."Experts"("Name")
         VALUES ('${name}')`
  );
}

async updateExpert(id, name) {
  await this.pool.query(
    `UPDATE public."Experts"
                SET "Name"='${name}'
                WHERE id='${id}'`
  );
}

async deleteExpert(id) {
  await this.pool.query(
    `DELETE FROM public."Task_Criteria"
       WHERE "Expert_ID"='${id}'`
  );

  await this.pool.query(
    `DELETE FROM public."Experts"
                WHERE id='${id}'`
  );
}

async getCriteria() {
  const { rows } = await this.pool.query(
    `SELECT * FROM public."Criteria"
       ORDER BY id ASC`
```

```
    );

    return rows;
  }

  async addCriteria(name, isQuality) {
    await this.pool.query(
      `INSERT INTO public."Criteria"
        ("Name","IsQualityCriteria")
        VALUES ('${name}','${isQuality}')`
    );
  }

  async updateCriteria(id, name, isQuality) {
    await this.pool.query(
      `UPDATE public."Criteria"
        SET "Name"='${name}', "IsQualityCriteria"='${isQuality}'
        WHERE id='${id}'`
    );
  }

  async deleteCriteria(id) {
    await this.pool.query(
      `DELETE FROM public."Task_Criteria"
        WHERE "Criteria_ID"='${id}'`
    );

    await this.pool.query(
      `DELETE FROM public."Criteria"
                WHERE id='${id}'`
    );
  }

  async addTaskCriteria(array) {
    await this.pool.query(`INSERT INTO public."Task_Criteria"(
       "Task_ID", "Criteria_ID", "Expert_ID", "CriteriaValue")
       VALUES ${array
         .map((arr) => {
           return `(${Object.values(arr)})`;
         })
         .join(",")};`);
  }
}

module.exports = PgClient;
```

api.js

```
const PgClient = require("../database/PgClient");
const express = require("express");
const router = express.Router();
const pgController = new PgClient();

router.get("/", async (req, res) => {
  const tasks = await pgController.getAllTasks();
  res.send(JSON.stringify(tasks));
});

router.post("/addTask", async (req, res) => {
  await pgController.addTask(req.body.name);
  res.sendStatus(200);
```

```
});

router.post("/deleteTask", async (req, res) => {
  await pgController.deleteTask(req.body.id);
  res.sendStatus(200);
});

router.post("/updateTask", async (req, res) => {
  await pgController.updateTask(req.body.id, req.body.name);
  res.sendStatus(200);
});

router.get("/getTasks", async (req, res) => {
  const tasks = await pgController.getTasks();
  res.send(JSON.stringify(tasks));
});

router.get("/getExperts", async (req, res) => {
  const experts = await pgController.getExperts();
  res.send(JSON.stringify(experts));
});

router.post("/addExpert", async (req, res) => {
  await pgController.addExpert(req.body.name);
  res.sendStatus(200);
});

router.post("/deleteExpert", async (req, res) => {
  await pgController.deleteExpert(req.body.id);
  res.sendStatus(200);
});

router.post("/updateExpert", async (req, res) => {
  await pgController.updateExpert(req.body.id, req.body.name);
  res.sendStatus(200);
});

router.get("/getCriteria", async (req, res) => {
  const experts = await pgController.getCriteria();
  res.send(JSON.stringify(experts));
});

router.post("/addCriteria", async (req, res) => {
  await pgController.addCriteria(req.body.name, req.body.isQuality);
  res.sendStatus(200);
});

router.post("/deleteCriteria", async (req, res) => {
  await pgController.deleteCriteria(req.body.id);
  res.sendStatus(200);
});

router.post("/updateCriteria", async (req, res) => {
  await pgController.updateCriteria(
    req.body.id,
    req.body.name,
    req.body.isQuality
  );
  res.sendStatus(200);
});
```

```
router.post("/addTaskCriteria", async (req, res) => {
  await pgController.addTaskCriteria(req.body.array);
  res.sendStatus(200);
});

module.exports = router;
```

index.js

```
const express = require('express');
const router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

module.exports = router;
```

app.js

```
const createError = require('http-errors');
const express = require('express');
const path = require('path');
const cookieParser = require('cookie-parser');
const logger = require('morgan');
const cors = require('cors');

const indexRouter = require('./routes/index');
const apiRouter = require('./routes/api');

const app = express();

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
app.use(cors({
  origin: '*',
}));

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/api', apiRouter);

app.use(function(req, res, next) {
  next(createError(404));
});

app.use(function(err, req, res, next) {
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```