



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

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

\_\_\_\_\_ Наталія АУШЕВА

« \_\_\_\_ » \_\_\_\_\_ 2026 р.

**З А В Д А Н Н Я**

**на дипломну роботу студенту**

Клещенко Юрію Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи «Вебсистема підтримки колективного прийняття рішень на основі методів експертного оцінювання».

керівник роботи Рудик Володимир Іванович, асистент каф. цифрових технологій в енергетиці

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» травня 2026 р. № 1992-с

2. Термін подання роботи студентом 08.06.2026р.

3. Вихідні дані до роботи: мови програмування Python 3.11, JavaScript; фреймворки та бібліотеки FastAPI, SQLAlchemy 2.0, Pydantic, ReportLab, React 18, Vite, Recharts, React Router; реляційна СКБД PostgreSQL 15; проксі-сервер NGINX; Docker та Docker Compose; підсистема автентифікації JWT з підписом RS256, bcrypt; засоби розробки Git, Visual Studio Code, Postman, pytest.

4. Перелік питань, які потрібно розробити: 1) проаналізувати методи колективного прийняття рішень та існуючі підходи до експертного оцінювання (зважений середній бал, метод Борда, метод Кондорсе, метод Делфі, метод парних порівнянь АНР); 2) обґрунтувати вибір архітектурних рішень та технологічного стеку; 3) спроєктувати модель даних та архітектуру вебсистеми з розмежуванням відповідальності між модулями, розробити серверну частину на основі Flask з реалізацією п'яти методів агрегування експертних оцінок; 4) виконати тестування системи на демонстраційні сценаріях та проаналізувати результати.

5. Орієнтовний перелік ілюстративного матеріалу: діаграма прецедентів (use case); діаграма діяльності процесу проведення сесії оцінювання; концептуальна та логічна моделі даних (ER-діаграма); схема модульної архітектури вебсистеми; блок-схема алгоритму обробки експертних оцінок; скріншоти інтерфейсу користувача (панель

організатора, форма введення оцінок, сторінка результатів); фрагмент згенерованого PDF-звіту; графік динаміки коефіцієнта конкордації Кендалла W між раундами.

6. Дата видачі завдання 01.09.2025р.

### КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання дипломної роботи                                                                                                                     | Термін виконання етапів дипломної роботи | Примітка |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|----------|
| 1     | Затвердження теми роботи                                                                                                                                    | 01.09.2025                               | Виконано |
| 2     | Аналіз предметної області, огляд та вибір методів оцінювання: зважений середній бал, метод Борда, метод Кондорсе, метод Делфі, метод парних порівнянь (АНР) | 23.09–27.10.25                           | Виконано |
| 3     | Обґрунтування технологічного стеку та проєктування модульної архітектури вебсистеми                                                                         | 28.10–17.11.25                           | Виконано |
| 4     | Розробка моделі даних, ER-діаграми та схеми взаємодії компонентів системи                                                                                   | 18.11–01.12.25                           | Виконано |
| 5     | Реалізація серверної частини: маршрути, підсистема автентифікації, реалізація п'яти методів агрегування експертних оцінок та модуля метрик узгодженості     | 02.12.25–16.02.26                        | Виконано |
| 6     | Розробка вебінтерфейсу користувача                                                                                                                          | 17.02–27.04.26                           |          |
| 7     | Оформлення пояснювальної записки                                                                                                                            | 28.04–25.05.26                           | Виконано |
| 8     | Захист програмного продукту                                                                                                                                 | 14.05.26                                 | Виконано |
| 9     | Передзахист                                                                                                                                                 | 04.06.26                                 | Виконано |
| 10    | Захист                                                                                                                                                      | 18.06.26                                 | Виконано |

Студент

\_\_\_\_\_  
(підпис)

Керівник роботи

\_\_\_\_\_  
(підпис)

Юрій КЛЕЩЕНКО  
(ім'я, ПРІЗВИЩЕ)

Володимир РУДИК  
(ім'я, ПРІЗВИЩЕ)

# АНОТАЦІЯ

Дипломна робота виконана на 48 сторінках, містить 7 ілюстрацій, 6 таблиць, 1 додаток, 21 джерело у переліку посилань.

Метою роботи є розробка веб-системи підтримки колективного прийняття рішень на основі методів експертного оцінювання, яка забезпечує формування сесій оцінювання, керування експертною панеллю, збір індивідуальних оцінок, обчислення результатів декількома методами, аналіз узгодженості думок експертів та візуалізацію підсумків.

Основний зміст роботи включає аналіз предметної області та існуючих підходів до колективного прийняття рішень, огляд і систематизацію методів експертного оцінювання, проектування архітектури системи та моделі бази даних, програмну реалізацію серверної частини та веб-інтерфейсу, а також тестування й аналіз отриманих результатів.

Методи та засоби: методи експертного оцінювання – зважений середній бал, метод Борда, метод Кондорсе, метод Делфі, метод парних порівнянь АНР; метрики узгодженості – коефіцієнт конкордації Кендалла  $W$ , коефіцієнт варіації, ентропія Шеннона; мова програмування Python, фреймворк Flask, СКБД SQLite, бібліотека ReportLab, шаблонізатор Jinja2.

Результат: програмна система – веб-застосунок підтримки колективного прийняття рішень, що реалізує п'ять методів агрегування експертних оцінок, порівняльний аналіз їх результатів, перевірку статистичної значущості узгодженості та генерацію PDF-звіту.

Ключові слова: КОЛЕКТИВНЕ ПРИЙНЯТТЯ РІШЕНЬ, МЕТОДИ ЕКСПЕРТНОГО ОЦІНЮВАННЯ, ВЕБСИСТЕМА, ЗВАЖЕНИЙ СЕРЕДНІЙ БАЛ, МЕТОД БОРДА, МЕТОД КОНДОРСЕ, МЕТОД ДЕЛФІ, АНР, УЗГОДЖЕНІСТЬ, KENDALL  $W$ , МЕТОД БОРДА, МЕТОД КОНДОРСЕ, МЕТОД ДЕЛФІ, АНР, УЗГОДЖЕНІСТЬ, KENDALL  $W$ .

# ABSTRACT

The thesis consists of 48 pages and includes 7 figures, 6 tables, 1 appendix, and 21 references.

The aim of the thesis is to develop a web-based system for supporting collective decision-making based on expert evaluation methods, which facilitates the creation of evaluation sessions, management of the expert panel, collection of individual evaluations, calculation of results using multiple methods, analysis of expert consensus, and visualization of the results.

The main content of the work includes an analysis of the subject area and existing approaches to collective decision-making, a review and systematization of expert evaluation methods, the design of the system architecture and database model, the software implementation of the server-side and web interface, as well as testing and analysis of the results obtained.

Methods and tools: expert evaluation methods—weighted average score, Borda method, Condorcet method, Delphi method, AHP pairwise comparison method; consistency metrics—Kendall's W concordance coefficient, coefficient of variation, Shannon entropy; Python programming language, Flask framework, SQLite DBMS, ReportLab library, Jinja2 templating engine.

Result: a software system—a web application for supporting collective decision-making that implements five methods for aggregating expert assessments, comparative analysis of their results, verification of the statistical significance of consistency, and generation of a PDF report.

Keywords: COLLECTIVE DECISION-MAKING, EXPERT EVALUATION METHODS, WEB SYSTEM, WEIGHTED AVERAGE SCORE, BORD METHOD, CONDORSE METHOD, DELPHI METHOD, AHP, CONSISTENCY, KENDALL W, BORD METHOD, CONDORSE METHOD, DELPHI METHOD, AHP, CONSISTENCY, KENDALL W.

# ЗМІСТ

|                                                                 |    |
|-----------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ ..... | 7  |
| ВСТУП.....                                                      | 9  |
| 1 АНАЛІЗ ЗАДАЧІ РОЗРОБКИ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ .....         | 11 |
| 1.1 Поняття колективного прийняття рішень.....                  | 11 |
| 1.2 Аналіз існуючих систем підтримки прийняття рішень .....     | 13 |
| 1.3 Постановка задачі.....                                      | 16 |
| 2 МЕТОДИ ЕКСПЕРТНОГО ОЦІНЮВАННЯ.....                            | 18 |
| 2.1 Класифікація методів експертного оцінювання .....           | 18 |
| 2.2 Зважений середній бал .....                                 | 20 |
| 2.3 Метод Борда .....                                           | 20 |
| 2.4 Метод Кондорсе .....                                        | 21 |
| 2.5 Метод Делфі .....                                           | 22 |
| 2.6 Метод парних порівнянь (АНР).....                           | 22 |
| 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ.....                             | 24 |
| 3.1 Вибір технологій .....                                      | 24 |
| 3.2 Функціональні вимоги.....                                   | 25 |
| 3.3 Архітектура системи .....                                   | 30 |
| 3.4 Концептуальна схема бази даних .....                        | 33 |
| 3.5 Реалізація методів експертного оцінювання .....             | 36 |
| 4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ.....                            | 39 |
| 4.1 Системні вимоги.....                                        | 39 |
| 4.2 Сценарій роботи користувачів з системою.....                | 39 |

|                                                                          |    |
|--------------------------------------------------------------------------|----|
| 4.3 Тестування системи (методика, валідація і аналіз ефективності) ..... | 41 |
| ВИСНОВКИ.....                                                            | 47 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....                                         | 49 |
| ДОДАТОК А.....                                                           | 51 |

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

АНР (Analytic Hierarchy Process) – метод аналізу ієрархій (метод парних порівнянь Сааті)

API (Application Programming Interface) – програмний інтерфейс взаємодії між компонентами системи

CRUD (Create, Read, Update, Delete) – базові операції роботи з даними: створення, читання, оновлення, видалення

CSS (Cascading Style Sheets) – каскадні таблиці стилів для оформлення вебсторінок

CV (Coefficient of Variation) – коефіцієнт варіації, міра відносного розкиду оцінок

DSS (Decision Support System) – система підтримки прийняття рішень

DRY (Don't Repeat Yourself) – принцип проєктування програмного забезпечення, що забороняє дублювання логіки

HTML (HyperText Markup Language) – мова розмітки гіпертексту для створення вебсторінок

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту, основа обміну даними у вебі

IQR (Interquartile Range) – міжквартильний розмах, різниця між третім і першим квартилями розподілу

ISO/IEC – міжнародна організація зі стандартизації та міжнародна електротехнічна комісія

JWT (JSON Web Token) – стандарт токенів автентифікації

ORM (Object-Relational Mapping) – технологія відображення об'єктів на реляційну базу даних

PDF (Portable Document Format) – формат документів з фіксованим

відображенням, незалежним від платформи

PRAGMA – директива SQLite для налаштування параметрів бази даних

REST (Representational State Transfer) – архітектурний стиль побудови вебслужб

СКБД – система керування базами даних

SQL (Structured Query Language) – мова структурованих запитів для роботи з реляційними базами даних

SQLite – вбудована реляційна система керування базами даних, що зберігає дані у єдиному файлі

UML (Unified Modeling Language) – уніфікована мова моделювання програмних систем

URL (Uniform Resource Locator) – уніфікований покажчик ресурсу, адреса вебсторінки або ресурсу

W – коефіцієнт конкордації Кендалла, міра узгодженості думок групи експертів (від 0 до 1)

АНП – аналітичний мережевий процес

БД – база даних

ВНЗ – вищий навчальний заклад

КППР – колективне прийняття рішень

МЕО – методи експертного оцінювання

ПЗ – програмне забезпечення

СППР – система підтримки прийняття рішень

$\chi^2$  (хі-квадрат) – статистичний критерій для перевірки значущості коефіцієнта конкордації

$\sigma$  (сигма) – стандартне відхилення вибірки оцінок

$\Sigma$  (велика сигма) – знак суми в математичних формулах

$\mu$  (мю) – середнє арифметичне значення вибірки

$\rho$  (ро) – коефіцієнт кореляції Спірмена між ранжуваннями

$\tau$  (тау) – коефіцієнт кореляції Кендалла між двома ранжуваннями

## ВСТУП

Сучасні управлінські, організаційні та технологічні задачі дедалі частіше мають колективний характер. На практиці це означає, що рішення не ухвалюється однією особою, а формується на основі узгодження позицій групи фахівців, кожен з яких має власний досвід, рівень компетентності та систему пріоритетів. Саме тому системи підтримки колективного прийняття рішень залишаються важливим напрямом розвитку прикладної інформатики та інженерії програмного забезпечення. Застосування методів експертного оцінювання дає змогу перевести суб'єктивні судження в формалізовану модель, порівняти альтернативи, виявити рівень згоди між учасниками та обґрунтувати вибір рекомендованого варіанта [1 -3]. Актуальність цієї проблематики безпосередньо підтверджується у наданому технічному завданні, де визначено потребу у вебсистемі, що підтримує множину методів експертного оцінювання, багатораундові процедури та порівняння отриманих результатів.

На відміну від звичайних форм опитування або простого голосування, експертне оцінювання передбачає не лише фіксацію відповідей, а й інтелектуальну обробку вхідних даних. Для коректної підтримки такого процесу вебсистема повинна забезпечувати керування сесіями, опис альтернатив, облік коефіцієнтів компетентності експертів, збір оцінок у різних форматах, виконання математичних розрахунків за різними підходами, а також візуальне подання результатів для організатора. Саме така постановка задачі була закладена у вихідному описі проєкту, де окремо наголошено на необхідності реалізувати не менше п'яти методів з різних класів, модуль аналізу узгодженості та механізм генерації звітів.

Для досягнення поставленої мети в роботі розв'язуються такі задачі:

- 1) проаналізувати методи колективного прийняття рішень та існуючі підходи до експертного оцінювання – зважений середній бал, метод Борда, метод Кондорсе, метод Делфі та метод парних порівнянь АНР;

- 2) обґрунтувати вибір технологічного стеку та архітектурних рішень для розробки вебсистеми підтримки колективного прийняття рішень;
- 3) спроектувати модульну архітектуру вебсистеми, реляційну модель бази даних та UML-діаграми, що описують функціональні сценарії та взаємодію компонентів;
- 4) розробити серверну частину вебсистеми на основі фреймворку Flask із реалізацією п'яти методів агрегування експертних оцінок та модуля аналізу узгодженості думок;
- 5) реалізувати вебінтерфейс користувача та модуль генерації PDF-звітів за результатами сесії оцінювання;
- 6) виконати тестування системи на демонстраційному прикладному сценарії та проаналізувати коректність обчислювальних результатів і ефективність реалізованого підходу.

У роботі використано методи системного аналізу, порівняння, формалізації, моделювання, проектування баз даних, алгоритмічної розробки, статистичного оцінювання узгодженості та експериментального тестування [4-6]. Практичне значення роботи полягає у створенні прототипу вебзастосунку, який може бути використаний як основа для автоматизації конкурсних відборів, внутрішньо-організаційних експертних рад, проєктних комісій, навчальних кейсів та інших групових процедур вибору альтернатив.

Структура роботи відповідає затвердженому змісту, що містить чотири розділи: аналіз предметної області, проектування системи, програмну реалізацію та тестування з аналізом результатів. У першому розділі розглянуто теоретичні засади колективного прийняття рішень та методи експертного оцінювання. У другому розділі подано архітектурні рішення, вимоги до системи, UML-моделі та опис алгоритмів. Третій розділ присвячено програмній реалізації вебсистеми, а четвертий – перевірці працездатності, аналізу обчислювальних результатів і оцінці ефективності запропонованого підходу.

# **1 АНАЛІЗ ЗАДАЧІ РОЗРОБКИ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ**

В ситуаціях, коли прийняте рішення може значною мірою вплинути на становище в компанії, країні або світі, залучення декількох незалежних фахівців стає обов'язковим. Проте слід також вдатись до чітких процедур збору думок і оцінок, методів їх обробки та інструментів, які будуть перевіряти узгодженості.

## **1.1 Поняття колективного прийняття рішень**

Колективне прийняття рішень – це процес вироблення узгодженого вибору групою учасників, які аналізують спільну проблему, розглядають сукупність альтернатив і формують підсумкову рекомендацію. На відміну від індивідуального рішення, груповий формат включає взаємодію різних точок зору, обмін аргументами, узгодження критеріїв та пошук компромісу. У літературі з систем підтримки прийняття рішень колективний формат розглядається як окремий клас задач, у яких ключову роль відіграють не лише дані, а й комунікація між людьми, структура експертної групи, правила агрегування оцінок і якість процедури обговорення [1 -3]. Для цифрового середовища це означає, що інформаційна система повинна підтримувати не просто введення даних, а повний цикл організації групової роботи.

Проблемність колективного вибору пояснюється кількома причинами. По-перше, експерти можуть мати різний рівень компетентності, тому не всі оцінки доцільно вважати рівнозначними. По-друге, одна й та сама альтернатива може виглядати найкращою за одним критерієм і поступатися за іншим, через що необхідне багатокритеріальне узгодження. По-третє, групові рішення схильні до впливу соціально-психологічних факторів: авторитетної думки, ефекту більшості, конформізму, інформаційної асиметрії та часових обмежень. Саме тому експертні методи намагаються мінімізувати хаотичність обговорення і перевести процес у

формалізовані кроки: визначення проблеми, підготовка альтернатив, збір індивідуальних оцінок, обробка результатів, аналіз узгодженості та підготовка висновку [11; 12].

У прикладних інформаційних системах колективне прийняття рішень використовується у найрізноманітніших сферах: вибір програмних платформ, оцінювання проєктних заявок, визначення пріоритетів розвитку організації, підбір постачальників, формування дорожніх карт, конкурсний відбір ініціатив, експертна оцінка ризиків та інновацій. Незалежно від конкретної предметної області, структура задачі є подібною: існує множина альтернатив  $A = \{a_1, a_2, \dots, a_n\}$ , множина експертів  $E = \{e_1, e_2, \dots, e_m\}$  і, за потреби, набір критеріїв  $C = \{c_1, c_2, \dots, c_k\}$ . Кожний експерт формує оцінку альтернатив, після чого система виконує агрегування отриманих даних та формує підсумковий рейтинг. Такий підхід є зручним для комп'ютерної реалізації, оскільки дозволяє розділити інтерфейс збору даних і математичний модуль обробки.

Важливою властивістю колективного прийняття рішень є перевірка узгодженості. Якщо експерти суттєво розходяться в поглядах, підсумковий рейтинг може бути статистично нестійким і потребуватиме додаткового раунду оцінювання або перегляду критеріїв. У класичних роботах з теорії ранжувань для цього використовують коефіцієнт конкордації Кендалла, рангові кореляції, дисперсійні міри та ентропійні показники [6; 9]. Для вебсистеми це означає необхідність не лише вивести лідируючу альтернативу, а й пояснити, наскільки надійним є такий результат. Саме пояснюваність і прозорість обчислень є однією з ключових вимог до сучасних систем підтримки прийняття рішень.

Отже, колективне прийняття рішень доцільно розглядати як інтеграцію організаційної процедури, математичного апарату та програмної інфраструктури. У контексті даної роботи особливий інтерес становлять ситуації, коли необхідно швидко організувати сесію оцінювання, залучити декілька експертів, зафіксувати їхні індивідуальні судження й порівняти результати різними методами. Такий підхід дає змогу перейти від неструктурованого обговорення до відтворюваної цифрової

процедури, де всі етапи – від створення сесії до генерації звіту – виконуються в межах єдиної вебсистеми.

У цифрових системах колективне прийняття рішень доцільно розглядати ще й як задачу організації даних. Кожне індивідуальне судження повинно бути зафіксоване так, щоб над ним можна було виконати машинну обробку, повторний аналіз і аудит. Це означає, що система повинна зберігати не тільки кінцевий вибір, але й контекст його появи: хто оцінював, коли саме, у якому раунді, за якою шкалою, з якою вагою компетентності. Такий підхід принципово відрізняє системи підтримки прийняття рішень від звичайних чатів або засобів обговорення. У межах даної роботи ця властивість реалізується через формалізовані сутності, реляційну модель даних і повторюваний алгоритм обчислення. Саме завдяки цьому можна не лише отримати відповідь на питання 'що обрано', а й обґрунтувати 'чому саме це рішення виявилось кращим'. Для освітнього проєкту така прозорість має особливу цінність, оскільки демонструє зв'язок між теоретичною моделлю групового вибору та її практичною програмною реалізацією.

## **1.2 Аналіз існуючих систем підтримки прийняття рішень**

На практиці користувачі часто намагаються розв'язувати задачі експертного оцінювання за допомогою загальнодоступних інструментів – електронних форм, табличних процесорів, систем опитувань або офісних засобів спільної роботи. Такі рішення зручні для первинного збору даних, але не покривають повний цикл групового прийняття рішень. Вони рідко підтримують одночасний запуск декількох алгоритмів, облік компетентності експертів, багатораундовий режим і автоматичний аналіз узгодженості. У результаті значна частина обчислень переноситься в ручний режим, що збільшує ризик помилок, сповільнює процес і знижує відтворюваність результатів [1; 2].

Другу групу становлять спеціалізовані системи підтримки прийняття рішень.

Для них характерні розвинені механізми моделювання, сценарного аналізу та багатокритеріальної оцінки. Проте багато таких рішень орієнтовані або на вузьку предметну область, або на корпоративний сегмент, де критичними стають складність впровадження, ліцензійні обмеження та низька гнучкість адаптації під конкретний навчальний чи дослідницький кейс. Для навчального бакалаврського проєкту важливо створити систему, яка є достатньо простою для розгортання, але водночас має чітку модульну архітектуру і демонструє повний цикл обробки експертних оцінок.

Порівняльний аналіз наявних підходів показує декілька типових прогалин (табл. 1.1). По-перше, більшість систем використовує лише один формат подання оцінок, наприклад, бальну шкалу або просте ранжування. По-друге, відсутність уніфікованого математичного ядра не дозволяє виконати коректне порівняння результатів різних методів на тому самому наборі експертних даних. По-третє, показники узгодженості або зовсім не розраховуються, або залишаються прихованими від користувача. По-четверте, інструменти візуалізації зазвичай обмежуються таблицею результатів без аналітичних графіків та без динаміки по раундах.

Таблиця 1.1 – Порівняння підходів до підтримки експертного оцінювання

| Тип інструменту         | Збір оцінок | Порівняння методів | Узгодженість | Історія раундів      |
|-------------------------|-------------|--------------------|--------------|----------------------|
| Електронні форми        | Так         | Ні                 | Ні           | Обмежено             |
| Табличні процесори      | Так         | Частково вручну    | Частково     | Так                  |
| Спеціалізовані DSS      | Так         | Частково           | Так          | Залежить від системи |
| Розроблювана вебсистема | Так         | Так                | Так          | Так                  |

З огляду на ці недоліки, доцільно сформулювати вимоги до нової системи. Вона повинна об'єднати в одному вебінтерфейсі етапи створення сесії, керування експертною групою, введення оцінок, виконання обчислень та побудови звітів. Система має підтримувати щонайменше п'ять методів різних класів, працювати з коефіцієнтами компетентності, зберігати історію раундів і автоматично виявляти, наскільки стабільним є колективний результат. Саме таке спрямування і закладено у технічному описі проєкту, де модульність, історичність та аналітичність визначено як базові характеристики цільового продукту.

Таким чином, аналіз існуючих підходів підтверджує наявність прикладної ніші для вебсистеми навчально-дослідницького типу, яка поєднує зручність вебінтерфейсу, прозорість алгоритмів та наочність результатів. На відміну від загальних інструментів опитування, така система не лише збирає відповіді, а й виконує пояснюване порівняння декількох підходів до агрегування оцінок. Це безпосередньо підвищує якість колективного вибору та дає змогу краще обґрунтувати підсумкове рішення.

Ще однією проблемою поширених інструментів є слабка пояснюваність результату. Коли користувач заповнює електронну форму, а далі отримує лише зведену таблицю відповідей, він змушений самостійно вирішувати, за яким правилом перетворювати ці дані на рішення. Внаслідок цього різні організатори можуть на основі однакового набору відповідей отримувати різні підсумки. Розроблювана система усуває цю проблему, оскільки математичні правила обробки даних закладені в її ядро й є відтворюваними. Кожна сесія має однакову логіку проходження, а кожен метод повертає прозорий набір числових показників. Це робить систему придатною не лише для практичного використання, а й для навчання, коли важливо показати студенту або члену комісії, яким саме чином групове обговорення переходить у формалізований рейтинг.

### 1.3 Постановка задачі

Виходячи з проведеного аналізу предметної області, задачу даної роботи можна сформулювати так: необхідно розробити вебсистему, яка забезпечує підтримку колективного прийняття рішень на основі методів експертного оцінювання та надає користувачам інструменти для порівняння результатів різних методів на єдиному наборі вхідних даних. Вебсистема повинна підтримувати ролі організатора та експерта, зберігати опис сесій, альтернатив, коефіцієнти компетентності та оцінки по раундах, виконувати розрахунок підсумкового рейтингу і формувати аналітичний звіт.

Функціональна постановка задачі передбачає наявність таких ключових операцій: створення та налаштування сесії, введення альтернатив і критеріїв, реєстрація експертів, призначення ваг компетентності, збір індивідуальних оцінок, запуск обчислення за обраними методами, оцінювання рівня узгодженості, побудова візуалізацій і генерація звіту. З погляду даних система працює з сутностями 'сесія', 'альтернатива', 'експерт' та 'оцінка', а також з похідними результатами – рейтингами, метриками узгодженості та підсумковими висновками.

Нефункціональна постановка задачі включає вимоги до зручності використання, зрозумілості інтерфейсу, повторюваності обчислень, розширюваності архітектури та можливості подальшого масштабування. Оскільки система задумується як дослідницько-прикладний прототип, важливо забезпечити модульність обчислювального ядра, щоб нові методи можна було додавати без повного переписування застосунку. Крім того, значення має прозорість обчислень: користувач повинен бачити не лише переможця, а й числові показники, на основі яких зроблено рекомендацію.

Отже, підсумкова задача бакалаврської роботи полягає не просто у створенні вебсайту з формами введення, а в розробці цілісної вебсистеми з математичним ядром, підсистемою збереження даних, засобами візуалізації та звітності. Саме така постановка відповідає ідеї сучасної системи підтримки прийняття рішень, де

інформаційна частина, алгоритмічний блок і пояснювані результати функціонують як єдиний програмний комплекс.

З технічної точки зору задача містить кілька рівнів складності. На першому рівні система повинна забезпечити базову CRUD-логіку для сутностей сесії, експертів і альтернатив. На другому рівні необхідно правильно перетворити збережені дані у вхідні структури для алгоритмів. На третьому рівні потрібно організувати аналітичне представлення результатів так, щоб користувач міг швидко співставити методи між собою. Таким чином, постановка задачі охоплює і веброзробку, і математичну обробку даних, і дизайн інформаційної взаємодії.

## **2 МЕТОДИ ЕКСПЕРТНОГО ОЦІНЮВАННЯ**

При будь-якому колективному прийнятті рішень, найголовніше – це вибрати правильний метод агрегування суджень кожного з експертів. Проте слід розуміти, що жоден з методів не є універсальним, кожен з них має свої властивості, обмеження, переваги і недоліки. З цих причин в сучасній практиці прийнято застосовувати декілька методів одночасно, щоб мати змогу порівняти результати.

### **2.1 Класифікація методів експертного оцінювання**

Методи експертного оцінювання призначені для формалізації індивідуальних суджень фахівців і подальшого отримання колективного висновку. Найчастіше їх класифікують за формою подання вхідних даних та способом агрегування результату: методи бального оцінювання, ранжування, голосування, парних порівнянь, ітеративні процедури та багатокритеріальні методи [4; 5; 9]. Для програмної системи така класифікація має практичне значення, оскільки від неї залежать тип інтерфейсу введення, структура збережених даних, набір перевірок коректності та спосіб візуалізації результату.

Для цілей бакалаврської роботи було обрано п'ять методів: зважений середній бал, метод Борда, метод Кондорсе, метод Делфі та метод парних порівнянь (табл. 2). Такий набір відповідає вимозі представити різні класи підходів і дає змогу виконати змістовне порівняння результатів, як того вимагає технічне завдання. Крім того, ці методи мають різну чутливість до розподілу оцінок, структури ранжувань та експертних ваг, що дозволяє продемонструвати цінність мультиалгоритмічного підходу у межах однієї вебсистеми.

Таблиця 2 – Класифікація методів експертного оцінювання

| Клас методів       | Приклади              | Форма вхідних даних   | Переваги                     | Обмеження                 |
|--------------------|-----------------------|-----------------------|------------------------------|---------------------------|
| Бальні             | Зважений середній бал | Числові оцінки        | Простота інтерпретації       | Чутливість до шкали       |
| Рангові            | Борда                 | Порядок альтернатив   | Зручно при відносному виборі | Не показує силу переваги  |
| Парні зіставлення  | Кондорсе, АНР         | Попарні переваги      | Локальна деталізація         | Більша складність         |
| Ітеративні         | Делфі                 | Кілька раундів оцінок | Підвищує консенсус           | Потребує часу             |
| Багатокритеріальні | TOPSIS, ELECTRE       | Матриці критеріїв     | Глибокий аналіз              | Складніше для користувача |

Вибір набору методів для системи не може бути випадковим. Якщо реалізувати лише близькі між собою алгоритми, користувач отримає майже однакові результати і не побачить справжньої цінності порівняння. Тому в роботі свідомо поєднано методи, що дивляться на одні й ті самі оцінки під різними кутами: один працює з абсолютними балами, інший – з порядком місць, третій – з попарними перемогами, четвертий – із процедурою зближення думок у часі, а п'ятий – з матричним поданням відносних переваг. У сумі це створює багатовимірний погляд на задачу. Якщо лідер збігається для всіх методів, організатор може вважати рішення більш обґрунтованим. Якщо ж результати розбігаються, це вказує на чутливість задачі до форми подання експертних суджень. Саме такий дослідницький ефект є одним із ключових результатів розроблюваної вебсистеми.

## 2.2 Зважений середній бал

До класу бальних методів належить зважений середній бал. Експерт оцінює кожну альтернативу у числовій шкалі, після чого система обчислює агрегований показник з урахуванням ваги компетентності. Формально для альтернативи  $a_j$  інтегральна оцінка може бути подана як

$$S(a_j) = \frac{\sum (w_i * x_{ij})}{\sum w_i}$$

де  $w_i$  – коефіцієнт компетентності  $i$ -го експерта,

$x_{ij}$  – поставлений ним бал.

Перевагою методу є простота інтерпретації та зручність порівняння альтернатив, однак він чутливий до вибору шкали та не завжди відображає відносні переваги між варіантами [1; 11].

Алгоритмічний рівень системи спроектовано як набір незалежних обчислювальних модулів. Спільним для всіх методів є те, що вони працюють на одному й тому самому наборі оцінок, але по-різному інтерпретують вихідні дані. Такий підхід дозволяє виконати коректне порівняння результатів. Перший модуль реалізує зважений середній бал. Алгоритм послідовно проходить по всіх експертах, множить кожну оцінку на відповідну вагу компетентності та нормує суму на загальну суму ваг. Підсумком є числовий рейтинг альтернатив у шкалі, сумісний з початковими бальними оцінками.

## 2.3 Метод Борда

Методи ранжування працюють з відносним порядком альтернатив, а не з абсолютними балами. У цій роботі обрано метод Борда, за яким кожна позиція у ранжуванні перетворюється на певну кількість балів, а потім сумується по всіх експертах. Такий підхід зручний для випадків, коли експерту складно призначити

точну числову оцінку, але легко розташувати варіанти від кращого до гіршого. Метод Борда є стійкішим до локальних спотворень шкали, проте може недооцінювати силу переваги між близькими та далекими альтернативами [7; 11].

Другий модуль реалізує метод Борда. На основі оцінок кожного експерта система будує локальне ранжування альтернатив від найкращої до найгіршої. Після цього кожній позиції призначається кількість очок: для  $n$  альтернатив перша позиція отримує  $n - 1$  балів, остання – 0. З урахуванням ваг експертів усі значення сумуються, після чого визначається альтернатива з найбільшим сумарним рахунком. Цей підхід корисний тоді, коли важливий саме порядок пріоритетів, а не різниця між конкретними числами.

## 2.4 Метод Кондорсе

Для класу парних зіставлень у роботі використано два підходи. Перший – метод Кондорсе, який порівнює альтернативи попарно та визначає переможця за більшістю індивідуальних переваг [8]. Другий – метод парних порівнянь у стилі АНР, де для кожної пари формується відношення переваги, а підсумкові пріоритети отримуються на основі нормалізованої матриці [5]. Методи цього класу добре демонструють локальну силу переваги однієї альтернативи над іншою, однак вимагають більшої кількості обчислень і ретельної перевірки узгодженості.

Третій модуль реалізує метод Кондорсе. Для кожної пари альтернатив ( $a_i$ ,  $a_j$ ) система визначає, скільки експертів надають перевагу першій, а скільки – другій. Якщо одна альтернатива перемагає іншу у більшості попарних зіставлень, їй нараховується перемога. Переможець Кондорсе – це альтернатива, яка перемагає всі інші або має найбільшу кількість попарних перемог. У загальному випадку можуть виникати цикли переваг, тому у прототипі передбачено використання кількості перемог як підсумкового показника для побудови рейтингу.

## 2.5 Метод Делфі

Окрему групу становлять ітеративні методи. Найвідомішим серед них є метод Делфі, який ґрунтується на багатораундовому опитуванні експертів з анонімним зворотним зв'язком між раундами [4]. Після кожного раунду учасники отримують узагальнену статистику та можуть переглянути власні оцінки. У цифровому форматі це створює передумови для поступового зближення позицій і підвищення коефіцієнта узгодженості. Саме тому у спроєктованій системі метод Делфі реалізовано як окремий модуль, що дозволяє врахувати оцінки попереднього й поточного раундів при формуванні консенсусного результату.

Четвертий модуль присвячений методу Делфі. У системі він реалізований як ітеративний механізм, який бере до уваги оцінки поточного та попереднього раундів. Ідея полягає в тому, щоб не просто дублювати нові оцінки, а сформувати консенсусну оцінку, де більша вага відводиться останньому раунду, але частково зберігається інерція попередніх значень. Це дозволяє моделювати процес поступового зближення позицій після анонімного зворотного зв'язку між раундами, що є суттю класичної процедури Делфі [4].

## 2.6 Метод парних порівнянь (АНР)

П'ятий модуль реалізує метод парних порівнянь у спрощеній АНР-подібній формі. На основі агрегованих оцінок будується матриця відносної переваги між усіма парами альтернатив. Далі матриця нормується за стовпцями, а підсумковий вектор пріоритетів знаходиться як середнє значення відповідних рядків. Попри спрощеність, такий підхід демонструє логіку методів класу АНР і дозволяє порівняти результат з іншими алгоритмами [5; 10]. Разом з алгоритмами працює підсистема аналітики, яка обчислює Kendall W, коефіцієнт варіації та ентропію. Саме вона забезпечує інтерпретацію того, наскільки стабільним є одержаний рейтинг.

Під час реалізації алгоритмів важливим було не лише отримати результат, а й забезпечити єдність формату повернення. Кожен метод повертає словник із двома ключами: `scores` і `winner`. Така уніфікація дає змогу виводити картки методів у шаблоні одним універсальним циклом, не прописуючи окрему логіку під кожен алгоритм. З практичного погляду це невелика, але дуже показова проєктна деталь: вона демонструє, як добре продуманий інтерфейс між модулями зменшує складність усієї системи. Саме через такі рішення прототип стає не набором окремих функцій, а цілісним програмним продуктом.

## 3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

На цьому етапі проектні рішення, які були створені в попередньому розділі, отримали відповідну програмну форму. Моделі даних перетворюються на таблиці баз даних, а алгоритми – на функції з реальною логікою. У цьому моменті стає очевидним, наскільки добре були прийняті рішення щодо проекту: добре розроблена система працює послідовно і без проблем, а кожен новий модуль органічно вписується в уже створену структуру.

### 3.1 Вибір технологій

Для реалізації прототипу обрано стек технологій, який поєднує простоту розгортання, наочність коду та достатню функціональність для бакалаврського проекту. Серверну частину розроблено мовою Python із використанням мікрофреймворку Flask. Вибір Python обумовлений її читабельним синтаксисом, великою кількістю бібліотек для аналітичних обчислень та швидкістю створення навчальних вебзастосунків [16]. Flask, у свою чергу, дає можливість реалізувати маршрутизацію, шаблони, роботу з формами та інтеграцію з базою даних без надлишкової складності, характерної для важчих фреймворків.

Для зберігання даних обрано SQLite. Її використання є виправданим у випадку локального або демонстраційного застосунку, де головним критерієм є швидкий запуск без додаткових серверних компонентів. Водночас реляційна модель даних залишається сумісною з повноцінними серверними СУБД, тому в майбутньому система може бути адаптована під PostgreSQL без зміни предметної моделі. На клієнтському рівні застосовано HTML, CSS і шаблони Jinja2, що дозволяє формувати динамічні сторінки на основі результатів, отриманих від серверної частини.

Для створення звітів використано бібліотеку reportlab. Це дозволило без залучення сторонніх офісних компонентів формувати PDF-документи з ключовими

результатами сесії. Важливо, що така реалізація може працювати у будь-якому стандартному Python-оточенні, що позитивно впливає на переносимість застосунку. Додатково для підготовки ілюстрацій та результатів тестування було використано matplotlib і засоби побудови програмних схем, що є типовим інструментарієм для навчальних дослідницьких робіт.

Обраний стек технологій добре відповідає цілям бакалаврської роботи. Він дозволяє продемонструвати повний цикл створення прикладного вебзастосунку: від моделювання даних і серверної логіки до формування вебінтерфейсу та аналітичної обробки. Важливо й те, що всі компоненти є достатньо прозорими для детального опису в пояснювальній записці, що відповідає вимозі наявності програмної реалізації як складової кваліфікаційної роботи.

Додатковою перевагою обраного стеку є його доступність для перевірки й тиражування. Викладач, рецензент або інший студент може розгорнути проєкт без спеціалізованого середовища: достатньо стандартного Python-інтерпретатора і пакунків із requirements.txt. Це важливо для захисту кваліфікаційної роботи, тому що демонстраційна система повинна бути стабільною та простою у запуску. Крім того, Python є природним вибором для проєктів, де поєднуються вебтехнології та обчислювальна логіка, оскільки в одній мові можна реалізувати і сервер, і математичні методи, і частину підготовки графічних матеріалів.

## **3.2 Функціональні вимоги**

Функціональні вимоги визначають, що саме має робити система для кінцевого користувача. На підставі поставленої задачі до обов'язкових функцій належать: створення сесії експертного оцінювання, введення опису проблеми та переліку альтернатив, реєстрація експертів і задання їх коефіцієнтів компетентності, введення оцінок, запуск алгоритмів, відображення результатів у табличній та графічній формах, а також генерація звітів. Функціональні вимоги зображено в таблиці 3.1.

Таблиця 3.1 – Основні функціональні вимоги до системи

| Позначення | Опис вимоги                               | Пріоритет |
|------------|-------------------------------------------|-----------|
| F1         | Створення та редагування сесій оцінювання | Високий   |
| F2         | Введення альтернатив і опису задачі       | Високий   |
| F3         | Реєстрація експертів і ваг компетентності | Високий   |
| F4         | Збір оцінок у раундах                     | Високий   |
| F5         | Запуск п'яти методів оцінювання           | Високий   |
| F6         | Розрахунок Kendall W, CV та ентропії      | Високий   |
| F7         | Відображення рейтингів і візуалізацій     | Середній  |
| F8         | Формування PDF-звіту                      | Середній  |

Нефункціональні вимоги описують якісні характеристики програмного продукту (табл. 3.2). Для спроектованої вебсистеми критичними є зрозумілість інтерфейсу, швидкодія на малих і середніх наборах даних, розширюваність, модифікованість, повторюваність обчислень і відмовостійкість щодо типових помилок введення. Важливим є також забезпечення логічної цілісності даних: система не повинна втрачати зв'язок між експертом, сесією, альтернативою та раундом. З точки зору якості програмного продукту ці вимоги корелюють з характеристиками функціональної придатності, зручності використання та супроводжуваності, описаними в ISO/IEC 25010 [18].

Таблиця 3.2 – Нefункціональні вимоги

| Категорія      | Вимога                                         | Практичне значення                  |
|----------------|------------------------------------------------|-------------------------------------|
| Зручність      | Зрозумілий інтерфейс і проста навігація        | Зменшення часу навчання користувача |
| Продуктивність | Швидкий розрахунок на малих і середніх наборах | Комфортна робота в браузері         |

|                 |                                              |                                 |
|-----------------|----------------------------------------------|---------------------------------|
| Модифікованість | Можливість додавання нових методів           | Розширення без перепроєктування |
| Надійність      | Цілісність зв'язків між сутностями           | Уникнення втрати даних          |
| Портативність   | Можливість запуску без окремого сервера СУБД | Проста демонстрація проекту     |

Кінець таблиці 3.2

Під час проєктування було закладено також вимоги безпеки і керованості. Оскільки система працює з експертними судженнями, вона повинна коректно розділяти ролі організатора та експерта, а також бути готовою до подальшого розширення механізмами автентифікації, логування та контролю доступу. У прототипі ці засоби реалізовано у спрощеному вигляді, однак архітектура дозволяє інтегрувати їх надалі без суттєвої перебудови. Таким чином, функціональні й нефункціональні вимоги утворюють основу для проєктних рішень, які деталізуються в наступних підрозділах.

Під час формування вимог було враховано ще одну важливу особливість – баланс між універсальністю й простотою. Якщо система буде надмірно гнучкою, вона стане складною у використанні. Якщо ж надто спрощеною, то втратить дослідницьку цінність. Тому в проєкті прийнято компромісне рішення: система підтримує найважливіші сценарії експертного оцінювання без перевантаження другорядними опціями. Наприклад, у прототипі не реалізовано складне керування окремими критеріями доступу чи асинхронне сповіщення користувачів, зате повністю покрито базову логіку сесії, раундів, методів та аналітики. Саме такий баланс робить систему придатною і для демонстрації, і для подальшого розширення

Для формалізації поведінки системи та її взаємодії з користувачами використано UML-моделювання. Це дозволяє на ранньому етапі відокремити функціональні сценарії, зрозуміти межі відповідальності компонентів і підготувати основу для подальшої реалізації [17]. У роботі застосовано use case діаграму, яка відображає основних акторів – організатора та експерта – і пов'язує їх із базовими

сценаріями: створенням сесії, введенням оцінок, запуском обчислень, переглядом результатів та формуванням звітів (рис. 3.1). Така модель дає змогу швидко перевірити, чи всі бізнес-функції покрито системою.

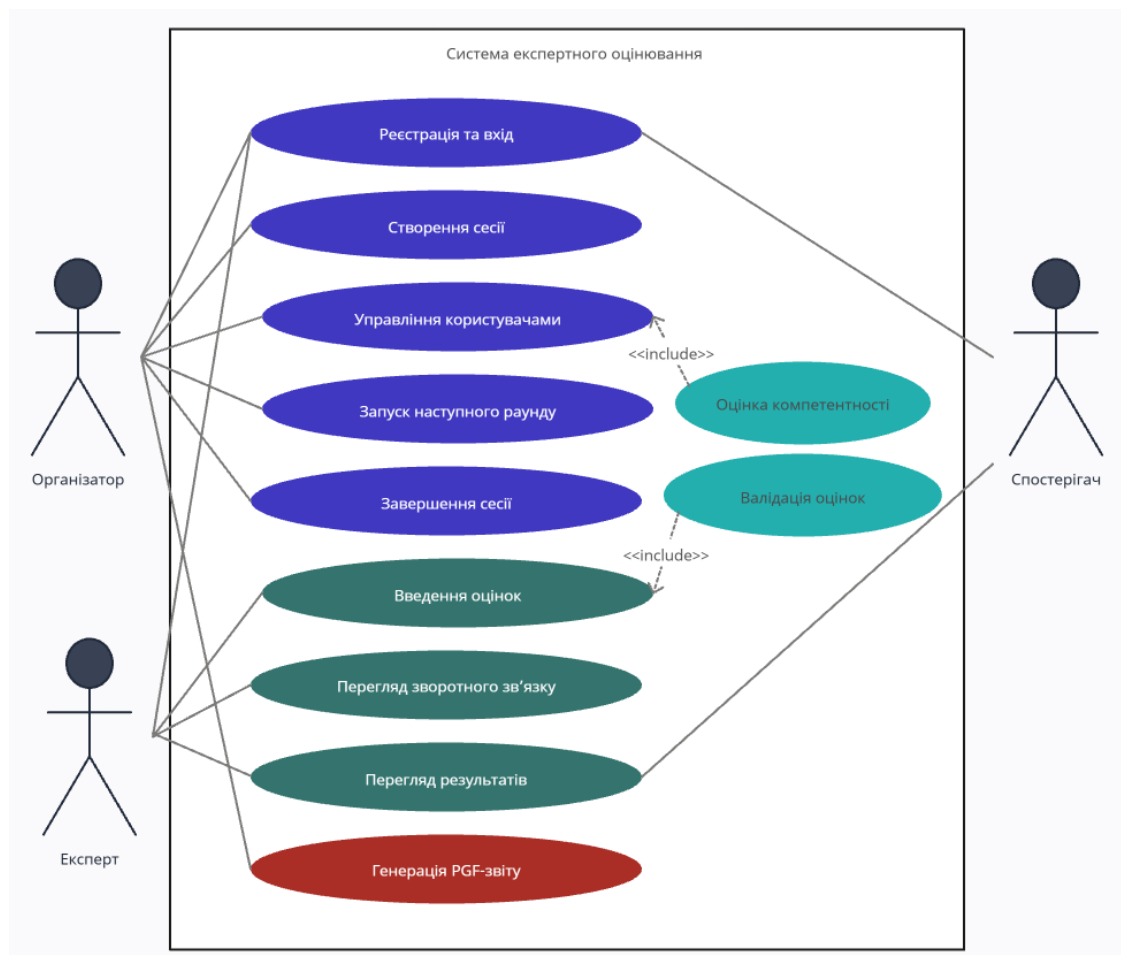


Рисунок 3.1 – Діаграма прецедентів

Крім діаграми прецедентів, важливою є модель взаємодії. Sequence-діаграма дозволяє показати динаміку обробки сценарію ”проведення сесії оцінювання”: організатор створює сесію через вебінтерфейс; сервер перевіряє дані й записує їх у базу; експерти надсилають оцінки; сервер формує набір даних для модулів; обчислювальні сервіси повертають рейтинги та аналітичні показники; після цього результат відображається у вебінтерфейсі й може бути експортований до PDF (рис. 3.2). Така модель корисна тим, що дозволяє побачити не лише структуру системи, а й порядок обміну повідомленнями між підсистемами.

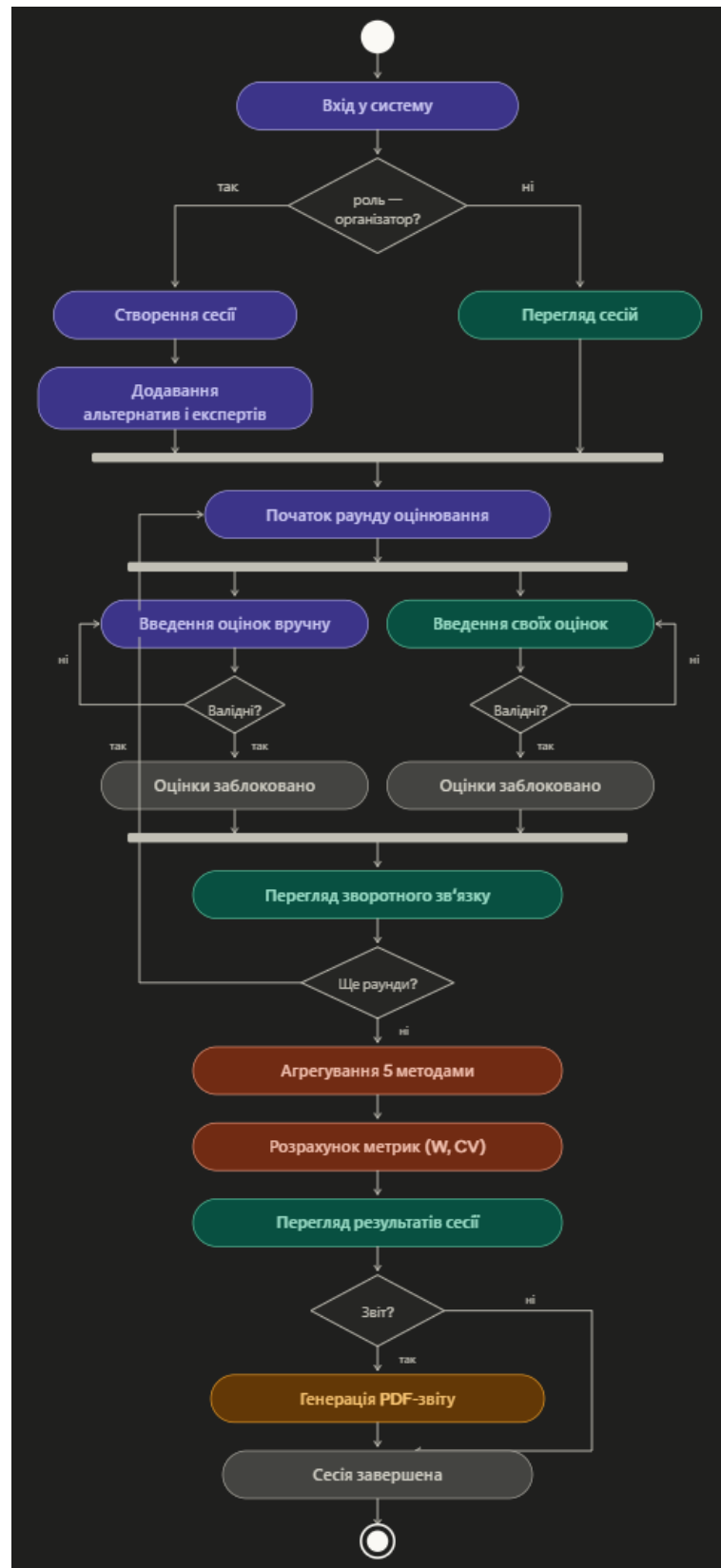


Рисунок 3.2 – Діаграма діяльності для обробки експертних оцінок

UML-моделі у межах цього проєкту виконують не декоративну, а практичну функцію. Вони допомагають вирішити, де саме має знаходитися алгоритмічна логіка, які об'єкти потрібні для передавання даних, які сутності варто зберігати в базі, а які – обчислювати на льоту. Наприклад, результати окремих методів доцільно не зберігати постійно в таблицях, а формувати під час запиту на перегляд сесії, щоб уникнути застарілих даних при повторному запуску обчислень. Такі рішення простіше обґрунтувати, коли перед очима є модель взаємодії компонентів.

Використання UML також позитивно впливає на якість супроводу системи. Студент або інший розробник, який надалі вдосконалюватиме прототип, отримує не лише вихідний код, а й формалізований опис сценаріїв роботи. Це знижує залежність проєкту від інтуїтивного розуміння автора й робить систему більш придатною до командної розробки. Саме тому в бакалаврській роботі UML-діаграми слугують важливою проміжною ланкою між аналізом предметної області та програмною реалізацією.

Важливо підкреслити, що UML-моделі не суперечать гнучкому стилю реалізації на Flask, а навпаки дисциплінують його. Мікрофреймворки часто приваблюють простотою, але водночас створюють ризик хаотичного зростання коду, коли маршрути, шаблони та алгоритми переплітаються між собою. Попереднє моделювання дозволяє цього уникнути: ще до початку програмування розробник бачить ролі, сценарії та межі компонентів. У межах навчального проєкту це є особливо корисним, бо дозволяє продемонструвати не лише навички кодування, а й уміння проєктувати систему на концептуальному рівні.

### **3.3 Архітектура системи**

Архітектура розроблюваної вебсистеми спроектована за модульним принципом. Такий підхід дає змогу розділити відповідальність між інтерфейсом користувача, серверною логікою, підсистемою збереження даних, блоком

алгоритмічної обробки та модулем аналітики. У контексті бакалаврського проєкту модульність є критично важливою, оскільки вона дозволяє не лише реалізувати базову функціональність, а й продемонструвати можливість розширення системи новими методами оцінювання без руйнування вже створеної структури. Загальна архітектурна схема наведена на рисунку 3.3.

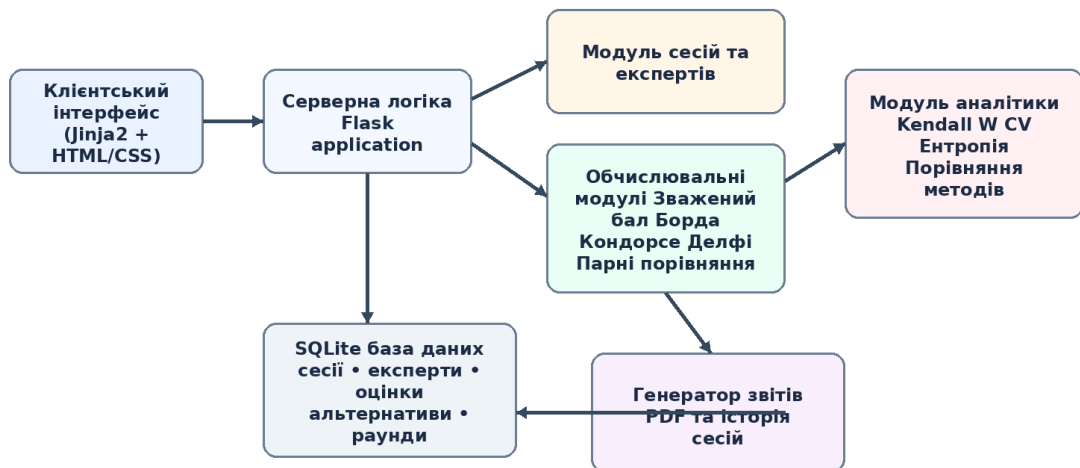


Рисунок 3.3 – Загальна архітектура вебсистеми

На рівні взаємодії з користувачем система реалізує вебінтерфейс, який надає організаторові засоби створення сесій, перегляду підсумків, формування звітів та керування експертами. Для експерта вебінтерфейс виконує іншу роль: відображає перелік альтернатив, дозволяє ввести оцінки й отримати зворотний зв'язок у багатораундовому режимі. Серверний рівень забезпечує маршрутизацію запитів, перевірку коректності введених даних, виклик обчислювальних модулів та підготовку даних для відображення у шаблонах.

Ядром архітектури виступає набір алгоритмічних сервісів. Кожен метод експертного оцінювання реалізовано як окрему функцію з уніфікованим інтерфейсом: на вході модуль отримує словник оцінок експертів, їх ваги та список альтернатив, а на виході повертає числові показники і рекомендованого переможця. Завдяки такому підходу вебсистема може виконувати паралельне порівняння різних підходів на

одних і тих самих даних. Поруч із цим працює модуль метрик, що обчислює коефіцієнт конкордації Кендалла, коефіцієнт варіації та ентропію розподілу оцінок.

Підсистема збереження даних побудована на реляційній моделі з використанням SQLite як вбудованої бази даних. Вона містить таблиці сесій, альтернатив, експертів та оцінок, що дозволяє зберігати як поточний стан системи, так і історію раундів. Реляційний підхід є доречним для цієї предметної області, оскільки зв'язки між сутностями мають чітку структуру: одна сесія включає багато альтернатив і багато експертів, а одна оцінка пов'язує конкретного експерта з конкретною альтернативою в межах заданого раунду.

Окремий архітектурний шар становить підсистема звітності. Після виконання обчислень вона формує документальний підсумок сесії, який містить назву задачі, опис альтернатив, значення аналітичних показників, результати всіх методів і рекомендоване рішення. У прототипі ця функція реалізована через генерацію PDF-файлу, однак архітектура дозволяє в подальшому розширити її експортуванням у DOCX, XLSX або інтеграцією з зовнішніми API. Загалом архітектура системи задовольняє вимогу гнучкості та придатності до подальшої модернізації.

Застосована архітектура також відповідає принципу низького зв'язування і високої згуртованості. Логіка доступу до даних зосереджена біля функцій роботи з БД, логіка алгоритмів винесена до окремих сервісів, а шаблони відповідають лише за представлення. Таке розмежування особливо важливе у випадку систем підтримки прийняття рішень, тому що математичне ядро має бути максимально незалежним від деталей інтерфейсу. У майбутньому це дозволить, наприклад, замінити вебшаблони на REST API або мобільний клієнт, не переписуючи самі методи оцінювання. Для бакалаврського рівня така архітектурна дисципліна є важливим показником зрілості програмного рішення.

### 3.4 Концептуальна схема бази даних

Проектування бази даних виконано відповідно до інформаційної моделі предметної області. Базова сутність 'Sessions' відображає окрему процедуру експертного оцінювання і містить назву, опис та часові атрибути. Сутність 'Alternatives' представляє варіанти, між якими здійснюється вибір, а сутність 'Expert' містить інформацію про учасників експертної панелі та їх ваги компетентності. Центральною таблицею є 'Scores', де фіксуються оцінки кожного експерта для кожної альтернативи з прив'язкою до конкретного раунду. ER-діаграма наведена на рисунку 3.4.

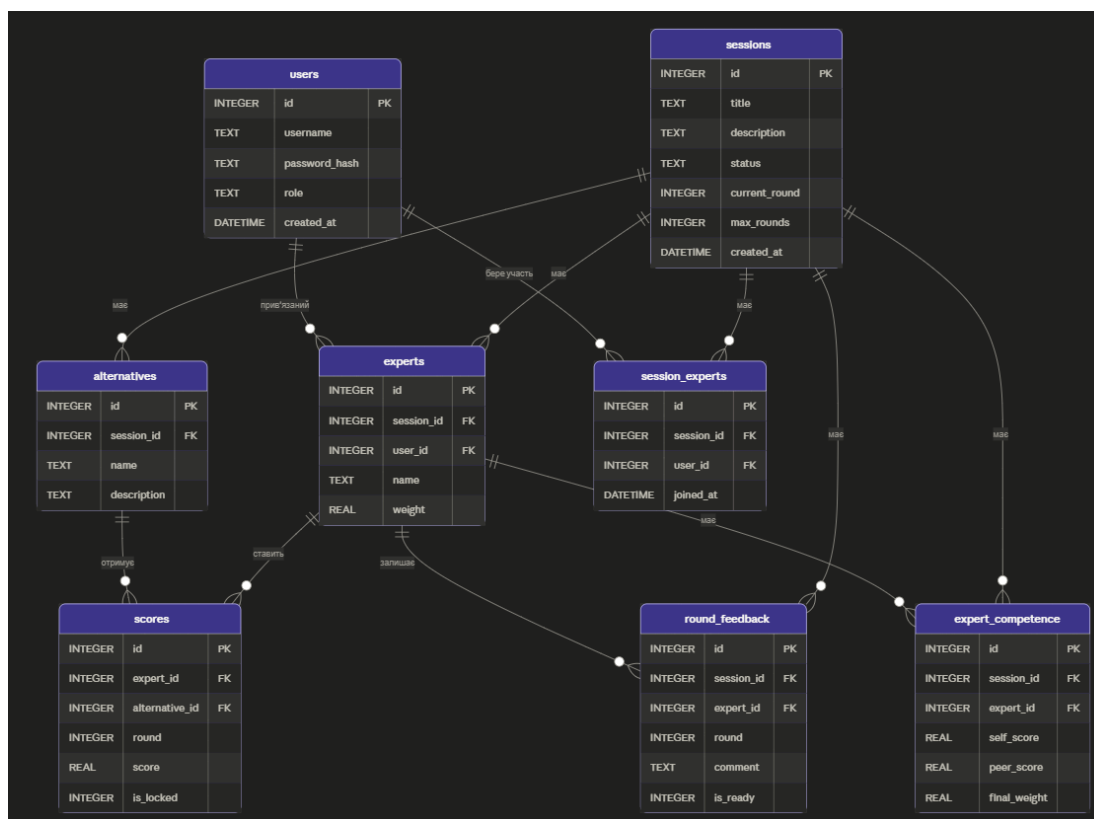


Рисунок 3.4 – ER-діаграма бази даних

Реляційна схема побудована так, щоб зберігати історію і водночас уникати дублювання даних. Наприклад, альтернативи та експерти зберігаються окремо й

пов'язуються з оцінками через зовнішні ключі. Це дає змогу легко отримати відповіді на типові запити: показати всі оцінки певної сесії; знайти середнє значення по конкретній альтернативі; проаналізувати зміни між раундами; перевірити, чи всі експерти завершили введення даних. Крім того, така схема добре масштабується при зростанні кількості сесій і не потребує змін у структурі таблиць для нових сценаріїв використання.

Під час проектування було враховано й питання цілісності. Зовнішні ключі забезпечують логічний зв'язок між таблицями, а обов'язкові поля не дозволяють зберегти неповноцінний запис без назви сесії, альтернативи чи експерта. Для прототипу цього достатньо, щоб продемонструвати надійну модель даних. У реальному виробничому варіанті доцільно було б доповнити схему окремими таблицями ролей, журналом подій, таблицями критеріїв оцінювання та об'єктами аудиту, однак для бакалаврської роботи така деталізація не є необхідною.

Вибір SQLite пояснюється кількома факторами. По-перше, вона не потребує окремого серверного процесу й легко переноситься разом із застосунком. По-друге, для демонстраційного прототипу її продуктивності цілком достатньо. По-третє, її використання спрощує розгортання проєкту та перевірку на будь-якому навчальному комп'ютері. При потребі архітектура дозволяє перенести систему на PostgreSQL або іншу СУБД практично без зміни логіки застосунку, оскільки модель предметної області вже спроектована реляційно і чітко нормалізована.

Ще одним аргументом на користь обраної моделі БД є можливість виконувати порівняльний аналіз не лише в межах однієї сесії, а й між кількома сесіями в майбутньому. Якщо система використовуватиметься довше, можна буде накопичувати історичний масив даних і досліджувати, як змінюється поведінка експертних груп, які методи частіше дають стійкий результат, які типи задач виявляються більш конфліктними. Отже, навіть спрощена реляційна структура вже зараз створює передумови для аналітики вищого рівня. Це підсилює науково-прикладну цінність розробки.

Реалізація бази даних у програмному коді виконана через набір SQL-інструкцій, що створюють таблиці `sessions`, `alternatives`, `experts` і `scores`. Кожна таблиця має первинний ключ, а зв'язки між сутностями задаються через зовнішні ключі. Такий підхід дозволяє зберігати дані в чітко нормалізованому вигляді та підтримувати цілісність моделі. Вибір явних SQL-скриптів замість важкого ORM робить структуру застосунку зрозумілішою для освітнього проєкту: студент бачить, які саме таблиці створюються й які поля використовуються в логіці системи.

Особливу увагу приділено таблиці `scores`, оскільки саме вона містить ключові числові дані. Один запис у цій таблиці відповідає оцінці конкретної альтернативи конкретним експертом у межах конкретного раунду. Завдяки полю `round_no` система може зберігати не тільки підсумкові значення, а й історію зміни думки експертів, що є критичним для реалізації Делфі-подібного режиму. Така структура також спрощує аналітику: наприклад, можна окремо досліджувати стабільність оцінок між раундами.

У процесі реалізації застосовано допоміжні запити для вибірки даних у потрібному порядку. Так, під час формування сесії відбираються всі оцінки, об'єднані з назвами альтернатив і даними експертів. Це дозволяє вже на рівні SQL отримати зрозумілий для відображення набір даних. Крім того, у коді передбачено початкове заповнення демонстраційного прикладу, що включає одну сесію, чотири альтернативи, п'ять експертів і два раунди оцінювання. Наявність стартового набору даних є корисною для тестування та презентації системи.

Реалізація бази даних у запропонованому вигляді є збалансованим рішенням між простотою і функціональністю. Вона не перевантажена надлишковими таблицями, але водночас повністю покриває сценарії, описані в технічному завданні: збереження сесій, альтернатив, експертів, раундів і оцінок. На практиці це створює надійну основу для обчислювальних модулів і формування звітності.

Для навчального проєкту доречним є й те, що робота з SQLite показує студенту фундаментальні принципи взаємодії прикладної програми з даними: створення таблиць, виконання `SELECT`, `INSERT`, `DELETE`, формування з'єднаних запитів, перетворення результатів у формат, придатний для логіки застосунку. Це підсилює прикладний характер бакалаврської роботи, оскільки демонструє не лише знання теорії БД, а й реальне використання СУБД у складі вебзастосунку.

### 3.5 Реалізація методів експертного оцінювання

Серверна частина застосунку побудована навколо центрального файлу `app.py`, який ініціалізує Flask-застосунок, налаштовує з'єднання з базою даних, визначає маршрути й координує взаємодію між шаблонами, модулями алгоритмів та підсистемою звітності. На старті застосунок виконує ініціалізацію БД, перевіряє наявність демонстраційних даних і, за потреби, заповнює базу тестовою сесією. Така організація дозволяє одразу продемонструвати роботу системи без додаткового ручного наповнення.

У серверній частині реалізовано декілька базових маршрутів. Маршрут головної сторінки відображає перелік наявних сесій. Маршрут створення нової сесії приймає опис задачі, список альтернатив, експертів і коефіцієнтів компетентності, перевіряє цілісність введених даних і формує первинні записи в базі. Маршрут перегляду конкретної сесії виконує збирання всіх пов'язаних даних: саму сесію, альтернативи, експертів, усі оцінки, а також результати розрахунків і аналітичні показники. Завдяки цьому шаблон отримує уже повністю підготовлений набір структурованої інформації.

Ключовим елементом серверної логіки є функція `session_summary`, яка виконує агрегацію даних сесії. Вона перетворює сирі записи з таблиці оцінок у формат, зручний для роботи алгоритмічних модулів: словники з оцінками експертів по раундах, списки альтернатив і ваг компетентності. Після цього функція послідовно викликає кожен із п'яти методів, а також розраховує Kendall W, коефіцієнти варіації й ентропію. Така централізація логіки дозволяє уникнути дублювання коду та полегшує подальше розширення системи.

Окремо реалізовано маршрут генерації PDF-звіту. Під час його виклику система формує короткий документ з назвою сесії, описом, показниками узгодженості та підсумками всіх методів. У практичному сенсі це демонструє завершеність процесу: організатор не лише отримує рейтинг на екрані, а й може зберегти формалізований результат для подальшого використання. Загалом серверна частина забезпечує усі

базові бізнес-процеси, закладені у технічному завданні, і слугує ядром логіки прототипу.

З інженерної точки зору важливо, що серверна частина реалізує не лише бізнес-логіку, а й контроль коректності вхідних даних. Наприклад, під час створення сесії перевіряється мінімальна кількість альтернатив та експертів, а також відповідність кількості коефіцієнтів кількості учасників. Такі перевірки є необхідними, адже без них математичні модулі могли б отримувати неконсистентний вхід і формувати недостовірні результати. У результаті сервер виконує роль 'гаранта коректності', що є типовою функцією для сучасних вебсистем.

Реалізація алгоритмів винесена в окремий пакет `services`. Така організація підвищує читабельність коду та відповідає принципам модульного проектування [13; 19]. У файлі `methods.py` зосереджено функції, що реалізують п'ять методів експертного оцінювання. Кожна з них приймає на вхід структуру `latest_scores` або `round_map`, список альтернатив і масив даних про експертів. Уніфікація інтерфейсу дає змогу легко додавати нові методи, не змінюючи серверні маршрути або шаблони.

Функція `weighted_score_method` обчислює інтегральні бали шляхом перемноження експертних оцінок на ваги компетентності. Функція `borda_method` перетворює локальні ранжування у бали за правилом Борда. Реалізація `condorcet_method` формує попарні зіставлення між усіма альтернативами та підраховує кількість перемог. Для підтримки багатораундового режиму реалізовано `delphi_method`, що комбінує оцінки поточного та попереднього раундів у пропорції 0.7 до 0.3. Нарешті, `ahp_like_method` будує матрицю відносної переваги й обчислює вектор пріоритетів на основі нормалізації. Повний код даних функцій наведено в Додатку А.

Окремий файл `metrics.py` містить аналітичні функції. Функція `build_rankings_from_scores` перетворює числові оцінки на впорядковані списки альтернатив для кожного експерта. На основі цього `kendall_w_from_rankings` обчислює коефіцієнт конкордації, який характеризує узгодженість колективної думки. Додатково `variation_coefficient` визначає відносну варіативність оцінок по кожній альтернативі, а `entropy_metric` оцінює рівень невизначеності розподілу. Разом

ці функції дозволяють перейти від простого ранжування до аналітичного тлумачення результатів.

У межах бакалаврського проєкту особливу увагу приділено пояснюваності реалізації. Алгоритми написані без надмірної оптимізації та з максимально прозорою логікою, щоб кожен крок обчислення можна було відстежити під час демонстрації або захисту. Такий стиль є доцільним у навчальній роботі, оскільки показує не лише те, що система видає результат, а й те, яким чином цей результат формується. Саме це забезпечує науково-прикладну цінність програмної реалізації.

Модульна реалізація алгоритмів створює ще одну важливу перевагу: можливість порівнювати програмну складність різних підходів. У межах майбутнього розвитку можна буде легко додати, наприклад, TOPSIS, ELECTRE або метод медіани Кемені й одразу інтегрувати їх у спільний інтерфейс системи. Таким чином, створене ядро є не закритим набором функцій, а платформою для подальших досліджень. Для бакалаврської роботи це цінно тим, що проєкт має потенціал продовження на магістерському або науковому рівні.

## **4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ**

Розроблена система потребує перевірки. Це не просто переконатися, що код працює без помилок; важливо також переконатися, що система дає правильні результати на реальних даних і поводить себе коректно в різних умовах використання. Етап, який відрізняє ефективний прототип від надійного програмного продукту, називається тестуванням. У цьому розділі система перевіряється на демонстраційному прикладі, а результати аналізуються та оцінюється ефективність використання методу.

### **4.1 Системні вимоги**

Перед реалізацією програмної частини проєкту, потрібно чітко розуміти, з чого повинна складатись система, налаштувати взаємодію її частин між собою для того, щоб уникнути хаотичного додавання функцій без розуміння загального плану.

### **4.2 Сценарій роботи користувачів з системою**

Клієнтська частина системи реалізована у вигляді набору HTML-шаблонів і таблиць стилів. Її основне завдання – зробити складний аналітичний процес зрозумілим для користувача. На головній сторінці відображається перелік доступних сесій, що дозволяє швидко перейти до перегляду вже створених процедур оцінювання. Форма створення нової сесії містить поля для введення назви, опису, переліку альтернатив, експертів і коефіцієнтів компетентності. Інтерфейс побудовано так, щоб мінімізувати кількість дій користувача й водночас забезпечити достатню кількість вхідних даних для запуску алгоритмів.

Сторінка перегляду сесії є основним аналітичним екраном системи. Вона містить загальну інформацію про сесію, таблицю оцінок експертів, блок статистичних показників і картки з результатами кожного методу. Така композиція інтерфейсу не випадкова: користувач спочатку бачить контекст задачі, далі – вхідні дані, після чого може порівняти різні методи та оцінити надійність результату. Завдяки цьому система залишається зручною навіть для користувача, який не є фахівцем з математичних методів.

У стилях застосовано нейтральну візуальну палітру, карткове групування блоків і чітку типографіку. Це робить інтерфейс сучасним і придатним для демонстрації на захисті. Разом із тим структура шаблонів залишається простою: усі сторінки наслідують базовий шаблон із заголовком, навігацією та контейнером контенту. Така організація не лише спрощує підтримку коду, а й відповідає принципам DRY – повторювані елементи не дублюються в кожному файлі окремо.

Для цілей бакалаврської роботи інтерфейс є достатньо функціональним і завершеним (рис. 4.1). Він демонструє, що система не обмежується набором алгоритмів, а надає повноцінний користувацький шар для керування сесіями, введення даних, перегляду підсумків та експорту звіту. Саме поєднання серверної логіки з продуманим інтерфейсом дозволяє говорити про створення вебсистеми, а не просто бібліотеки окремих обчислювальних функцій.

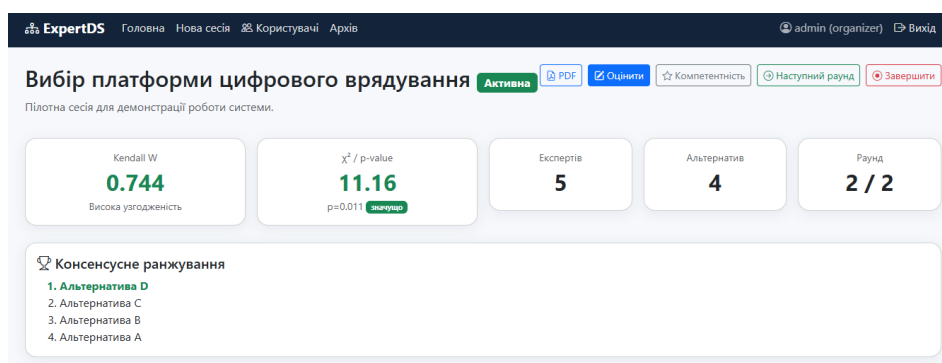


Рисунок 4.1 – Макет панелі організатора

Під час розробки інтерфейсу було враховано принцип інформаційної ієрархії. Користувач насамперед повинен бачити головну мету сторінки, тому заголовки та ключові індикатори винесено у верхню частину екрана. Далі подаються таблиці з вхідними даними, а після них – аналітичні блоки з підсумками. Така послідовність відповідає природній логіці сприйняття: від контексту і фактів до висновку. У підсумку інтерфейс не приховує математичну складність задачі, але робить її зрозумілою для кінцевого користувача.

### **4.3 Тестування системи (методика, валідація і аналіз ефективності)**

Метою тестування було перевірити коректність функціонування вебсистеми, достовірність обчислювальних результатів і зручність використання створеного прототипу. Методика тестування включала три групи перевірок: функціональну, аналітичну та інтеграційну. Функціональна перевірка оцінювала, чи коректно створюються сесії, зберігаються альтернативи, експерти та оцінки, чи працює навігація між сторінками та генерація PDF-звіту. Аналітична перевірка мала підтвердити правильність розрахунку п'яти методів і статистичних показників. Інтеграційна перевірка контролювала узгоджену роботу вебінтерфейсу, серверної логіки, бази даних і обчислювальних модулів.

Для демонстраційного експерименту було сформовано тестову сесію 'Вибір платформи цифрового врядування', де чотири альтернативи оцінювалися п'ятьма експертами в два раунди. Такий приклад є достатньо складним, щоб перевірити роботу всіх алгоритмів, але водночас залишається інтерпретованим у пояснювальній записці. На першому раунді експерти мали більш розбіжні оцінки, тоді як на другому – після узагальненого зворотного зв'язку – їх позиції частково зблизилися (табл. 4.1). Саме це дозволило оцінити не лише підсумковий рейтинг, а й динаміку коефіцієнта узгодженості.

Таблиця 4.1 – Оцінки експертів у другому раунді

| Експерт   | Вага | Альтернатива<br>А | Альтернатива<br>В | Альтернатива<br>С | Альтернатива<br>D |
|-----------|------|-------------------|-------------------|-------------------|-------------------|
| Експерт 1 | 0.95 | 7                 | 8                 | 8                 | 10                |
| Експерт 2 | 0.90 | 7                 | 8                 | 9                 | 9                 |
| Експерт 3 | 0.85 | 7                 | 8                 | 8                 | 9                 |
| Експерт 4 | 0.80 | 7                 | 8                 | 8                 | 9                 |
| Експерт 5 | 0.88 | 7                 | 9                 | 8                 | 9                 |

Під час функціонального тестування було перевірено такі сценарії: створення нової сесії з коректними даними; спроба створення сесії з неповним набором альтернатив або ваг; відкриття сторінки сесії; перегляд таблиці оцінок; генерація PDF-звіту. У всіх основних сценаріях система відпрацювала без помилок. Додатково перевірено стійкість до некоректного введення, зокрема до різної кількості експертів і коефіцієнтів компетентності. У таких випадках користувач отримував повідомлення про помилку, а некоректні дані не потрапляли до бази.

Аналітичне тестування виконувалося шляхом зіставлення результатів, отриманих системою, з ручними перевірочними розрахунками на невеликому наборі даних. Це дозволило переконатися, що логіка зважування, ранжування та попарних порівнянь реалізована коректно. Крім того, тестування показало, що всі модулі мають узгоджений формат повернення результату, що істотно спрощує інтеграцію в шаблони інтерфейсу і в підсистему звітності.

Крім зазначених перевірок, тестування мало ще одну мету – переконатися, що система є придатною для демонстрації на захисті. Для цього оцінювалися простота запуску, стабільність відображення сторінок, наявність зрозумілих демонстраційних даних і відсутність кроків, які потребують складного налаштування. Прототип показав себе добре: після запуску створюється тестова сесія, а всі необхідні результати доступні одразу через вебінтерфейс. Це важливий практичний результат, оскільки програмна реалізація має не лише існувати, а й бути реальною придатною до

показу комісії.

Результати експериментального тестування показали, що за всіма п'ятьма методами переможцем стала 'Альтернатива D – інтегрована'. Це свідчить про достатню стійкість результату на обраному наборі даних. Однак числові показники різних методів відрізнялися. Зважений середній бал показав високу абсолютну перевагу цієї альтернативи, метод Борда підтвердив її лідерство у підсумкових рангах, метод Кондорсе зафіксував перемогу в усіх попарних зіставленнях, Делфі продемонстрував стабільність після двох раундів, а метод парних порівнянь сформував вектор пріоритетів, де ця альтернатива також посіла перше місце (табл. 4.2).

Таблиця 4.2 – Порівняння результатів методів

| Метод                  | Лідер          | Показник лідера |
|------------------------|----------------|-----------------|
| Зважений середній бал  | Альтернатива D | 9.217           |
| Метод Борда            | Альтернатива D | 2.594           |
| Метод Кондорсе         | Альтернатива D | 3.000           |
| Метод Делфі            | Альтернатива D | 9.217           |
| Метод парних порівнянь | Альтернатива D | 0.283           |

Показник Kendall W для першого раунду становив 0.472, що відповідає помірній узгодженості думок експертів. Після другого раунду значення зросло до 0.744, тобто рівень узгодженості став суттєво вищим (рис. 4.3). Це підтверджує доцільність багатораундового режиму та показує, що Делфі-подібний механізм анонімного зворотного зв'язку є корисним для зближення позицій експертів. Для практичного користувача така динаміка означає, що другий раунд не був зайвим, а справді підвищив якість підсумкового рішення.

Аналіз коефіцієнтів варіації показав, що найменш спірною альтернативою була 'Альтернатива A – базова', для якої всі експерти у другому раунді дали однаковий бал. Для решти альтернатив коефіцієнт варіації залишався низьким і не перевищував 0.05,

що також свідчить про відносну стабільність оцінок. Ентропійні показники для всіх альтернатив були близькими, що підтверджує відсутність хаотичного розподілу оцінок. У сукупності ці метрики свідчать, що набір даних є придатним для побудови змістовного колективного рейтингу.

Крім числових результатів, тестування продемонструвало працездатність візуальної частини системи. Користувач отримує таблицю оцінок, картки з результатами методів, блок ключових індикаторів і можливість згенерувати звіт. Таким чином, система не лише виконує математичні розрахунки, а й подає їх у формі, зручній для практичного сприйняття. Саме ця інтеграція аналітики та представлення результатів є сильною стороною розробленого прототипу.

На рисунку 4.2 показаний фрагмент інтерфейсу, який бачить користувач після оголошення результатів.

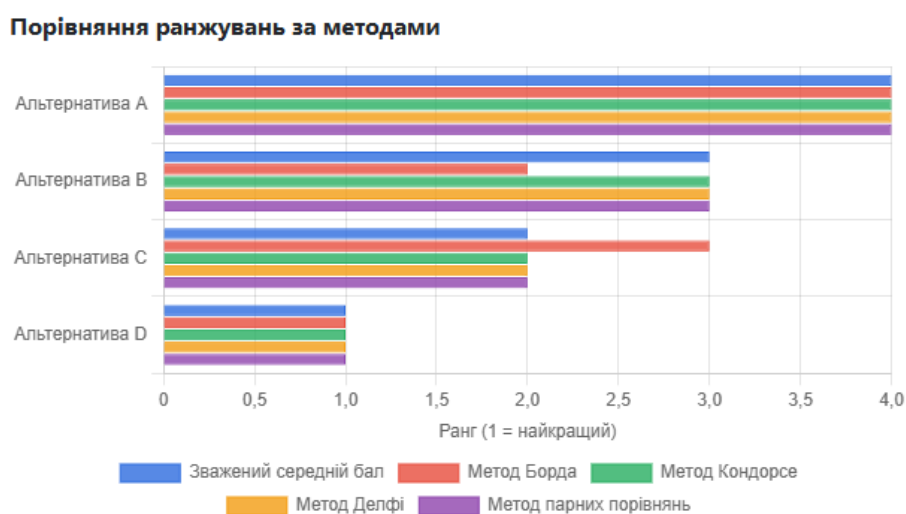


Рисунок 4.2 – Порівняння результатів методів

Ці результати мають і методологічне значення. Вони показують, що застосування декількох методів не є надмірністю, а навпаки створює підстави для більш надійного висновку. У даному прикладі всі методи підтримали одну альтернативу, але під різними кутами: через абсолютні бали, через рангові позиції, через попарні перемоги, через матричні пріоритети та через ітераційний консенсус.

Саме така багатостороння перевірка і становить головну перевагу створеної вебсистеми над простими засобами збору відповідей.

Користувач може навести на кожну з альтернатив і побачити результати, які отримали, використовуючи всі методи експертного оцінювання.



Рисунок 4.3 – Динаміка коефіцієнта Kendall W

Результат зростання коефіцієнта Kendall W оформлено у вигляді графіка, які чітко показує як змінювалась загальна думка експертів.

Ефективність запропонованої вебсистеми доцільно оцінювати у трьох аспектах: функціональному, алгоритмічному та організаційному. Функціональна ефективність полягає у тому, що система автоматизує повний цикл проведення експертної сесії. Якщо без вебсистеми користувачеві довелося б окремо готувати форми, вручну зводити оцінки, переносити дані до таблиць і самостійно обчислювати показники, то в прототипі всі ці етапи виконуються в єдиному середовищі. Це суттєво зменшує часові витрати та ймовірність механічних помилок.

Алгоритмічна ефективність проявляється у можливості паралельно запускати декілька методів і порівнювати результати на одному наборі даних. Практична цінність цього полягає в тому, що організатор не залежить від одного математичного підходу й може перевірити, наскільки стійким є підсумковий вибір. Якщо всі методи демонструють однакове лідерство, довіра до результату зростає. Якщо ж результати

розходяться, це сигналізує про необхідність додаткового аналізу, перегляду критеріїв або запуску нового раунду. Таким чином, система підвищує не лише швидкість, а й якість обґрунтування рішення.

Організаційна ефективність пов'язана з прозорістю процесу. Збереження історії оцінювання, наявність звіту, відображення коефіцієнта узгодженості та чітке розділення ролей роблять процедуру більш контрольованою та відтворюваною. Це особливо важливо для конкурсних, освітніх та управлінських сценаріїв, де потрібно не лише отримати результат, а й мати можливість пояснити, чому саме він був обраний. У такому сенсі створена система виконує важливу функцію цифрової формалізації групового вибору.

Разом з тим проведений аналіз дозволяє вказати напрями подальшого розвитку. До них належать: розширення набору методів, додавання повноцінної автентифікації, підтримка критеріїв оцінювання, асинхронна взаємодія в реальному часі, використання зовнішньої СУБД, розгортання в хмарному середовищі та інтеграція більш розвинених механізмів візуалізації. Незважаючи на ці перспективи, уже у поточному вигляді система демонструє достатній рівень завершеності для кваліфікаційної роботи бакалавра й підтверджує досягнення поставленої мети.

Якщо порівняти створений прототип із ручною процедурою проведення експертної сесії, можна констатувати помітний вигрaш у керованості процесом. Система стандартизує введення даних, зменшує потребу в ручних перерахунках, пришвидшує формування звіту та забезпечує накопичення історії. Це особливо важливо в тих випадках, коли сесії проводяться регулярно й організатору потрібно мати єдиний інструмент для повторюваних процедур. Отже, ефективність системи проявляється не лише в швидкості обчислення, а й у підвищенні зрілості самого процесу прийняття рішень.

# ВИСНОВКИ

У бакалаврській роботі розв'язано актуальну задачу розробки вебсистеми підтримки колективного прийняття рішень на основі методів експертного оцінювання.

Отримано наступні результати:

1. Проведений аналіз предметної області показав, що сучасні організації потребують інструментів, які дозволяють формалізувати експертні судження, автоматично агрегувати оцінки та оцінювати рівень узгодженості між учасниками. На основі цього було обґрунтовано доцільність створення спеціалізованої вебсистеми, що поєднує організаційну, алгоритмічну та аналітичну підтримку групового вибору.

2. У ході роботи досліджено основні класи методів експертного оцінювання та обрано п'ять методів для реалізації: зважений середній бал, метод Борда, метод Кондорсе, метод Делфі та метод парних порівнянь. Такий вибір дав змогу представити різні підходи до агрегування оцінок і забезпечити змістовне порівняння результатів. Було спроектовано архітектуру вебсистеми, визначено функціональні й нефункціональні вимоги, побудовано UML-моделі та реляційну модель бази даних.

3. Практичним результатом роботи стала програмна реалізація прототипу вебсистеми з використанням Python, Flask, SQLite та HTML/CSS. Створений застосунок забезпечує створення сесій оцінювання, введення альтернатив і даних про експертів, збереження оцінок по раундах, запуск п'яти методів, обчислення коефіцієнта конкордації Кендалла, коефіцієнта варіації та ентропії, а також формування PDF-звіту. Отже, вимога наявності програмної реалізації повністю виконана.

4. Результати тестування підтвердили працездатність системи. На демонстраційному наборі даних усі методи виявили ту саму лідируючу альтернативу, а коефіцієнт Kendall W зріс з 0.472 у першому раунді до 0.744 у другому. Це свідчить про доцільність багатораундового режиму й підтверджує, що розроблений прототип

може не лише автоматично обчислювати рейтинги, а й підтримувати процес досягнення експертного консенсусу.

Таким чином, поставлену мету досягнуто, а всі основні задачі дослідження виконано. Запропонована вебсистема має практичну цінність як навчально-прикладний інструмент для організації експертних процедур у різних предметних областях. Подальший розвиток роботи може бути пов'язаний з розширенням набору методів, введенням багатокритеріального оцінювання, удосконаленням механізмів безпеки та розгортанням системи у повноцінній клієнт-серверній інфраструктурі.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 Turban E., Sharda R., Delen D. Decision Support and Business Intelligence Systems. 10th ed. USA, Upper Saddle River : Pearson, 2011.
- 2 Power D. J. Decision Support Systems: Concepts and Resources for Managers. USA, Westport : Greenwood Publishing Group, 2002.
- 3 Keen P. G. W., Scott Morton M. S. Decision Support Systems: An Organizational Perspective. USA, Reading : Addison-Wesley, 1978.
- 4 Linstone H. A., Turoff M. The Delphi Method: Techniques and Applications. USA, Reading : Addison-Wesley, 1975.
- 5 Saaty T. L. The Analytic Hierarchy Process. USA, New York : McGraw-Hill, 1980.
- 6 Kendall M. G., Smith B. B. The Problem of m Rankings. The Annals of Mathematical Statistics. 1939. Vol. 10. No. 3. P. 275–287.
- 7 Borda J.-C. Memoire sur les elections au scrutin. Histoire de l'Academie Royale des Sciences. 1781.
- 8 Condorcet M. Essay on the Application of Analysis to the Probability of Majority Decisions. France, Paris, 1785.
- 9 Hwang C. L., Yoon K. Multiple Attribute Decision Making: Methods and Applications. Germany, Berlin : Springer, 1981.
- 10 Roy B. Multicriteria Methodology for Decision Aiding. Netherlands, Dordrecht : Kluwer Academic Publishers, 1996.
- 11 Belton V., Stewart T. Multiple Criteria Decision Analysis: An Integrated Approach. Netherlands, Dordrecht : Kluwer Academic Publishers, 2002.
- 12 Goodwin P., Wright G. Decision Analysis for Management Judgment. 5th ed. USA, Hoboken : Wiley, 2014.
- 13 Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. USA, New York : McGraw-Hill, 2019.

- 14 Sommerville I. Software Engineering. 10th ed. USA, Hoboken : Pearson, 2016.
- 15 Elmasri R., Navathe S. Fundamentals of Database Systems. 7th ed. USA, Hoboken : Pearson, 2017.
- 16 Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. USA, Sebastopol : O'Reilly Media, 2018.
- 17 Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. USA, Boston : Addison-Wesley, 2004.
- 18 ISO/IEC 25010:2011. Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. Switzerland, Geneva : ISO, 2011.
- 19 Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma et al. USA, Boston : Addison-Wesley, 1994.
- 20 Introduction to Algorithms / T. H. Cormen et al. 3rd ed. USA, Cambridge : MIT Press, 2009.
- 21 Taha H. A. Operations Research: An Introduction. 10th ed. USA, Hoboken : Pearson, 2017.

## ДОДАТОК А

### Повний текст файлу services/methods/ahp.py

```
from __future__ import annotations

from app.utils.math_utils import normalize, safe_divide

METHOD_NAME = "Метод парних порівнянь (АНР)"
METHOD_CLASS = "priority"


def _pairwise_scale(v1: float, v2: float) -> float:
    if v2 == 0:
        return 9.0
    ratio = v1 / v2
    if ratio >= 1:
        return min(9.0, max(1.0, ratio))
    return 1 / min(9.0, max(1.0, 1 / ratio))


def calculate(alternatives: list, expert_scores: list, competency_weights: list) -> dict:
    """
    AHP-like method: build pairwise comparison matrix from weighted scores,
    then derive priorities via eigenvector approximation.
    Args:
        alternatives: list of alternative names
        expert_scores: list of dicts {alt_name: score} per expert
    """
```

competency\_weights: normalized weights summing to 1.0

Returns:

```
    {"ranking": [...], "scores": {...}, "details": {...}}
    """
from collections import defaultdict

# Compute weighted aggregate scores first
totals = defaultdict(float)
denom = sum(competency_weights) or 1.0
for score_map, weight in zip(expert_scores, competency_weights):
    for alt in alternatives:
        totals[alt] += score_map.get(alt, 0) * weight
for alt in totals:
    totals[alt] /= denom

n = len(alternatives)
if n == 0:
    return {"ranking": [], "scores": {}, "details": {}}

matrix = [[1.0] * n for _ in range(n)]
for i in range(n):
    for j in range(n):
        if i != j:
            matrix[i][j] = _pairwise_scale(
                totals.get(alternatives[i], 0),
                totals.get(alternatives[j], 0),
            )

col_sums = [sum(matrix[i][j] for i in range(n)) for j in range(n)]
normalized = [
```

```

    [matrix[i][j] / col_sums[j] if col_sums[j] else 0 for j in range(n)]
    for i in range(n)
]
priorities = {alternatives[i]: sum(normalized[i]) / n for i in range(n)}

# Consistency ratio (simplified)
weighted_sum = [
    sum(matrix[i][j] * priorities[alternatives[j]] for j in range(n))
    for i in range(n)
]
lambda_max = sum(
    weighted_sum[i] / priorities[alternatives[i]]
    for i in range(n)
    if priorities[alternatives[i]] > 0
) / n
ci = safe_divide(lambda_max - n, n - 1) if n > 1 else 0
ri_table = {1: 0, 2: 0, 3: 0.58, 4: 0.90, 5: 1.12, 6: 1.24, 7: 1.32, 8: 1.41, 9: 1.45}
ri = ri_table.get(n, 1.49)
cr = safe_divide(ci, ri)

ranking = sorted(alternatives, key=lambda a: -priorities.get(a, 0))
normalized_scores = normalize({a: priorities.get(a, 0) for a in alternatives})

return {
    "ranking": ranking,
    "scores": normalized_scores,
    "details": {
        "priorities": priorities,
        "lambda_max": round(lambda_max, 4),
        "ci": round(ci, 4),
    }
}

```

```

        "cr": round(cr, 4),
        "consistent": cr < 0.1,
    },
}

```

### Повний текст файлу `services/methods/borda.py`

```

from __future__ import annotations
from collections import defaultdict

```

```

from app.utils.math_utils import normalize

```

```

METHOD_NAME = "Метод Борда"

```

```

METHOD_CLASS = "ranking"

```

```

def calculate(alternatives: list, expert_scores: list, competency_weights: list) -> dict:

```

```

    """

```

```

    Borda count: alternative ranked i-th gets (n-i) points, weighted by competency.

```

```

    Args:

```

```

        alternatives: list of alternative names

```

```

        expert_scores: list of dicts {alt_name: score} per expert

```

```

        competency_weights: normalized weights summing to 1.0

```

```

    Returns:

```

```

        {"ranking": [...], "scores": {...}, "details": {...}}

```

```

    """

```

```

    n = len(alternatives)

```

```

    totals = defaultdict(float)

```

```

    for score_map, weight in zip(expert_scores, competency_weights):

```

```

        ranking = sorted(alternatives, key=lambda a: (-score_map.get(a, 0), a))

```

```

for i, alt in enumerate(ranking):
    totals[alt] += (n - i - 1) * weight

ranking = sorted(alternatives, key=lambda a: -totals.get(a, 0))
normalized = normalize({a: totals.get(a, 0) for a in alternatives})

return {
    "ranking": ranking,
    "scores": normalized,
    "details": {"borda_points": dict(totals)},
}

```

### Повний текст файлу services/methods/condorcet.py

```

from __future__ import annotations

METHOD_NAME = "Метод Кондорсе"
METHOD_CLASS = "pairwise"

def calculate(alternatives: list, expert_scores: list, competency_weights: list) -> dict:
    """
    Condorcet method: weighted pairwise wins matrix.

    Args:
        alternatives: list of alternative names
        expert_scores: list of dicts {alt_name: score} per expert
        competency_weights: normalized weights summing to 1.0

    Returns:
        {"ranking": [...], "scores": {...}, "details": {...}}
    """
    wins = {a: 0.0 for a in alternatives}

```

```
pairwise_matrix = {a: {b: 0.0 for b in alternatives} for a in alternatives}
```

```
for i, a in enumerate(alternatives):
```

```
    for b in alternatives[i + 1:]:
```

```
        score_a = score_b = 0.0
```

```
        for score_map, weight in zip(expert_scores, competency_weights):
```

```
            sa = score_map.get(a, 0)
```

```
            sb = score_map.get(b, 0)
```

```
            if sa > sb:
```

```
                score_a += weight
```

```
            elif sb > sa:
```

```
                score_b += weight
```

```
        if score_a > score_b:
```

```
            wins[a] += 1
```

```
            pairwise_matrix[a][b] = 1.0
```

```
        elif score_b > score_a:
```

```
            wins[b] += 1
```

```
            pairwise_matrix[b][a] = 1.0
```

```
        else:
```

```
            wins[a] += 0.5
```

```
            wins[b] += 0.5
```

```
            pairwise_matrix[a][b] = 0.5
```

```
            pairwise_matrix[b][a] = 0.5
```

```
max_wins = len(alternatives) - 1 or 1
```

```
ranking = sorted(alternatives, key=lambda a: -wins.get(a, 0))
```

```
normalized = {a: wins[a] / max_wins for a in alternatives}
```

```
return {
```

```
    "ranking": ranking,
```

```

"scores": normalized,
"details": {"wins": wins, "pairwise_matrix": pairwise_matrix},
}

```

### Повний текст файлу services/methods/delphi.py

```

from __future__ import annotations
from collections import defaultdict

```

```

from app.utils.math_utils import normalize, safe_divide

```

```

METHOD_NAME = "Метод Делфі"

```

```

METHOD_CLASS = "iterative"

```

```

def calculate(
    alternatives: list,
    expert_scores: list,
    competency_weights: list,
    round_history: list | None = None,
) -> dict:
    """

```

*Delphi iterative method.*

*Args:*

*alternatives: list of alternative names*

*expert\_scores: list of dicts {alt\_name: score} per expert (current round)*

*competency\_weights: normalized weights summing to 1.0*

*round\_history: optional list of previous-round expert\_scores lists*

*(same shape as expert\_scores). When provided, the most recent previous round is mixed in via:*

$score = 0.7 * current + 0.3 * previous\_round\_mean$   
*If None or empty → simple weighted average.*

*Returns:*

```
    {"ranking": [...], "scores": {...}, "details": {...}}
    """
    denom = sum(competency_weights)

    # Compute per-alternative mean of previous round (across experts) if available.
    prev_means: dict = {}
    if round_history:
        prev_scores = round_history[-1]
        if prev_scores:
            for alt in alternatives:
                vals = [sm.get(alt, 0) for sm in prev_scores]
                prev_means[alt] = safe_divide(sum(vals), len(vals))

    totals: dict = defaultdict(float)
    for score_map, weight in zip(expert_scores, competency_weights):
        for alt in alternatives:
            current = score_map.get(alt, 0)
            if prev_means:
                value = 0.7 * current + 0.3 * prev_means.get(alt, current)
            else:
                value = current
            totals[alt] += value * weight

    for alt in list(totals):
        totals[alt] = safe_divide(totals[alt], denom, default=totals[alt])
```

```

ranking = sorted(alternatives, key=lambda a: -totals.get(a, 0))
normalized = normalize({a: totals.get(a, 0) for a in alternatives})

```

```

return {
    "ranking": ranking,
    "scores": normalized,
    "details": {
        "weighted_totals": dict(totals),
        "iterative": bool(prev_means),
    },
}

```

```

def calculate_from_rounds(alternatives: list, round_map: dict, expert_data: list) -> dict:

```

```

    """

```

```

    Full Delphi with round history. Used internally by models.session_summary.

```

```

    round_map: {round_no: {expert_id: {alt: score}}}

```

```

    expert_data: [{"id": ..., "weight": ...}]

```

```

    """

```

```

    weights = {e["id"]: e["weight"] for e in expert_data}

```

```

    round_numbers = sorted(round_map)

```

```

    if not round_numbers:

```

```

        return {"ranking": [], "scores": {}, "details": {}}

```

```

    totals = defaultdict(float)

```

```

    final_round = round_numbers[-1]

```

```

    prev_round = round_numbers[-2] if len(round_numbers) > 1 else final_round

```

```

    denom = sum(weights.values()) or 1.0

```

```

    for expert_id in round_map[final_round]:

```

```

for alt in alternatives:
    current = round_map[final_round][expert_id].get(alt, 0)
    previous = round_map[prev_round][expert_id].get(alt, current)
    consensus = (current * 0.7 + previous * 0.3) * weights.get(expert_id, 1.0)
    totals[alt] += consensus

for alt in totals:
    totals[alt] /= denom

ranking = sorted(alternatives, key=lambda a: -totals.get(a, 0))
total_sum = sum(totals.values()) or 1.0
normalized = {a: totals[a] / total_sum for a in alternatives}

return {
    "ranking": ranking,
    "scores": normalized,
    "details": {"weighted_totals": dict(totals)},
}

```

#### **Повний текст файлу services/methods/weighted\_score.py**

```

from __future__ import annotations
from collections import defaultdict

from app.utils.math_utils import normalize, safe_divide

METHOD_NAME = "Зважений середній бал"
METHOD_CLASS = "scoring"

def calculate(alternatives: list, expert_scores: list, competency_weights: list) -> dict:

```

"""

*Args:*

*alternatives: list of alternative names*

*expert\_scores: list of dicts {alt\_name: score} per expert*

*competency\_weights: normalized weights summing to 1.0*

*Returns:*

*{"ranking": [...], "scores": {...}, "details": {...}}*

"""

```
totals = defaultdict(float)
```

```
denom = sum(competency_weights)
```

```
for score_map, weight in zip(expert_scores, competency_weights):
```

```
    for alt in alternatives:
```

```
        totals[alt] += score_map.get(alt, 0) * weight
```

```
for alt in list(totals):
```

```
    totals[alt] = safe_divide(totals[alt], denom, default=totals[alt])
```

```
ranking = sorted(alternatives, key=lambda a: -totals.get(a, 0))
```

```
normalized = normalize({a: totals.get(a, 0) for a in alternatives})
```

```
return {
```

```
    "ranking": ranking,
```

```
    "scores": normalized,
```

```
    "details": {"raw_scores": dict(totals)},
```

```
}
```