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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
«___» _____2026р.

Дипломна робота
на здобуття ступеня бакалавра

За освітньою програмою «Цифрові технології в енергетиці»

Спеціальності 122 «Комп'ютерні науки»

на тему: «Вебсистема підтримки прийняття управлінських рішень на основі методів багатокритеріального аналізу»

Виконав: студент 4 курсу, групи ТР-22

Борко Дмитро Васильович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: професор каф. цифрових технологій в енергетиці,

д.т.н., проф.Сліпченко Володимир Георгійович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент: завідувач каф., теплової та альтернативної енергетики

к.т.н., доц. Пешко Віталій Анатолійович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Н.контроль доцент каф. цифрових технологій в енергетиці,

д.ф Артем ГУРІН _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ — 2026

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
«___» _____2026р.

З А В Д А Н Н Я
на дипломну роботу студенту

Борку Дмитру Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Вебсистема підтримки прийняття управлінських рішень на основі методів багатокритеріального аналізу».

керівник роботи Сліпченко Володимир Георгійович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «28» травня 2026р. № 1992-с

2. Термін подання роботи студентом 04.06.2026р.

3. Вихідні дані до роботи: мова програмування Python 3.12; фреймворк веб-інтерфейсу Streamlit; реляційна СУБД SQLite з доступом через ORM SQLAlchemy; бібліотеки наукових обчислень NumPy, SciPy, Pandas; бібліотека візуалізації Plotly; бібліотека генерації PDF FPDF2; засоби розробки — Git, Visual Studio Code.

4. Перелік питань, які потрібно розробити: 1) проаналізувати предметну область підтримки прийняття управлінських рішень, існуючі програмні рішення та обґрунтувати вибір засобів розробки; 2) систематизувати методи багатокритеріального аналізу (SAW, WPM, TOPSIS, VIKOR, PROMETHEE II, AHP) та обґрунтувати їх вибір; 3) спроєктувати архітектуру веб-системи, модель даних та програмні модулі; 4) реалізувати програмні компоненти системи: обчислювальний MCDM-модуль, репозиторії даних, веб-інтерфейс, модулі порівняння, генерації звітів та сценарного аналізу; 5) виконати тестування системи та сформулювати рекомендації щодо вибору методу.

5. Орієнтовний перелік ілюстративного матеріалу: діаграма використання (use case), архітектурна діаграма компонентів, ER-діаграма бази даних, діаграма класів модулів, діаграма послідовностей виконання розрахунків, блок-схема узагальненого MCDM-процесу, скріншоти інтерфейсу користувача.

6. Дата видачі завдання 07.02.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Аналіз предметної області та систематизація існуючих рішень підтримки прийняття рішень	07.02 – 13.02.2026	Виконано
2	Огляд та обґрунтування вибору методів MCDM (SAW, WPM, TOPSIS, VIKOR, PROMETHEE II, AHP)	14.02 – 20.02.2026	Виконано
3	Обґрунтування засобів розробки та проєктування архітектури веб-системи	21.02 – 27.02.2026	Виконано
4	Розробка моделі даних та програмних модулів	28.02 – 13.03.2026	Виконано
5	Реалізація обчислювального MCDM-модуля та репозиторіїв даних	14.03.2026 – 03.04.2026	Виконано
6	Реалізація веб-інтерфейсу, модулів порівняння та сценарного аналізу	04.04 – 24.04.2026	Виконано
7	Тестування системи та порівняльний аналіз методів	25.04 – 08.05.2026	Виконано
8	Оформлення пояснювальної записки	09.05 – 31.05.2026	Виконано
9	Захист програмного продукту	14.05.2026	Виконано
10	Передзахист	04.06.2026	Виконано
11	Захист	19.06.2026	Виконано

Студент _____ Дмитро БОРКО
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Володимир СЛІПЧЕНКО
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 48 сторінках, містить 19 ілюстрацій, 12 таблиць, 1 додаток, 20 джерел в переліку посилань.

Метою роботи є розробка вебсистеми підтримки прийняття управлінських рішень на основі методів багатокритеріального аналізу, що реалізує MCDM-методи з різних класів, забезпечує їх порівняльний аналіз, сценарне моделювання та зручний інтерактивний інтерфейс.

Основний зміст роботи включає аналіз предметної області та існуючих систем підтримки прийняття рішень, систематизацію методів багатокритеріального аналізу, проєктування архітектури та моделі даних, програмну реалізацію обчислювальних модулів і веб-інтерфейсу, а також тестування й порівняльний аналіз методів.

Методи та засоби: методи багатокритеріального аналізу (SAW, WPM, TOPSIS, VIKOR, PROMETHEE II, AHP), метод аналізу ієрархій Сааті для визначення ваг, коефіцієнт рангової кореляції Спірмена; мова Python, фреймворк Streamlit, СУБД SQLite з ORM SQLAlchemy, бібліотеки NumPy, SciPy, Pandas, Plotly, FPDF2.

Результат: програмний застосунок — вебсистема підтримки прийняття управлінських рішень, що реалізує шість MCDM-методів, порівняння результатів за коефіцієнтом Спірмена, What-if та Sensitivity аналіз, генерацію PDF-звіту.

Розроблений додаток може бути впроваджений на підприємствах та в організаціях для обґрунтування управлінських рішень у задачах вибору постачальників, оцінювання проєктів, кадрових та інвестиційних рішень.

Ключові слова: БАГАТОКРИТЕРІАЛЬНИЙ АНАЛІЗ, ПІДТРИМКА ПРИЙНЯТТЯ РІШЕНЬ, MCDM, TOPSIS, VIKOR, PROMETHEE, AHP, ВЕБ-ЗАСТОСУНОК, STREAMLIT.

ABSTRACT

The diploma project consists of 48 pages and includes 19 illustrations, 12 tables, 1 appendix, and 20 references.

The aim of the study is to develop a web-based decision support system based on multi-criteria decision-making methods that implements at least six MCDM methods from different classes, provides their comparative analysis, scenario modelling and a convenient interactive interface.

The main content of the thesis includes the analysis of the subject area and existing decision support systems, the systematization of multi-criteria analysis methods, the design of the architecture and data model, the software implementation of the computational modules and the web interface, as well as testing and comparative analysis of the methods.

Methods and tools: multi-criteria decision-making methods (SAW, WPM, TOPSIS, VIKOR, PROMETHEE II, AHP), the Saaty analytic hierarchy process for weight determination, the Spearman rank correlation coefficient; the Python language, the Streamlit framework, the SQLite database with the SQLAlchemy ORM, the NumPy, SciPy, Pandas, Plotly and FPDF2 libraries.

Result: a software application — a web-based decision support system that implements six MCDM methods, comparison of results by the Spearman coefficient, What-if and Sensitivity analysis, and PDF report generation.

The developed application can be implemented at enterprises and organizations to justify managerial decisions in supplier selection, project evaluation, personnel and investment decision tasks.

Keywords: MULTI-CRITERIA ANALYSIS, DECISION SUPPORT, MCDM, TOPSIS, VIKOR, PROMETHEE, AHP, WEB APPLICATION, STREAMLIT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

MCDM (Multi-Criteria Decision Making) — багатокритеріальне прийняття рішень.

СППР — система підтримки прийняття рішень.

ОПР — особа, що приймає рішення.

SAW (Simple Additive Weighting) — метод простого зваженого додавання.

WPM — Weighted Product Method) — метод зваженого добутку.

TOPSIS — Technique for Order of Preference by Similarity to Ideal Solution.

VIKOR — VlseKriterijumska Optimizacija I Kompromisno Resenje.

PROMETHEE — Preference Ranking Organization Method for Enrichment Evaluations.

AHP (Analytic Hierarchy Process) — метод аналізу ієрархій.

PIS (Positive Ideal Solution) — позитивне ідеальне рішення.

NIS (Negative Ideal Solution) — негативне ідеальне рішення.

CR (Consistency Ratio) — відношення узгодженості суджень.

CI (Consistency Index) — індекс узгодженості.

ORM (Object-Relational Mapping) — об’єктно-реляційне відображення.

SQL (Structured Query Language) — мова структурованих запитів.

CRUD (Create, Read, Update, Delete) — базові операції з даними.

UI (User Interface) — інтерфейс користувача.

API (Application Programming Interface) — програмний інтерфейс.

PDF (Portable Document Format) — портативний формат документів.

PBKDF2 (Password-Based Key Derivation Function 2) — функція формування ключа з пароля.

ВСТУП

У сучасних умовах розвитку економіки та управління процес прийняття обґрунтованих рішень набуває особливої важливості. Зростання складності організаційного середовища, глобалізація та динамічні зміни ринкових умов вимагають від керівників і аналітиків застосування науково обґрунтованих методів оброблення та оцінювання великих масивів різнорідної інформації. Значна частина управлінських задач належить до класу багатокритеріальних задач прийняття рішень (Multi-Criteria Decision Making, MCDM), для яких характерна наявність кількох, як правило, суперечливих критеріїв оцінювання, множини альтернативних варіантів та необхідність одночасно враховувати якісні та кількісні показники.

На сьогодні розроблено значну кількість методів багатокритеріального аналізу, які класифікують за різними ознаками: методи зваженого оцінювання (SAW, WPM), методи відстані до ідеального рішення (TOPSIS, VIKOR), методи переважання (PROMETHEE, ELECTRE), ієрархічні методи (AHP, ANP). Кожен клас має власні математичні особливості, переваги та обмеження, а тому жоден метод не є універсальним. Це обумовлює потребу в інструменті, який поєднує методи різних класів та дає змогу порівнювати їх результати.

Актуальність теми. Наявні програмні рішення для багатокритеріального аналізу або охоплюють лише окремий клас методів, або є комерційними чи монолітними десктопними застосунками без зручного веб-доступу та вбудованих засобів порівняльного аналізу. Відсутність єдиного відкритого веб-інструменту, що поєднує кілька методів MCDM з різних класів, забезпечує порівняння результатів і сценарне моделювання, підтверджує актуальність розробки такої системи.

Мета роботи — розробка вебсистеми підтримки прийняття управлінських рішень на основі методів багатокритеріального аналізу, що реалізує MCDM-методів з різних класів, забезпечує їх порівняльний аналіз, сценарне моделювання та зрозумілий інтерактивний інтерфейс.

Для досягнення поставленої мети визначено такі основні завдання:

1) проаналізувати методи підтримки прийняття управлінських рішень, дослідити існуючі програмні рішення та обґрунтувати вибір засобів розробки;

2) систематизувати методи багатокритеріального аналізу, навести їхні математичні моделі та обґрунтувати вибір методів з різних класів;

3) спроєктувати архітектуру вебсистеми, модель даних та структуру програмних модулів;

4) реалізувати програмні компоненти системи: обчислювальний MCDM-модуль, репозиторії доступу до даних, веб-інтерфейс, модулі порівняння результатів, генерації звітів та сценарного аналізу;

5) виконати тестування системи та сформулювати практичні рекомендації щодо вибору методу залежно від характеристик задачі.

Практична цінність роботи полягає у створенні готового до використання веб-інструменту, який стане у пригоді підприємствам, інжиніринговим та консалтинговим компаніям, органам управління для обґрунтування рішень у задачах вибору постачальників, оцінювання проєктів, кадрових та інвестиційних рішень.

Структура та обсяг роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і одного додатку. У першому розділі проаналізовано предметну область, існуючі рішення та обґрунтовано вибір засобів розробки. У другому розділі систематизовано методи багатокритеріального аналізу та обґрунтовано їх вибір. У третьому розділі описано проєктування й програмну реалізацію системи. У четвертому розділі наведено результати тестування та порівняльний аналіз методів. Роботу оформлено відповідно до вимог чинних стандартів [2, 3]. Загальний обсяг основного тексту — 48 сторінок; робота містить 19 рисунків та 11 таблиць; список використаних джерел налічує 20 найменувань.

1 АНАЛІЗ ЗАДАЧІ РОЗРОБКИ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ УПРАВЛІНСЬКИХ РІШЕНЬ ТА ЗАСОБІВ РЕАЛІЗАЦІЇ

Цей розділ окреслює сферу застосування системи та логіку подальшого викладу. Спершу формалізовано постановку прикладної задачі: визначено вхідні та вихідні дані, функціональні можливості системи. Далі розглянуто наявні програмні рішення підтримки прийняття рішень та їхні обмеження. У заключних підрозділах обґрунтовано вибір комплексу засобів розробки серверної логіки, веб-інтерфейсу та сховища даних, який забезпечує реалізацію всіх поставлених завдань.

1.1 Постановка прикладної задачі

Особа, що приймає рішення (ОПР), у багатьох управлінських ситуаціях стикається з потребою обрати найкращий варіант із кількох альтернатив, які оцінюються за множиною різнорідних і часто суперечливих критеріїв. Прикладами таких задач є вибір постачальника обладнання, оцінювання інвестиційних проєктів, добір персоналу, вибір технології чи стратегії розвитку. Складність полягає в тому, що критерії мають різну природу, різні одиниці вимірювання та протилежні напрями оптимальності: одні потрібно максимізувати (наприклад, якість), інші — мінімізувати (наприклад, вартість).

Формально задача багатокритеріального вибору ставиться так. Задано множину альтернатив $A = \{A_1, A_2, \ldots, A_m\}$ та множину критеріїв $C = \{C_1, C_2, \ldots, C_n\}$. Кожній парі «альтернатива — критерій» відповідає числова оцінка x_{ij}, що утворює матрицю рішень X розміром m×n. Кожен критерій має тип: «benefit» (більше значення краще) або «cost» (менше значення краще), а також ваговий

коефіцієнт w_j, що відображає його відносну важливість, причому $\Sigma w_j = 1$. Необхідно отримати впорядкування (ранжування) альтернатив від найкращої до найгіршої з урахуванням ваг та типів критеріїв.

Прикладна задача полягає в розробці програмного засобу, який автоматизує всі етапи цього процесу та позбавляє ОПР від розрізнених ручних розрахунків в електронних таблицях. Для цього система повинна забезпечувати:

1) формалізоване зберігання задачі: критеріїв з типами та одиницями вимірювання, альтернатив та матриці числових оцінок;

2) визначення вагових коефіцієнтів критеріїв вручну з автоматичною нормалізацією або методом парних порівнянь Сааті з перевіркою узгодженості суджень;

3) розрахунок ранжування альтернатив щонайменше шістьма методами MCDM з різних класів;

4) порівняльний аналіз результатів різних методів з кількісною оцінкою їх узгодженості;

5) сценарне моделювання «що-якщо» та аналіз чутливості ранжування до зміни ваг критеріїв;

6) генерацію підсумкового звіту та збереження історії прийнятих рішень;

7) розмежування доступу користувачів та ізоляцію їхніх задач.

Вхідними даними системи є: назва та опис задачі; перелік критеріїв із зазначенням типу (benefit/cost) та одиниці вимірювання; перелік альтернатив; матриця числових оцінок альтернатив за критеріями; вагові коефіцієнти критеріїв.

Вихідними даними системи є: ранжування альтернатив за кожним із реалізованих методів; інтегральні оцінки альтернатив; матриця коефіцієнтів рангової кореляції Спірмена між методами; рекомендована альтернатива; результати аналізу чутливості; PDF-звіт із підсумками розрахунків.

Постановка задачі визначає функціональне ядро системи – обчислювальні модулі MCDM [1] – та допоміжні підсистеми зберігання, візуалізації та аналізу, проєктування й реалізацію яких описано в наступних розділах.

1.2 Аналіз наявних рішень підтримки прийняття рішень

Перед розробкою власної системи проведено огляд існуючих програмних рішень, що реалізують методи багатокритеріального аналізу. Розглянуто три основні групи інструментів: універсальні електронні таблиці, спеціалізовані комерційні MCDM-додатки та відкриті програмні бібліотеки.

Найпоширенішим інструментом для оцінювання альтернатив залишаються електронні таблиці (Microsoft Excel, Google Sheets, LibreOffice Calc). Вони дозволяють реалізувати найпростіші методи зваженого оцінювання за допомогою формул, забезпечують гнучкість та не потребують спеціальних навичок програмування. Проте таблиці мають суттєві обмеження: відсутність вбудованих складних методів (TOPSIS, VIKOR, PROMETHEE, AHP), необхідність ручного переналаштування формул при зміні кількості критеріїв чи альтернатив, відсутність перевірки узгодженості суджень, відсутність структурованого зберігання історії рішень та інструментів порівняльного аналізу методів. Кожна нова задача фактично потребує побудови нового аркуша «з нуля».

Також існують спеціалізовані MCDM-додатки. До цієї групи належать комерційні десктопні та веб-програми: Expert Choice і SuperDecisions (реалізують метод аналізу ієрархій AHP), Visual PROMETHEE (методи родини PROMETHEE), 1000minds, Decision Lens.

Ці продукти забезпечують зручний графічний інтерфейс та коректну реалізацію відповідних методів, проте здебільшого зосереджені на одному класі методів, є комерційними з обмеженим або платним доступом, а деякі — монолітними десктопними застосунками без можливості веб-доступу та спільної роботи. Це ускладнює порівняння результатів різних класів методів у єдиному середовищі.

Існують відкриті програмні бібліотеки для мови Python, що реалізують методи MCDM: pymcdm (TOPSIS, VIKOR, SPOTIS та інші) [4], pyDecision (ELECTRE, PROMETHEE, AHP). Вони охоплюють широкий спектр методів та є

безкоштовними, проте призначені насамперед для програмістів і дослідників: вимагають написання коду, не мають готового графічного веб-інтерфейсу, засобів зберігання задач, порівняльного аналізу та звітності для кінцевого користувача-управлінця.

Узагальнення результатів огляду наведено в таблиці 1.1. Порівняння виконано за ключовими характеристиками, що визначають придатність інструмента для практичного багатокритеріального аналізу управлінських рішень.

Таблиця 1.1 — Порівняння наявних рішень підтримки прийняття рішень

Характеристика	**Електронні таблиці**	**Комерційні MCDM-додатки**	**Відкриті бібліотеки**	**Розроблювана система**
Кількість класів методів	1 (зважені)	1–2	декілька	4 класи, 6 методів
Веб-доступ без встановлення	частково	частково	ні	так
Порівняння методів (кореляція)	ні	ні	ні	так (Спірмен)
Перевірка узгодженості (AHP)	ні	так	частково	так (CR≤0.1)
Сценарний / What-if аналіз	вручну	частково	ні	так
Зберігання задач та історії	ні	так	ні	так
Вартість	низька	висока	безкоштовно	безкоштовно

Проведений аналіз показав, що жодне з наявних рішень одночасно не поєднує: підтримку кількох класів методів MCDM, зручний веб-інтерфейс без встановлення, вбудований порівняльний аналіз методів та сценарне моделювання у відкритому безкоштовному продукті. Саме цю нішу заповнює розроблювана у цій роботі вебсистема, яка реалізує шість методів з чотирьох класів, інструменти їх

порівняння за коефіцієнтом Спірмена, What-if та Sensitivity аналіз і структуроване зберігання задач у багатокористувацькому режимі.

1.3 Обґрунтування вибору засобів розробки

Для реалізації вебсистеми підтримки прийняття управлінських рішень обрано технологічний стек, що забезпечує швидку розробку, коректну реалізацію математичного апарату методів MCDM та зручний інтерактивний веб-інтерфейс без потреби в окремому фронтенд-стеку. Нижче обґрунтовано вибір кожного компонента.

1.3.1 Мова програмування Python

Основною мовою програмування обрано Python 3.12 [5]. Вибір зумовлений високою читабельністю коду, багатою екосистемою науково-обчислювальних бібліотек (NumPy, SciPy, Pandas), широким використанням у сфері аналізу даних, а також активною підтримкою спільноти та якісною документацією. Python дозволяє лаконічно реалізувати матричні операції, що становлять основу методів багатокритеріального аналізу, та інтегрувати обчислювальну логіку з веб-інтерфейсом у межах єдиного застосунку.

1.3.2 Фреймворк вебінтерфейсу Streamlit

Для створення веб-інтерфейсу застосовано фреймворк Streamlit [6]. Він забезпечує швидку розробку інтерактивних веб-застосунків мовою Python без написання HTML, CSS та JavaScript. Ключові переваги Streamlit: декларативний API, автоматичне реактивне оновлення стану сторінки при зміні вхідних даних, вбудовані компоненти для таблиць (st.data_editor), форм, кнопок, вкладок та графіків. Застосунок запускається однією командою streamlit run main.py на порту 8501 і взаємодіє з браузером через WebSocket-протокол, який Streamlit обслуговує

автоматично. Це суттєво скорочує час розробки та супроводу порівняно з класичним поділом на окремі клієнтську та серверну частини.

1.3.3 Бібліотеки оброблення даних та обчислень

Для роботи з даними та реалізації математичного апарату використано такі бібліотеки:

- NumPy [7] - векторні та матричні обчислення методів MCDM (функції np.power, np.prod, np.outer, операції з масивами);

- SciPy (scipy.stats.spearmanr) [8] - розрахунок коефіцієнтів рангової кореляції Спірмена між результатами різних методів;

- Pandas [9] - робота з матрицею оцінок у форматі DataFrame, нормалізація та pivot-перетворення при відновленні матриці рішень з реляційної таблиці evaluations;

- Plotly (plotly.express, plotly.graph_objects) [10] - побудова інтерактивних візуалізацій: стовпчастої діаграми, радарної діаграми та теплової карти кореляцій рангів.

1.3.4 Сховище даних: SQLite та SQLAlchemy

Зберігання даних реалізовано за допомогою вбудованої реляційної СУБД SQLite [11] з доступом через ORM-бібліотеку SQLAlchemy [12].

SQLite обрано з таких міркувань: не потребує окремого сервера баз даних, вся база зберігається в єдиному файлі mcdm_decisions.db, повністю підтримує SQL та ACID-транзакції й ідеально підходить для навчально-прикладного застосунку.

SQLAlchemy надає уніфікований інтерфейс виконання SQL-запитів та забезпечує безпечне параметризоване виконання команд, що унеможливлює SQL-ін’єкції. За потреби масштабування заміна SQLite на серверну СУБД (наприклад, PostgreSQL) потребує змін лише у шарі репозиторіїв без правок бізнес-логіки.

1.3.5 Генерація звітів та захист даних

Для формування звітів у форматі PDF застосовано бібліотеку FPDF2 [13], яка дозволяє програмно створювати структуровані документи з таблицями ранжувань і рекомендаціями без залежності від зовнішніх сервісів та з підтримкою кириличних шрифтів.

Для захисту облікових даних користувачів реалізовано хешування паролів за алгоритмом PBKDF2-HMAC-SHA256 зі 100 000 ітерацій та 16-байтною криптографічною сіллю; хеш зберігається у форматі «salt:hash». Користувацький інтерфейс стилізовано у темній колірній схемі через CSS-ін'єкцію.

Узагальнений перелік обраного технологічного стека наведено в таблиці 1.2.

Таблиця 1.2 — Технологічний стек системи

Технологія / бібліотека	**Версія**	**Призначення в системі**
Python	3.12	Основна мова програмування
Streamlit	≥1.32	Веб-фреймворк, UI та WebSocket-сервер (порт 8501)
SQLite	вбудована	Реляційна база даних (файл mcdm_decisions.db)
SQLAlchemy	≥2.0	ORM та SQL-інтерфейс до SQLite
Pandas	≥2.0	DataFrame, pivot, нормалізація матриці рішень
NumPy	≥1.26	Векторні та матричні обчислення MCDM
SciPy	≥1.12	spearmanr — кореляція рангів між методами
Plotly	≥5.18	Bar Chart, Radar Chart, Heatmap кореляцій
FPDF2	≥2.7	Генерація PDF-звіту

Таким чином, обраний набір технологій повністю покриває потреби розроблюваної системи та забезпечує її надійну й ефективну роботу.

2 МЕТОДИ БАГАТОКРИТЕРІАЛЬНОГО АНАЛІЗУ

У цьому розділі систематизовано методи багатокритеріального аналізу, наведено їхню класифікацію, математичні моделі та порівняльну характеристику. На основі аналізу обґрунтовано вибір шести методів з різних класів для реалізації у системі. Кожен метод розглядається з акцентом на математичний апарат та особливості, що визначають його застосовність.

2.1 Сутність та класифікація методів MCDM

Методи багатокритеріального аналізу (Multi-Criteria Decision Making, MCDM) призначені для оцінювання та ранжування скінченної множини альтернатив за декількома, як правило, суперечливими критеріями з урахуванням їхньої відносної важливості.

Основними компонентами будь-якої MCDM-задачі є:

- множина альтернатив $A = \{A_1, \ldots, A_m\}$;
- множина критеріїв $C = \{C_1, \ldots, C_n\}$;
- матриця оцінок $X = [x_{ij}]$ розміром $m \times n$;
- вектор вагових коефіцієнтів $W = (w_1, \ldots, w_n)$ з умовою $\Sigma w_j = 1$;
- напрям оптимальності кожного критерію (benefit — максимізація, cost — мінімізація).

Класифікація методів MCDM дозволяє систематизувати підходи за математичним апаратом та характером задач. Основні класи методів наведено нижче:

- методи зваженого оцінювання (weighted aggregation) - SAW, WPM, WSM;
- методи відстані до ідеального рішення (distance-based) - TOPSIS, VIKOR,

EDAS;

- методи переважання (outranking) - PROMETHEE, ELECTRE;

- ієрархічні методи (hierarchical) - AHP, ANP;

- методи на основі нечіткої логіки (fuzzy MCDM) та гібридні методи.

Класифікацію основних класів методів MCDM із зазначенням представників, принципу та типових сфер застосування наведено в таблиці 2.1.

Таблиця 2.1 — Класифікація основних класів методів MCDM

Клас методів	**Представники**	**Принцип**	**Сфери застосування**
Зважене оцінювання	SAW, WPM, WSM	Лінійна/мультиплікативна агрегація зважених оцінок	Швидке ранжування, чіткі критерії
Відстань до ідеалу	TOPSIS, VIKOR, EDAS	Відстань до позитивного/негативного ідеалу	Вибір постачальника, оцінка проєктів
Переважання	PROMETHEE, ELECTRE	Попарне порівняння з порогами переваги	Якісні критерії, планування ресурсів
Ієрархічні	AHP, ANP	Матриці парних порівнянь, власний вектор	Стратегічне управління, кадрові рішення
Нечіткі	Fuzzy AHP, FTOPSIS	Лінгвістичні змінні, нечіткі числа	Невизначеність, якісні критерії

Кожен клас методів MCDM має характерні переваги та обмеження, тому жоден із них не може вважатися універсальним для всіх задач прийняття рішень. Вибір конкретного методу залежить від особливостей предметної області, характеру критеріїв оцінювання, кількості альтернатив, рівня доступної інформації, наявності невизначеності, а також вимог до точності та прозорості отриманого результату. Крім того, різні методи можуть по-різному реагувати на зміну ваг критеріїв або вхідних даних, що також слід враховувати під час їх застосування. Незважаючи на відмінності в математичному апараті та механізмах

оцінювання, більшість методів MCDM реалізуються за схожою загальною процедурою. Узагальнену послідовність основних етапів багатокритеріального аналізу рішень наведено у вигляді блок-схеми на рисунку 2.1.

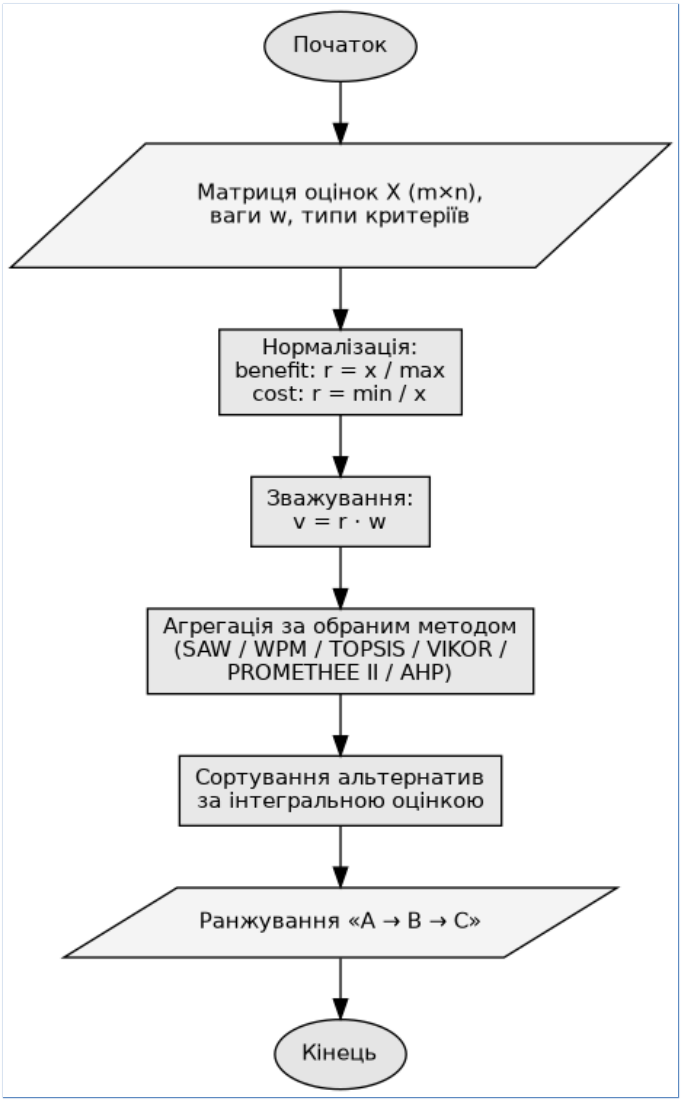


Рисунок 2.1 — Узагальнена схема процесу багатокритеріального аналізу

Таким чином, процес MCDM є послідовною процедурою, що охоплює формування альтернатив, оцінювання за критеріями та отримання підсумкового ранжування. Саме така загальна схема покладена в основу розроблюваної інформаційної системи підтримки прийняття рішень.

2.2 Методи зваженого оцінювання

Методи зваженого оцінювання належать до компенсаторних підходів: низька оцінка за одним критерієм може бути компенсована високою за іншим. Їхній математичний апарат починається з нормалізації матриці рішень для приведення критеріїв до безрозмірної шкали. У системі нормалізація реалізована функцією normalize_matrix(): для критеріїв типу benefit виконується ділення значення на максимум стовпця, для типу cost - ділення мінімуму стовпця на значення.

Метод простого зваженого додавання SAW (Simple Additive Weighting) обчислює інтегральну оцінку альтернативи як лінійну комбінацію зважених нормалізованих значень за формулою (2.1):

$$S(A_i) = \sum_j w_j \cdot \bar{r}_{ij} \quad (2.1)$$

де $\bar{r}_{ij}$ — нормалізована оцінка альтернативи A_i за критерієм C_j;

w_j — ваговий коефіцієнт критерію C_j.

Метод зваженого добутку WPM (Weighted Product Method) використовує мультиплікативну агрегацію за формулою (2.2):

$$P(A_i) = \prod_j (\bar{r}_{ij})^{w_j} \quad (2.2)$$

WPM менш чутливий до масштабу критеріїв порівняно з SAW, проте не допускає нульових нормалізованих значень. Обидва методи реалізовані у системі функціями saw() та wpm(). Основні переваги класу - математична прозорість, швидкість обчислень та легка інтерпретація;

2.3 Методи відстані до ідеального рішення

Методи цього класу ранжують альтернативи за геометричною відстанню до ідеального рішення. Метод TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) [17] - найпоширеніший представник.

Після нормалізації та зважування матриці визначаються позитивне ідеальне рішення PIS (максимуми для benefit та мінімуми для cost) і негативне ідеальне рішення NIS. Відстані альтернативи до PIS та NIS обчислюються за евклідовою метрикою (2.3) та (2.4):

$$D^+_i = \sqrt{\sum_j (v_{ij} - v_{+j})^2} \quad (2.3)$$

$$D^-_i = \sqrt{\sum_j (v_{ij} - v_{-j})^2} \quad (2.4)$$

Відносна близькість альтернативи до ідеального рішення визначається за формулою (2.5):

$$CC_i = \frac{D^-_i}{D^+_i + D^-_i} \quad (2.5)$$

де v_{ij} — зважені нормалізовані оцінки;

v^+_j, v^-_j — компоненти позитивного та негативного ідеальних рішень.

Альтернативи ранжуються за спаданням CC_i.

Метод VIKOR (VlseKriterijumska Optimizacija I Kompromisno Resenje) орієнтований на пошук компромісного рішення . Він обчислює три показники: міру групової корисності S, міру індивідуального жалю R та компромісний індекс Q (2.6):

$$Q_i = v \cdot \frac{S_i - S^*}{S^- - S^*} + (1 - v) \cdot \frac{R_i - R^*}{R^- - R^*} \quad (2.6)$$

де v — вага стратегії групового консенсусу (за замовчуванням v = 0,5);

S^*, S^-, R^*, R^- — найкращі та найгірші значення мір S і R.

Альтернативи ранжуються за зростанням Q (краща альтернатива має менший Q). У системі методи реалізовані функціями topsis() та vikor().

2.4 Методи переважання (outranking)

Outranking-методи є некомпенсаторними та ґрунтуються на попарному порівнянні альтернатив. Альтернатива A переважає альтернативу B, якщо вона є кращою за достатньою кількістю критеріїв і не суттєво гіршою за жодним. Метод

PROMETHEE II (Preference Ranking Organization Method for Enrichment Evaluations) [18] - найпоширеніший представник.

Для кожної пари альтернатив обчислюється агрегований індекс переваги (2.7):

$$\pi(A_i, A_j) = \sum_k w_k \cdot P_k(A_i, A_j) \quad (2.7)$$

Далі визначаються позитивний потік (2.8) та негативний потік (2.9):

$$\varphi^+(A_i) = \frac{1}{m-1} \sum_j \pi(A_i, A_j) \quad (2.8)$$

$$\varphi^-(A_i) = \frac{1}{m-1} \sum_j \pi(A_j, A_i) \quad (2.9)$$

де P_k - функція переваги за критерієм k;

m - кількість альтернатив.

Чистий потік $\varphi(A_i) = \varphi^+(A_i) - \varphi^-(A_i)$ визначає остаточне ранжування. У реалізованій функції promethee_ii() для кожної пари обчислюється різниця оцінок з урахуванням типу критерію, від'ємні різниці замінюються нулем, а потік нормується на (m−1).

2.5 Ієрархічні методи

Ієрархічні методи ґрунтуються на структурованому розбитті задачі. Метод аналізу ієрархій AHP (Analytic Hierarchy Process), розроблений Томасом Сааті [19], є найвідомішим представником. Пріоритетний вектор ω обчислюється як нормований власний вектор матриці парних порівнянь, що відповідає максимальному власному числу λ_{max} (2.10):

$$A\omega = \lambda_{max} \cdot \omega \quad (2.10)$$

Для перевірки узгодженості суджень розраховується індекс узгодженості CI (2.11) та відношення узгодженості CR (2.12):

$$CI = \frac{\lambda_{max} - n}{n-1} \quad (2.11)$$

$$CR = \frac{CI}{RI} \quad (2.12)$$

де n — порядок матриці;

RI — середній індекс випадкової узгодженості.

Якщо CR ≤ 0,10, судження вважаються достатньо узгодженими. У системі AHP застосовується у двох аспектах: метод парних порівнянь Сааті для визначення ваг критеріїв (з перевіркою CR) та функція ahp() для ранжування альтернатив методом геометричного середнього з подальшим перемноженням пріоритетних векторів на вектор ваг.

2.6 Порівняльний аналіз та обґрунтування вибору методів

Порівняльну характеристику методів за ключовими ознаками наведено в таблиці 2.2.

Таблиця 2.2 — Порівняльна характеристика методів MCDM

Метод	**Клас**	**Компенсаторність**	**Складність**	**Інтерпретованість**
SAW	Зважене оцінювання	Повна	Низька	Висока
WPM	Зважене оцінювання	Повна	Низька	Середня
TOPSIS	Відстань до ідеалу	Повна	Середня	Середня
VIKOR	Відстань до ідеалу	Часткова	Середня	Середня
PROMETHEE II	Переважання	Часткова	Середня	Середня
AHP	Ієрархічний	Повна	Висока	Висока

Для реалізації у системі обрано шість методів, що представляють усі чотири основні класи MCDM: SAW і WPM (зважене оцінювання), TOPSIS і VIKOR (відстань до ідеалу), PROMETHEE II (переважання) та AHP (ієрархічний). Вибір SAW і WPM зумовлений їхньою простотою та прозорістю — вони слугують базовими еталонами порівняння. TOPSIS і VIKOR обрано за геометричну

обґрунтованість та здатність знаходити компромісні рішення. PROMETHEE II включено як некомпенсаторний метод, що відповідає реальним управлінським ситуаціям з обов’язковими порогами. AHP додано як ієрархічний метод із вбудованою перевіркою узгодженості суджень.

Такий набір методів охоплює від повністю компенсаторних до некомпенсаторних підходів і дозволяє порівнювати результати між класами, виявляючи задачі, де методи дають суперечливі ранжування. Усі шість методів реалізовано у модулі mcdm_methods.py з уніфікованим інтерфейсом, що спрощує їх одночасний запуск та подальше розширення системи.

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

У цьому розділі описано проєктування та програмну реалізацію вебсистеми. Спершу сформульовано функціональні та нефункціональні вимоги й наведено діаграму використання. Далі описано архітектуру системи, модель даних і структуру програмних модулів, проілюстровані відповідними діаграмами. У заключних підрозділах розглянуто реалізацію ключових модулів із фрагментами програмного коду.

3.1 Функціональні та нефункціональні вимоги

На основі постановки задачі сформульовано функціональні вимоги до системи, наведені в таблиці 3.1.

Таблиця 3.1 — Функціональні вимоги до системи

ID	Вимога	Пріоритет
FR-01	Реєстрація та автентифікація користувачів (PBKDF2-SHA256)	Високий
FR-02	Створення, редагування та видалення задач з каскадним видаленням	Високий
FR-03	Визначення критеріїв (benefit/cost) та альтернатив задачі	Високий
FR-04	Заповнення матриці оцінок через інтерактивний редактор	Високий
FR-05	Ручне введення ваг критеріїв з автонормалізацією ($\Sigma w=1$)	Високий
FR-06	Визначення ваг методом парних порівнянь Сааті ($CR \leq 0{,}10$)	Середній

Кінець таблиці 3.1

ID	Вимога	Пріоритет
FR-07	Виконання розрахунків за 6 методами MCDM одночасно	Високий
FR-08	Порівняльна таблиця, Bar Chart, Radar Chart, Heatmap кореляцій	Високий
FR-09	Генерація PDF-звіту з ранжуваннями та рекомендаціями	Середній
FR-10	What-if аналіз: зміна матриці та перерахунок	Середній
FR-11	Sensitivity Analysis: аналіз чутливості до ваг	Середній
FR-12	Перегляд історії рішень за всіма задачами	Низький

Нефункціональні вимоги визначають якісні характеристики системи (таблиця 3.2).

Таблиця 3.2 — Нефункціональні вимоги до системи

ID	Вимога	Показник
NFR-01	Продуктивність розрахунку	≤ 2 с для 10 альтернатив × 6 критеріїв
NFR-02	Надійність зберігання даних	SQLite, ACID-транзакції
NFR-03	Безпека паролів	PBKDF2-SHA256, 100 000 ітерацій, сіль 16 байт
NFR-04	Зручність освоєння	≤ 15 хвилин для нового користувача
NFR-05	Розширюваність	нова функція у mcdm_methods.py + 1 рядок у словнику
NFR-06	Сумісність	Chrome, Firefox, Edge (Streamlit WebSocket)

Для наочного представлення основних сценаріїв взаємодії розроблено діаграму використання (use case). Вона охоплює двох акторів — Гість (незареєстрований користувач) та Користувач (ОПР) — і десять варіантів використання: від реєстрації та входу до формування задачі, виконання розрахунків, порівняння методів, сценарного аналізу та перегляду історії. Діаграму наведено на рисунку 3.1.

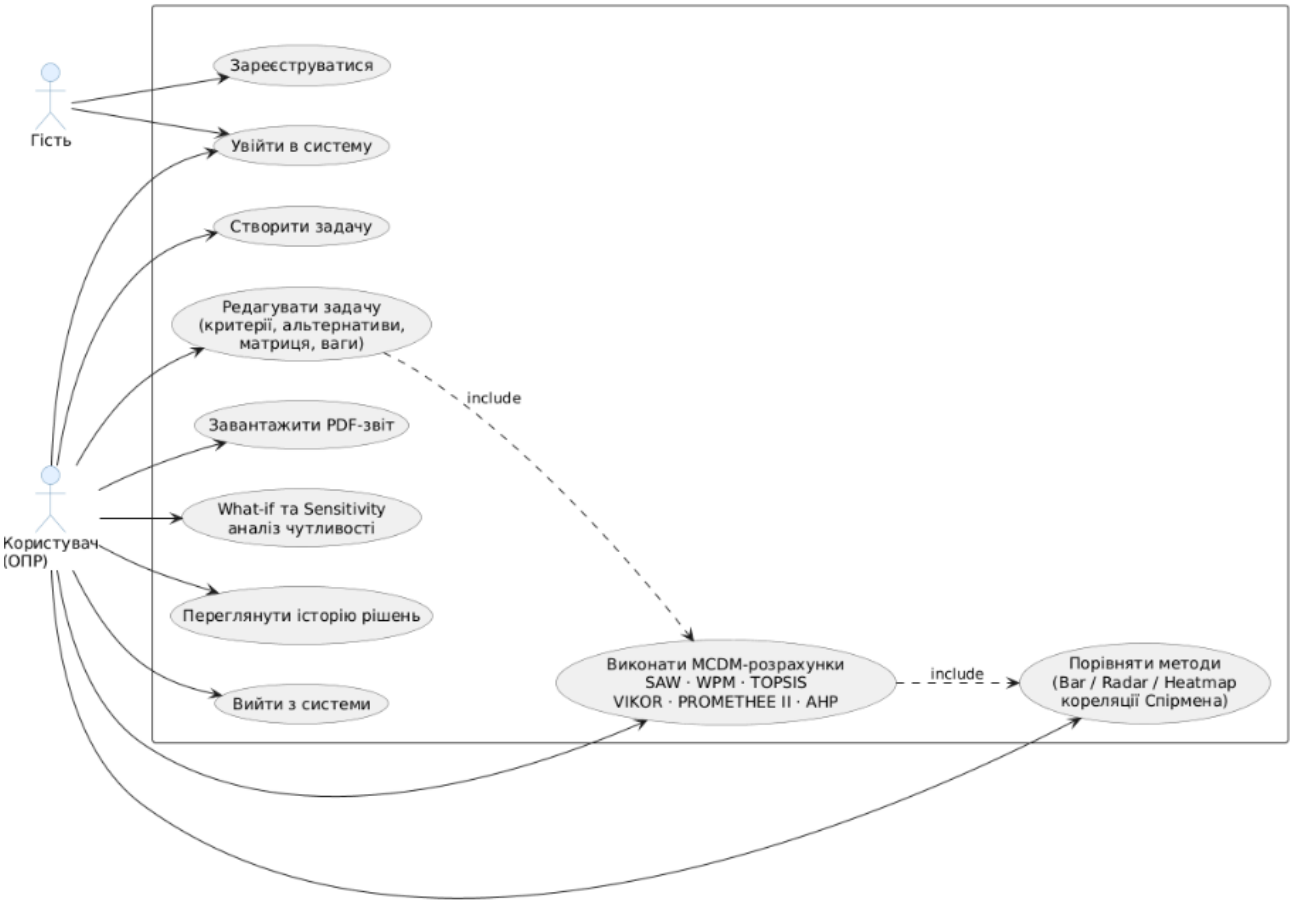


Рисунок 3.1 — Діаграма використання (use case) системи

Наведені вимоги повністю відповідають постановці задачі. Виконання всіх функціональних і нефункціональних вимог підтверджено результатами тестування у розділі 4.

3.2 Архітектура системи

Систему побудовано за трирівневою клієнт-серверною архітектурою на базі фреймворку Streamlit, що чітко розділяє рівень представлення, рівень бізнес-логіки та рівень доступу до даних у межах єдиного Python-застосунку. Точкою входу є файл main.py, який ініціалізує з’єднання з базою даних, застосовує тему

оформлення, перевіряє сесію користувача та забезпечує маршрутизацію між сторінками через бічну навігаційну панель.

Система складається з таких компонентів: рівень представлення (frontend/) — утиліти автентифікації, тема оформлення та вісім модулів сторінок; рівень бізнес-логіки (backend/mcdm/) — обчислювальний модуль із реалізацією шести методів; рівень доступу до даних (backend/repositories/) — сім модулів-репозиторіїв, що ізолюють SQL-запити; база даних SQLite. Архітектурну діаграму компонентів наведено на рисунку 3.2.

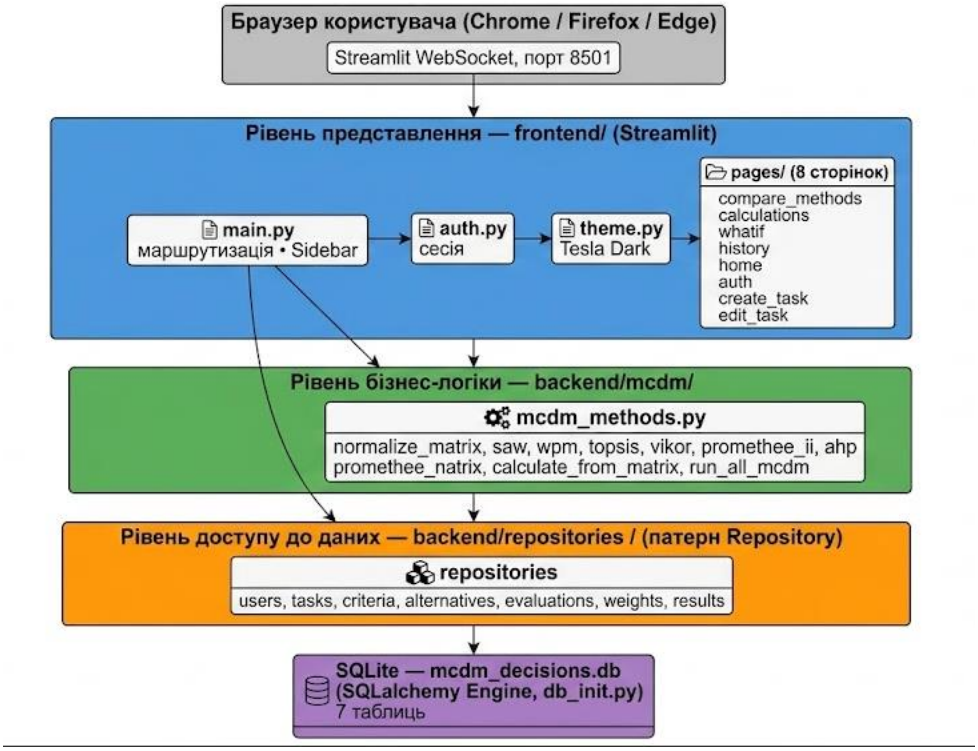


Рисунок 3.2 — Архітектурна діаграма компонентів системи

Ключовим архітектурним рішенням є патерн Repository: кожна таблиця має власний модуль-репозиторій із функціями читання та запису, завдяки чому сторінки інтерфейсу ніколи не звертаються до SQLite напряму. Це забезпечує єдину точку доступу до даних, спрощує тестування та дозволяє замінити СУБД без змін

у бізнес-логіці. Обчислювальний модуль mcdm_methods.py реалізує уніфікований інтерфейс: кожна функція-метод приймає матрицю оцінок, вектор ваг і типи критеріїв та повертає відсортований ряд альтернатив.

3.3 Модель даних

Модель даних реалізовано на SQLite та нормалізовано до третьої нормальної форми. База містить сім таблиць: users (користувачі), tasks (задачі), criteria (критерії), alternatives (альтернативи), evaluations (матриця оцінок), weights (ваги), mcdm_results (результати). Ініціалізація схеми виконується модулем db_init.py через CREATE TABLE IF NOT EXISTS, а міграції — через ALTER TABLE з перехопленням винятку.

Структуру ключових таблиць наведено в таблицях 3.3–3.4, а зв’язки між усіма сутностями — на ER-діаграмі (рисунок 3.3).

Таблиця 3.3 — Структура таблиці users

Поле	**Тип**	**Обмеження**	**Опис**
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор
username	TEXT	NOT NULL, UNIQUE	Логін користувача
email	TEXT	NOT NULL, UNIQUE	Електронна пошта
password_hash	TEXT	NOT NULL	PBKDF2-SHA256, формат «salt:hash»
created_at	TIMESTAMP	DEFAULT now	Дата реєстрації

Центральною сутністю моделі є таблиця tasks, що зберігає задачі користувача та пов'язана з таблицею users відношенням «один до багатьох». Структуру таблиці evaluations, яка реалізує зберігання матриці оцінок у реляційному вигляді, наведено

в таблиці 3.4.

Таблиця 3.4 — Структура таблиці evaluations (матриця оцінок)

Поле	Тип	Обмеження	Опис
id	INTEGER	PK, AUTOINCREMENT	Унікальний ідентифікатор
alternative_id	INTEGER	FK → alternatives.id	Альтернатива
criterion_id	INTEGER	FK → criteria.id	Критерій
value	REAL	NOT NULL	Оцінка (клітинка матриці)

Кожен запис таблиці evaluations містить посилання на альтернативу та критерій, що дозволяє однозначно ідентифікувати відповідну клітинку матриці оцінок. Аналогічно за принципом нормалізації побудовано таблиці criteria, alternatives, weights та mcdm_results, кожна з яких пов'язана із задачею. Повну структуру бази даних та зв'язки між усіма сімома таблицями наведено на ER-діаграмі (рисунок 3.3).

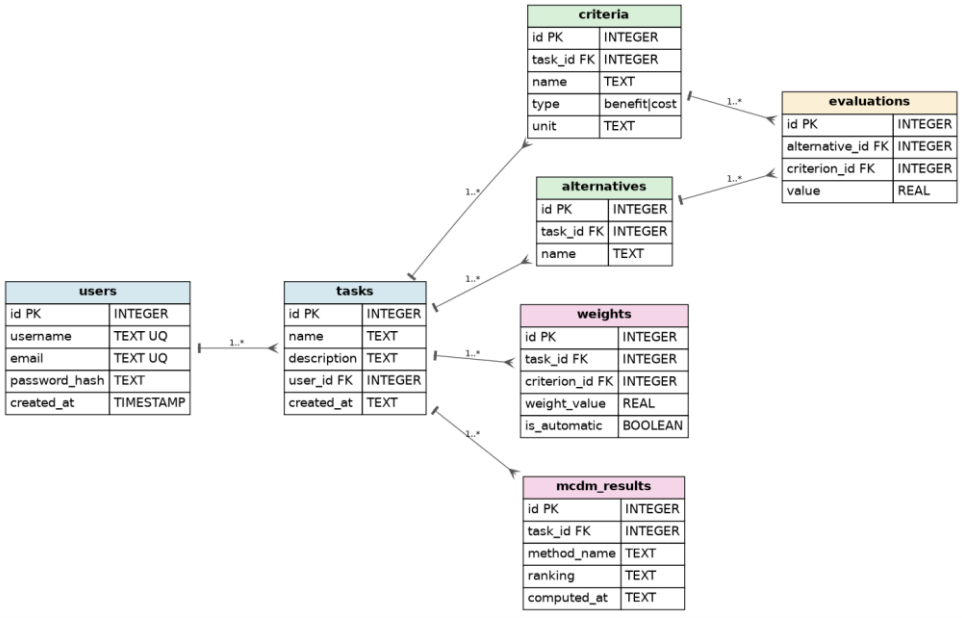


Рисунок 3.3 — ER-діаграма бази даних системи

Зв’язки побудовано за принципом реляційної цілісності: users ↔ tasks, tasks ↔ criteria, tasks ↔ alternatives, tasks ↔ weights, tasks ↔ mcdm_results — один до багатьох; alternatives ↔ evaluations та criteria ↔ evaluations — один до багатьох. Каскадне видалення задачі реалізовано програмно у репозиторії tasks через послідовне видалення пов’язаних записів.

3.4 Структура програмних модулів

Програмні модулі спроєктовано за принципом єдиної відповідальності з чітким розділенням рівнів. Структуру класів та модулів системи з основними методами наведено на діаграмі класів (рисунок 3.4).

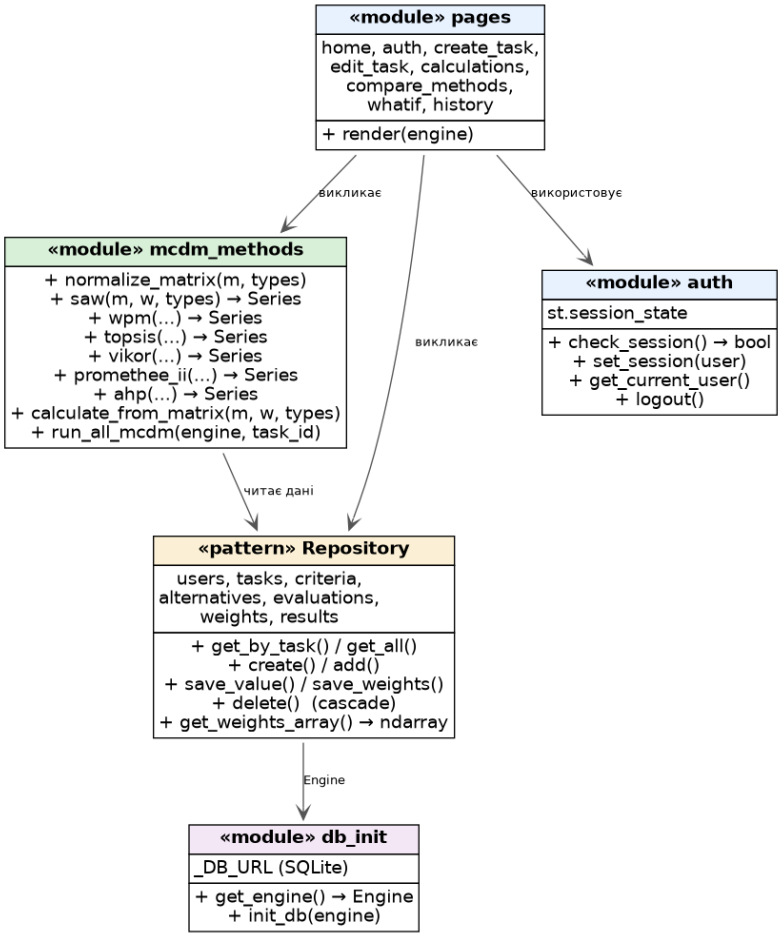


Рисунок 3.4 — Діаграма класів (модулів) системи

Уніфікований інтерфейс кожного MCDM-методу — функція виду f(matrix, weights, criteria_types) → Series — дозволяє додавати нові методи без зміни архітектури: достатньо реалізувати одну функцію та додати один запис до словника методів. Динаміку взаємодії компонентів під час виконання розрахунків ілюструє діаграма послідовностей (рисунок 3.5).

Рисунок 3.5 — Діаграма послідовностей виконання MCDM-розрахунків

Запропонована модульна архітектура забезпечує високу масштабованість, спрощує супровід програмного забезпечення та дозволяє легко розширювати функціональні можливості системи. Завдяки стандартизованим інтерфейсам взаємодії між компонентами реалізовані методи багатокритеріального аналізу можуть використовуватися незалежно один від одного, що підвищує гнучкість системи та спрощує подальшу інтеграцію нових алгоритмів і засобів аналізу.

3.5 Реалізація модуля ініціалізації та автентифікації

Точкою входу системи є файл main.py: він ініціалізує з’єднання з базою даних через SQLAlchemy, запускає створення таблиць, застосовує тему оформлення,

перевіряє поточну сесію та маршрутизує між сторінками. Ініціалізація бази даних (рисунок 3.6), виконується функцією init_db(), яка послідовно створює сім таблиць та виконує міграцію додавання стовпця user_id.

```
12  def init_db(engine):
13      with engine.connect() as conn:
14          for sql in [
15              """CREATE TABLE IF NOT EXISTS users (
16                  id            INTEGER PRIMARY KEY AUTOINCREMENT,
17                  username      TEXT NOT NULL UNIQUE,
18                  email         TEXT NOT NULL UNIQUE,
19                  password_hash TEXT NOT NULL,
20                  created_at    TIMESTAMP DEFAULT CURRENT_TIMESTAMP
21              )""",
```

Рисунок 3.6 — Ініціалізація бази даних

Автентифікацію реалізовано у репозиторії users із хешуванням паролів за алгоритмом PBKDF2-HMAC-SHA256. Сіль генерується випадково (16 байт), а хеш зберігається у форматі «salt:hash» (рисунок 3.7).

```
8  def _hash(password: str, salt: str | None = None) -> str:
9      if salt is None:
10         salt = os.urandom(16).hex()
11     h = hashlib.pbkdf2_hmac("sha256", password.encode(), salt.encode(), 100_000)
12     return f"{salt}:{h.hex()}"
13
14
15 def _verify(password: str, stored_hash: str) -> bool:
16     salt = stored_hash.split(":")[0]
17     return _hash(password, salt) == stored_hash
```

Рисунок 3.7 — Хешування паролів

Такий підхід забезпечує безпечне зберігання облікових даних користувачів і зменшує ризики компрометації паролів у разі несанкціонованого доступу до бази даних. Поєднання механізмів ініціалізації, міграції та автентифікації формує надійну основу для роботи системи, забезпечуючи її стабільність, цілісність даних

та можливість подальшого розвитку функціональності без суттєвих змін базової архітектури.

3.6 Реалізація обчислювальних модулів MCDM-методів

Усі шість методів реалізовано у модулі mcdm_methods.py з уніфікованим інтерфейсом. Базовою операцією є нормалізація матриці рішень (рисунок 3.8).

```
5   def normalize_matrix(matrix, criteria_types):
6       norm_matrix = matrix.copy().astype(float)
7       for j, crit_type in enumerate(criteria_types):
8           col = matrix.iloc[:, j].astype(float)
9           if crit_type == 'benefit':
10              norm_matrix.iloc[:, j] = col / col.max()
11          else:
12              norm_matrix.iloc[:, j] = col.min() / col
13      return norm_matrix
```

Рисунок 3.8 — Нормалізація матриці (mcdm_methods.py)

Методи зваженого оцінювання SAW і WPM реалізовано лаконічно засобами Pandas та NumPy (рисунок 3.9).

```
16  def saw(matrix, weights, criteria_types):
17      norm = normalize_matrix(matrix, criteria_types)
18      scores = (norm * weights).sum(axis=1)
19      return scores.sort_values(ascending=False)
20
21
22  def wpm(matrix, weights, criteria_types):
23      norm = normalize_matrix(matrix, criteria_types)
24      scores = np.prod(np.power(norm.astype(float), weights), axis=1)
25      return scores.sort_values(ascending=False)
```

Рисунок 3.9 — Методи SAW та WPM

Метод TOPSIS обчислює зважену нормалізовану матрицю, визначає ідеальні рішення та евклідові відстані до них (рисунок 3.10).

```python
def topsis(matrix, weights, criteria_types):
    norm = normalize_matrix(matrix, criteria_types)
    weighted = norm * weights
    ideal_best = weighted.max(axis=0)
    ideal_worst = weighted.min(axis=0)
    dist_best = np.sqrt(((weighted - ideal_best) ** 2).sum(axis=1))
    dist_worst = np.sqrt(((weighted - ideal_worst) ** 2).sum(axis=1))
    closeness = dist_worst / (dist_best + dist_worst)
    return closeness.sort_values(ascending=False)
```

Рисунок 3.10 — Метод TOPSIS

Метод VIKOR обчислює міри S, R та компромісний індекс Q; метод PROMETHEE II будує потоки переваги попарним порівнянням; метод AHP формує пріоритетні вектори методом геометричного середнього (рисунок 3.11).

```python
def ahp(matrix, weights, criteria_types):
    """
    AHP через попарні порівняння, похідні з відношень нормалізованих оцінок.
    Пріоритетний вектор обчислюється методом геометричного середнього.
    """
    norm = normalize_matrix(matrix, criteria_types)
    n_alts = norm.shape[0]
    priority_vectors = []

    for j in range(norm.shape[1]):
        col = norm.iloc[:, j].values.astype(float)
        col = np.where(col == 0, 1e-10, col)
        pairwise = np.outer(col, 1.0 / col)
        geo_means = np.power(np.prod(pairwise, axis=1), 1.0 / n_alts)
        s = geo_means.sum()
        priority_vectors.append(geo_means / s if s > 0 else np.ones(n_alts) / n_alts)

    priority_matrix = np.column_stack(priority_vectors)
    scores = priority_matrix @ weights
    return pd.Series(scores, index=norm.index).sort_values(ascending=False)
```

Рисунок 3.11 — Метод AHP

Усі методи запускаються однією функцією run_all_mcdm(), яка зчитує дані задачі з бази, формує матрицю рішень операцією pivot та послідовно застосовує всі шість методів, повертаючи ранжування у форматі «A → B → C». Завдяки

уніфікованому інтерфейсу додавання нового методу потребує лише однієї нової функції та одного запису у словнику методів.

3.7 Реалізація модулів порівняння та сценарного аналізу

Модуль порівняння результатів обчислює коефіцієнти рангової кореляції Спірмена між усіма парами методів за допомогою функції scipy.stats.spearmanr та будує матрицю кореляцій (рисунок 3.12). Результати відображаються у вигляді теплової карти, стовпчастої та радарної діаграм засобами Plotly.

```
216    # — Heatmap кореляції ————————————————————————————————————
217    st.subheader("Heatmap — кореляція методів (коефіцієнт Спірмена)")
218    n = len(methods_list)
219    corr_matrix = np.ones((n, n))
220
221    if len(all_alts) >= 3:
222        for i, m1 in enumerate(methods_list):
223            for j, m2 in enumerate(methods_list):
224                if i != j:
225                    corr, _ = stats.spearmanr(rank_arrays[m1], rank_arrays[m2])
226                    corr_matrix[i][j] = round(corr, 2)
```

Рисунок 3.12 — Матриця кореляцій Спірмена

Модуль сценарного аналізу містить дві функції. What-if аналіз дозволяє редагувати матрицю оцінок без збереження в базі та порівнювати нове ранжування з базовим, відображаючи зміну позицій альтернатив символами підвищення, зниження або незмінності. Sensitivity Analysis виконує sweep-аналіз: вагу обраного критерію змінюють від 0 до 1 з кроком 0,05 із рівномірним перерозподілом решти ваг, після чого для кожного значення перераховують ранжування та виявляють точки зміни лідера.

Генерацію PDF-звіту реалізовано через бібліотеку FPDF2: звіт містить назву задачі, таблицю ранжувань за всіма методами, рекомендовану альтернативу та підсумок; для коректного відображення кирилиці використано unicode-шрифт.

4 ТЕСТУВАННЯ ТА ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ

У цьому розділі описано методику тестування розробленої системи та продемонстровано роботу користувача з її основними функціями за допомогою знімків екрана. Наведено результати порівняльного аналізу методів MCDM на тестовому прикладі, що дозволило оцінити особливості їх застосування та отримані результати. Також виконано аналіз чутливості рішень до зміни ваг критеріїв і сформульовано практичні рекомендації щодо вибору найбільш доцільного методу залежно від умов задачі. Отримані результати підтвердили коректність роботи системи та її ефективність для підтримки багатокритеріального прийняття рішень.

4.1 Методологія та результати тестування

Тестування проводилося з метою перевірки коректності реалізації всіх функціональних і нефункціональних вимог. Стратегія тестування охоплювала модульне тестування функцій MCDM-методів, інтеграційне тестування взаємодії модулів через репозиторії, функціональне тестування сторінок веб-інтерфейсу та перевірку граничних випадків (нульові значення, вирівняні ваги, одна альтернатива).

Основну увагу приділено правильності нормалізації матриці для типів benefit і cost, коректності розрахунків усіх шести методів, збереженню та відновленню даних, роботі автентифікації й ізоляції задач між користувачами та генерації PDF-звіту з підтримкою кирилиці. Результати функціонального тестування наведено в таблиці 4.1.

Таблиця 4.1 — Результати тестування основних функцій системи

№	Тестований функціонал	Очікуваний результат	Статус
1	Реєстрація користувача	Запис у таблиці users, перенаправлення	Пройдено
2	Автентифікація	Сесію встановлено, доступ до розділів	Пройдено
3	Неправильний пароль	Повідомлення про помилку	Пройдено
4	Створення задачі	Запис у tasks, задача у списку	Пройдено
5	Додавання критеріїв	Запис у criteria, відображення у вкладці	Пройдено
6	Заповнення матриці	Значення збережено у evaluations	Пройдено
7	Ручне введення ваг	Автонормалізація, Σw = 1 у базі	Пройдено
8	Метод Сааті (CR > 0,10)	Попередження про неузгодженість	Пройдено
9	MCDM-розрахунки	6 записів у mcdm_results, таблиця рангів	Пройдено
10	Heatmap кореляцій	Матриця 6×6 кореляцій Спірмена	Пройдено
11	PDF-звіт	PDF із ранжуваннями та рекомендацією	Пройдено
12	What-if аналіз	Нові ранжування, дельта позицій	Пройдено
13	Sensitivity Analysis	Графік із точкою зміни лідера	Пройдено
14	Видалення задачі	Каскадне видалення пов’язаних записів	Пройдено
15	Ізоляція задач	Видно лише власні задачі (user_id)	Пройдено

Під час тестування критичних помилок не виявлено. Усі основні функції працюють стабільно, дані коректно зберігаються та відновлюються після перезавантаження сторінки, а інтерфейс реагує на дії користувача без помітних затримок, що задовольняє вимогу NFR-01 щодо продуктивності.

4.2 Взаємодія користувача з системою

Веб-інтерфейс системи спроєктовано для зручної навігації за допомогою бічної панелі, що забезпечує швидкий доступ до всіх розділів. Головна сторінка виконує функцію інформаційного дашборду, на якому відображаються ключові метрики: загальна кількість реалізованих методів, створених задач та виконаних розрахунків. Крім того, для зручності користувача на головному екрані візуалізовано покроковий робочий процес та наведено перелік шести доступних методів багатокритеріального аналізу (рисунок 4.1).

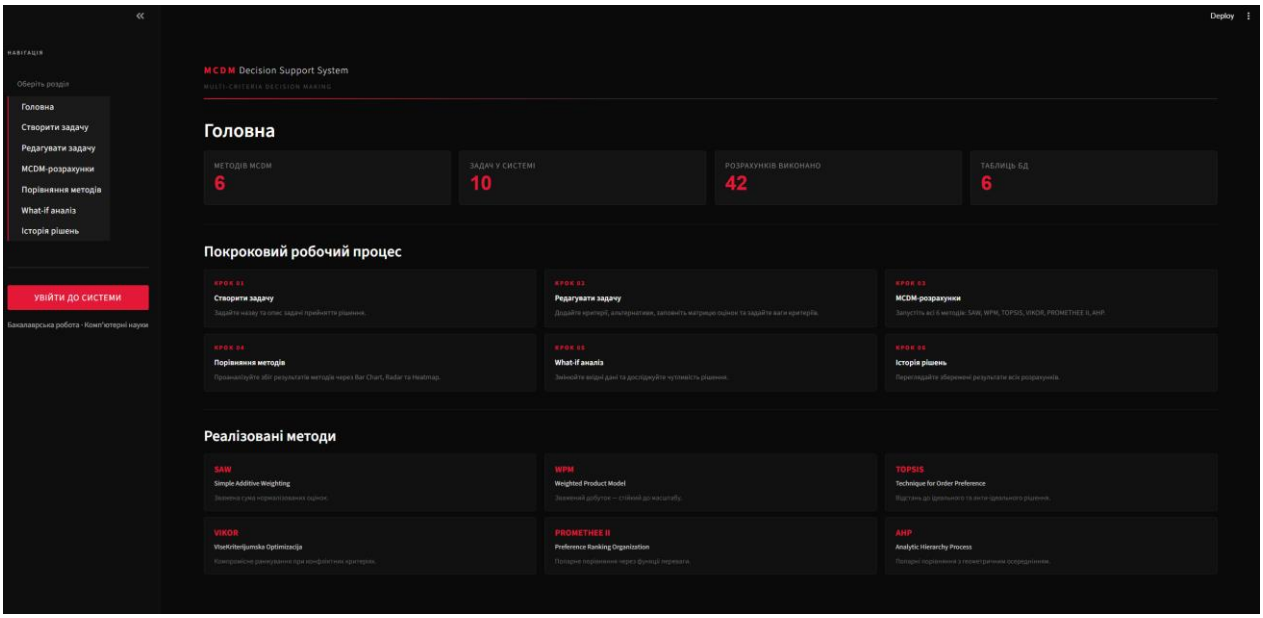


Рисунок 4.1 — Головна сторінка системи

Робота з системою починається зі створення задачі: користувач задає її назву та опис на відповідній сторінці (рисунок 4.2). Після цього у режимі редагування задачі через чотири вкладки додаються критерії (із зазначенням типу benefit/cost та одиниці вимірювання), альтернативи, заповнюється матриця оцінок та визначаються ваги критеріїв.

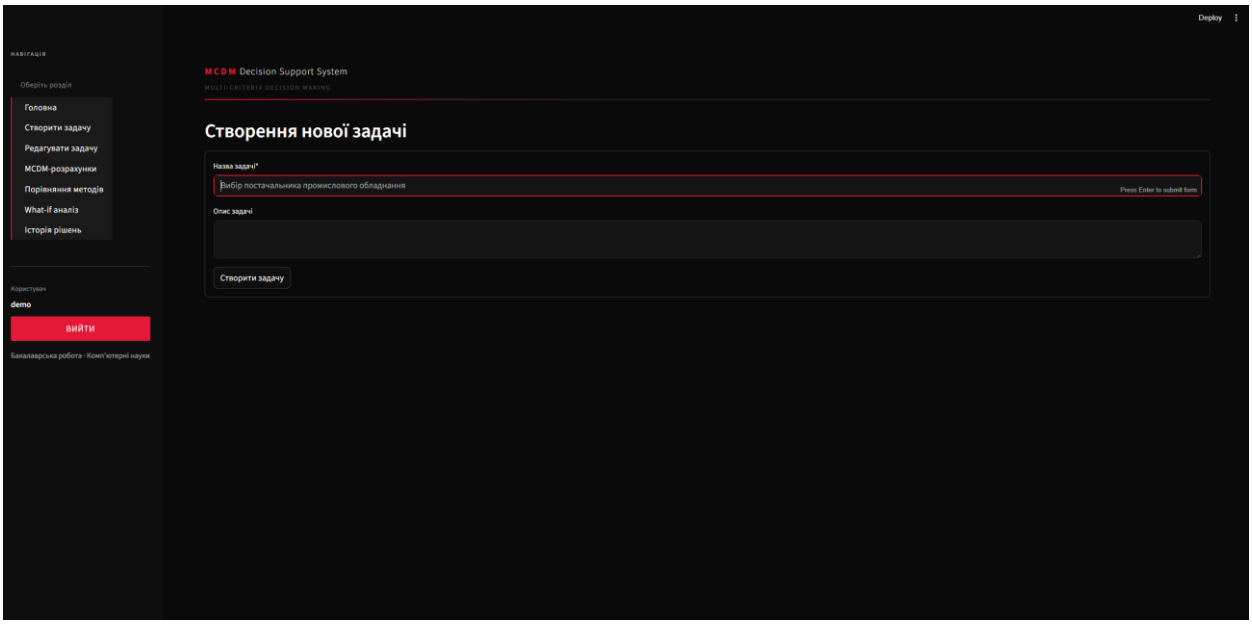


Рисунок 4.2 — Сторінка створення нової задачі

Вкладка визначення ваг підтримує два режими: ручне введення значень з автоматичною нормалізацією до суми, що дорівнює одиниці, та метод парних порівнянь Сааті з обчисленням відношення узгодженості CR (рисунок 4.3).

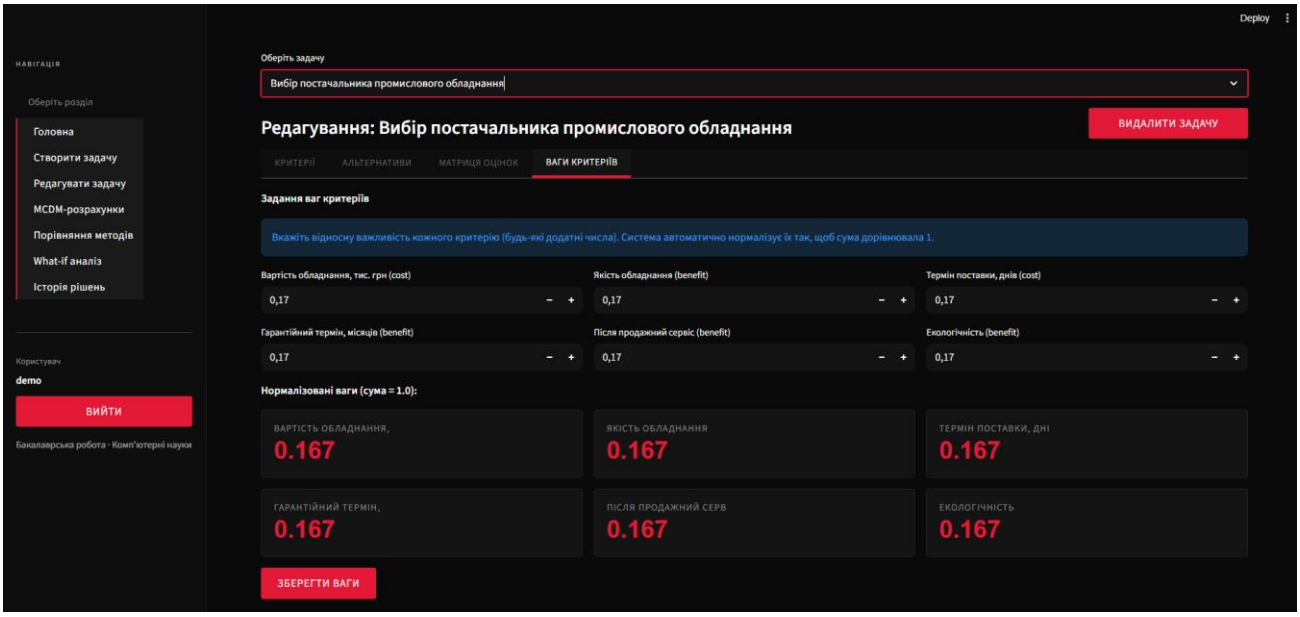
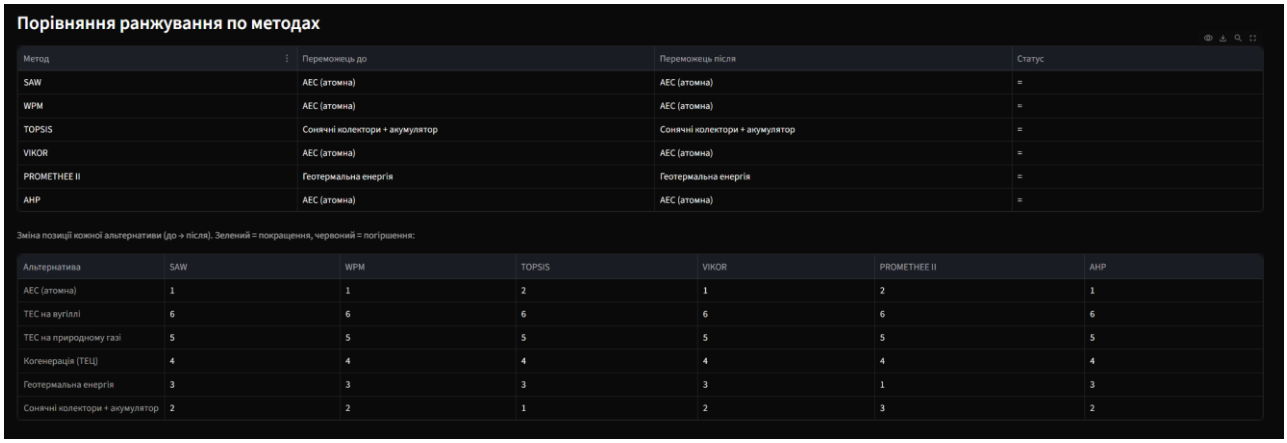


Рисунок 4.3 — Сторінка визначення ваг критеріїв

Після заповнення задачі на сторінці порівняння методів відображається порівняльна таблиця ранжувань за всіма методами, а також інтерактивні візуалізації (рисунок 4.4).



Порівняння ранжування по методах

Метод	Переможець до	Переможець після	Статус
SAW	АЕС (атомна)	АЕС (атомна)	=
WPM	АЕС (атомна)	АЕС (атомна)	=
TOPSIS	Сонячні колектори + акумулятор	Сонячні колектори + акумулятор	=
VIKOR	АЕС (атомна)	АЕС (атомна)	=
PROMETHEE II	Геотермальна енергія	Геотермальна енергія	=
AHP	АЕС (атомна)	АЕС (атомна)	=

Зміна позиції кожної альтернативи (до → після). Зелений = покращення, червоний = погіршення:

Альтернатива	SAW	WPM	TOPSIS	VIKOR	PROMETHEE II	AHP
АЕС (атомна)	1	1	2	1	2	1
ТЕС на вугіллі	6	6	6	6	6	6
ТЕС на природному газі	5	5	5	5	5	5
Когенерація (ТЕЦ)	4	4	4	4	4	4
Геотермальна енергія	3	3	3	3	1	3
Сонячні колектори + акумулятор	2	2	1	2	3	2

Рисунок 4.4 — Порівняльна таблиця ранжувань за методами

Радарна діаграма (рисунок 4.5) відображає профіль оцінок альтернатив за всіма критеріями одночасно, що дозволяє наочно порівнювати їхні сильні та слабкі сторони.

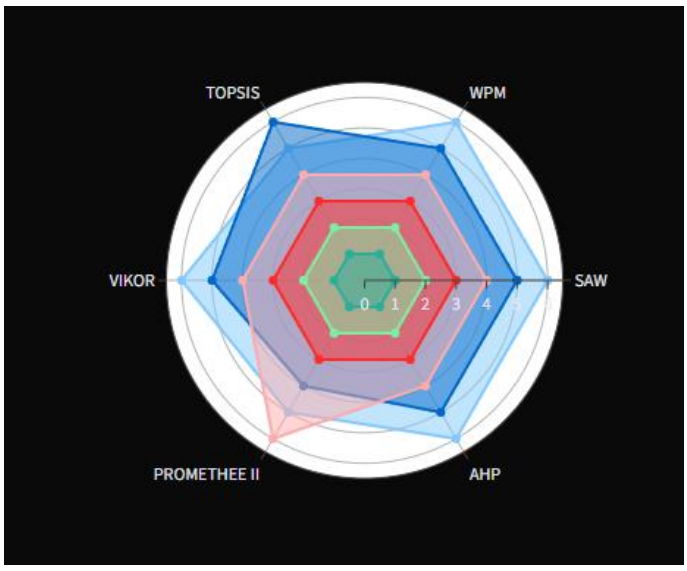


Рисунок 4.5 — Радарна діаграма порівняння методів

Теплова карта (рисунок 4.6) відображає коефіцієнти рангової кореляції Спірмена між результатами різних методів MCDM.

Рисунок 4.6 — Теплова карта кореляцій Спірмена між методами

Інтуїтивний інтерфейс дозволяє ОПР пройти повний цикл багатокритеріального аналізу - від формалізації задачі до отримання обґрунтованої рекомендації - без потреби у взаємодії з базою даних напряму, що відповідає вимозі NFR-04 щодо зручності освоєння.

4.3 Порівняльний аналіз методів на тестовому прикладі

Для оцінки практичної ефективності реалізованих методів проведено порівняльний аналіз на тестовій задачі «Вибір постачальника промислового обладнання», що містить чотири альтернативи та п’ять критеріїв. Параметри задачі наведено в таблиці 4.2.

Таблиця 4.2 — Параметри тестової задачі «Вибір постачальника»

Критерій	Тип	Вага	A	B	C	D
Ціна, тис. грн	cost	0,30	850	620	480	710
Якість, бали	benefit	0,25	9,2	7,5	6,8	8,1

Кінець таблиці 4.2

Критерій	Тип	Вага	A	B	C	D
Термін постачання, дні	cost	0,20	14	21	35	18
Гарантія, міс	benefit	0,15	36	24	18	30
Рейтинг надійності	benefit	0,10	4,8	4,1	3,9	4,4

Результати ранжування альтернатив за шістьма методами наведено в таблиці 4.3.

Таблиця 4.3 — Результати ранжування за методами MCDM

Метод	1 місце	2 місце	3 місце	4 місце
SAW	A	D	B	C
WPM	A	D	B	C
TOPSIS	A	D	B	C
VIKOR	A	D	C	B
PROMETHEE II	A	D	B	C
AHP	A	D	B	C

Аналіз показує, що п’ять із шести методів (SAW, WPM, TOPSIS, PROMETHEE II, AHP) демонструють повну узгодженість ранжування A → D → B → C. Метод VIKOR відрізняється лише позиціями B та C. Для кількісної оцінки узгодженості розраховано коефіцієнти рангової кореляції Спірмена між парами методів (таблиця 4.4).

Таблиця 4.4 — Матриця кореляцій Спірмена між методами (ρ)

Метод	SAW	WPM	TOPSIS	VIKOR	PROM. II	AHP
SAW	1,00	1,00	1,00	0,80	1,00	1,00
WPM	1,00	1,00	1,00	0,80	1,00	1,00
TOPSIS	1,00	1,00	1,00	0,80	1,00	1,00
VIKOR	0,80	0,80	0,80	1,00	0,80	0,80

Кінець таблиці 4.4

Метод	SAW	WPM	TOPSIS	VIKOR	PROM. II	AHP
PROM. II	1,00	1,00	1,00	0,80	1,00	1,00
AHP	1,00	1,00	1,00	0,80	1,00	1,00

Середній коефіцієнт кореляції між усіма парами методів становить $\rho \approx 0{,}97$, що свідчить про високу узгодженість результатів. Відмінність VIKOR від решти методів ($\rho = 0{,}80$) пояснюється специфікою компромісного підходу при близьких значеннях альтернатив B та C за критерієм «Ціна».

4.4 Аналіз чутливості до ваг критеріїв

Для перевірки стійкості отриманого результату проведено аналіз чутливості за допомогою модуля Sensitivity Analysis. Вагу критерію «Ціна» варіювали від 0 до 1 із кроком 0,05, при цьому система автоматично пропорційно перерозподіляла ваги решти критеріїв.

Результати показали, що постачальник A впевнено зберігає лідерство за методом SAW у широкому діапазоні ваги критерію «Ціна» (від 0 до 0,65). Лише при перевищенні порогу в 0,65 на перше місце виходить постачальник C завдяки своїй найнижчій вартості.

Оскільки поточна базова вага цінового критерію становить 0,30, вона знаходиться глибоко в межах стабільної зони. Таким чином, інструментарій системи дозволяє ефективно виявляти критичні точки зміни рішення, надаючи особі, що приймає рішення (ОПР), важливу інформацію для оцінки ризиків та обґрунтування вибору.

4.5 Рекомендації щодо вибору методу MCDM

На основі порівняльного аналізу та тестування сформульовано практичні рекомендації щодо вибору методу залежно від характеристик задачі:

- SAW та WPM - для задач з повністю кількісними критеріями, де потрібен прозорий і швидкий результат; SAW є еталоном для порівняння, WPM менш чутливий до масштабу;

- TOPSIS - для задач вибору постачальника, оцінювання проєктів та фінансових рішень; забезпечує рівноважну оцінку відносно обох ідеальних рішень і добре масштабується;

- VIKOR - для задач, де важливим є компромісне рішення, прийнятне для всіх зацікавлених сторін;

- PROMETHEE II - для задач з якісними критеріями та необхідністю моделювання порогів переваги завдяки некомпенсаторній природі;

- AHP - коли ваги критеріїв невідомі або суперечливі; матриця парних порівнянь структурує визначення пріоритетів, а перевірка CR контролює узгодженість суджень.

При високій невизначеності або конфлікті критеріїв рекомендується застосовувати кілька методів одночасно та аналізувати узгодженість за коефіцієнтом Спірмена, як реалізовано у системі. Значний розкид ранжувань ($\rho < 0,6$) сигналізує про складність задачі та потребу в уточненні вимог або залученні експертів.

ВИСНОВКИ

У дипломній роботі розв’язано актуальну задачу автоматизації підтримки прийняття управлінських рішень шляхом розробки вебсистеми на основі методів багатокритеріального аналізу. Поставлену мету досягнуто, а всі визначені у вступі завдання виконано в повному обсязі.

1. Проаналізовано предметну область підтримки прийняття управлінських рішень та формалізовано постановку задачі багатокритеріального вибору. Огляд наявних рішень (електронних таблиць, комерційних MCDM-додатків та відкритих бібліотек) показав, що жодне з них не поєднує підтримку кількох класів методів, зручний веб-доступ, порівняльний аналіз та сценарне моделювання у відкритому продукті; обґрунтовано вибір технологічного стека: Python 3.12, Streamlit, SQLite, SQLAlchemy, NumPy, SciPy, Pandas, Plotly, FPDF2.

2. Систематизовано методи багатокритеріального аналізу за чотирма класами, наведено їхні математичні моделі та порівняльну характеристику; обґрунтовано вибір шести методів — SAW, WPM, TOPSIS, VIKOR, PROMETHEE II та AHP, що репрезентують усі основні класи MCDM.

3. Спроєктовано трирівневу архітектуру системи на базі фреймворку Streamlit із застосуванням патерна Repository, розроблено модель даних із семи нормалізованих таблиць та структуру програмних модулів з уніфікованим інтерфейсом MCDM-методів, що проілюстровано діаграмами використання, архітектури, ER-діаграмою, діаграмою класів та діаграмою послідовностей.

4. Реалізовано програмні компоненти системи: обчислювальний модуль із шістьма методами MCDM, сім репозиторіїв доступу до даних, веб-інтерфейс із восьми сторінок, модулі порівняння методів за коефіцієнтом рангової кореляції Спірмена, генерації PDF-звіту та сценарного What-if і Sensitivity аналізу; автентифікацію реалізовано з хешуванням паролів за алгоритмом PBKDF2-SHA256.

5. Виконано тестування системи: усі 15 перевірених функцій пройшли тестування без критичних помилок. На тестовій задачі вибору постачальника п’ять із шести методів дали повністю узгоджене ранжування із середнім коефіцієнтом кореляції Спірмена 0,97; аналіз чутливості підтвердив стійкість рекомендованого рішення. Сформульовано практичні рекомендації щодо вибору методу залежно від характеристик задачі.

Практична цінність роботи полягає у створенні готового до використання відкритого веб-інструменту, придатного для обґрунтування управлінських рішень у задачах вибору постачальників, оцінювання проєктів, кадрових та інвестиційних рішень. Завдяки уніфікованому інтерфейсу обчислювальних методів та патерну Repository система є відкритою до розширення: додавання нового методу MCDM або заміна СУБД не потребують зміни архітектури.

Перспективними напрямами подальшого розвитку системи є впровадження методів нечіткого багатокритеріального аналізу (Fuzzy TOPSIS, Fuzzy AHP) для роботи з якісними критеріями в умовах невизначеності, додавання методів класу ELECTRE, реалізація колективного прийняття рішень кількома експертами та розгортання системи у хмарному середовищі з масштабованою СУБД.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Triantaphyllou E. Multi-Criteria Decision Making Methods: A Comparative Study. Dordrecht : Kluwer Academic Publishers, 2000. 320 p.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Київ : ДП «УкрНДНЦ», 2016. 26 с.

3. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : ДП «УкрНДНЦ», 2016. 17 с.

4. Kizielewicz B., Sałabun W. pymcdm — The universal library for solving multi-criteria decision-making problems. SoftwareX. 2023. Vol. 22. P. 1–8.

5. Python 3.12 Documentation. URL: https://docs.python.org/3.12 (дата звернення: 15.03.2026).

6. Streamlit Documentation. URL: https://docs.streamlit.io (дата звернення: 15.03.2026).

7. NumPy Documentation. URL: https://numpy.org/doc (дата звернення: 20.03.2026).

8. SciPy Documentation. scipy.stats.spearmanr. URL: https://docs.scipy.org/doc/scipy (дата звернення: 20.03.2026).

9. pandas Documentation. URL: https://pandas.pydata.org/docs (дата звернення: 20.03.2026).

10. Plotly Python Open Source Graphing Library. URL: https://plotly.com/python (дата звернення: 22.03.2026).

11. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 18.03.2026).

12. SQLAlchemy Documentation. The Database Toolkit for Python. URL: https://docs.sqlalchemy.org (дата звернення: 18.03.2026).

13. fpdf2 — Simple PDF generation for Python. URL https://py-pdf.github.io/fpdf2 (дата звернення: 22.03.2026).

14. Taherdoost H., Madanchian M. Multi-Criteria Decision Making (MCDM) Methods and Concepts. Encyclopedia. 2023. Vol. 3, No. 1. P. 77–87.

15. Ширмовська Н. Г., Шекета В. І. Системи підтримки прийняття рішень: навчальний посібник. Івано-Франківськ : ІФНТУНГ, 2018. 196 с.

16. Opricovic S., Tzeng G. H. Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. European Journal of Operational Research. 2004. Vol. 156, No. 2. P. 445–455.

17. Çelikbilek Y., Tüysüz F. An in-depth review of theory of the TOPSIS method: An experimental analysis. Journal of Management Analytics. 2020. Vol. 7, No. 2. P. 281–300.

18. Brans J. P., Vincke P. A Preference Ranking Organisation Method: The PROMETHEE Method for Multiple Criteria Decision-Making. Management Science. 1985. Vol. 31, No. 6. P. 647–656.

19. Saaty T. L. The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation. New York : McGraw-Hill, 1980. 287 p.

20. Mardani A. et al. VIKOR Technique: A Systematic Review of the State of the Art Literature on Methodologies and Applications. Sustainability. 2016. Vol. 8, No. 1.

ДОДАТОК А

Вебсистема підтримки прийняття управлінських рішень на основі методів багатокритеріального аналізу

Лістинг коду

НТУУ «КПІ ім. Ігоря Сікорського»

Аркушів 16

Київ — 2026

Програмні засоби:

— мова програмування — Python 3.12;

— середовище розробки — Visual Studio Code;

— фреймворк веб-інтерфейсу — Streamlit;

— реляційна база даних — SQLite;

— ORM для доступу до бази даних — SQLAlchemy 2.0;

— виконання операцій з масивами даних — NumPy;

— обробка матриці оцінок — Pandas;

— рангова кореляція методів — SciPy;

— інтерактивна візуалізація — Plotly;

— генерація PDF-звіту — FPDF2.

Лістинг Б.1 — Модуль mcdm_methods.py (обчислювальне ядро)

```
import numpy as np
import pandas as pd

def normalize_matrix(matrix, criteria_types):
    norm_matrix = matrix.copy().astype(float)
    for j, crit_type in enumerate(criteria_types):
        col = matrix.iloc[:, j].astype(float)
        if crit_type == 'benefit':
            norm_matrix.iloc[:, j] = col / col.max()
        else:
            norm_matrix.iloc[:, j] = col.min() / col
    return norm_matrix

def saw(matrix, weights, criteria_types):
    norm = normalize_matrix(matrix, criteria_types)
```

```
    scores = (norm * weights).sum(axis=1)
    return scores.sort_values(ascending=False)

def wpm(matrix, weights, criteria_types):
    norm = normalize_matrix(matrix, criteria_types)
    scores = np.prod(np.power(norm.astype(float), weights), axis=1)
    return scores.sort_values(ascending=False)

def topsis(matrix, weights, criteria_types):
    norm = normalize_matrix(matrix, criteria_types)
    weighted = norm * weights
    ideal_best = weighted.max(axis=0)
    ideal_worst = weighted.min(axis=0)
    dist_best = np.sqrt(((weighted - ideal_best) ** 2).sum(axis=1))
    dist_worst = np.sqrt(((weighted - ideal_worst) ** 2).sum(axis=1))
    closeness = dist_worst / (dist_best + dist_worst)
    return closeness.sort_values(ascending=False)

def vikor(matrix, weights, criteria_types):
    norm = normalize_matrix(matrix, criteria_types)
    f_max = norm.max(axis=0)
    f_min = norm.min(axis=0)
    denom = (f_max - f_min).replace(0, 1e-10)
    S = (weights * (f_max - norm) / denom).sum(axis=1)
    R = (weights * (f_max - norm) / denom).max(axis=1)
    s_range = S.max() - S.min()
    r_range = R.max() - R.min()
    Q = (
        0.5 * (S - S.min()) / (s_range if s_range > 0 else 1e-10)
```

```
            + 0.5 * (R - R.min()) / (r_range if r_range > 0 else 1e-10)
        )
        return Q.sort_values(ascending=True)

    def promethee_ii(matrix, weights, criteria_types):
        m = matrix.shape[0]
        phi = np.zeros(m)
        for i in range(m):
            for j in range(m):
                if i != j:
                    diff = matrix.iloc[i].astype(float) - matrix.iloc[j].astype(float)
                    pref = np.where(criteria_types == 'benefit', diff, -diff)
                    pref = np.maximum(0, pref)
                    phi[i] += np.sum(weights * pref)
        phi = phi / (m - 1)
        return pd.Series(phi, index=matrix.index).sort_values(ascending=False)

    def ahp(matrix, weights, criteria_types):
        """
        AHP через попарні порівняння, похідні з відношень нормалізованих оцінок.
        Пріоритетний вектор обчислюється методом геометричного середнього.
        """
        norm = normalize_matrix(matrix, criteria_types)
        n_alts = norm.shape[0]
        priority_vectors = []

        for j in range(norm.shape[1]):
            col = norm.iloc[:, j].values.astype(float)
```

```
            col = np.where(col == 0, 1e-10, col)
            pairwise = np.outer(col, 1.0 / col)
            geo_means = np.power(np.prod(pairwise, axis=1), 1.0 / n_alts)
            s = geo_means.sum()
            priority_vectors.append(geo_means / s if s > 0 else np.ones(n_alts) / n_alts)

        priority_matrix = np.column_stack(priority_vectors)
        scores = priority_matrix @ weights
        return pd.Series(scores, index=norm.index).sort_values(ascending=False)

    def calculate_from_matrix(matrix, weights, criteria_types):
        """Запустить все 6 методов на переданной матрице. Возвращает dict
метод → строка рангов."""
        methods = {
            "SAW": saw,
            "WPM": wpm,
            "TOPSIS": topsis,
            "VIKOR": vikor,
            "PROMETHEE II": promethee_ii,
            "AHP": ahp,
        }
        ranking_table = {}
        for name, func in methods.items():
            try:
                result = func(matrix.copy(), weights, criteria_types)
                ranking_table[name] = " → ".join(result.index)
            except Exception as e:
                ranking_table[name] = f"Помилка: {str(e)}"
        return ranking_table
```

```
    def run_all_mcdm(engine, task_id):
        from backend.repositories import weights as weight_repo

        criteria = pd.read_sql(f"SELECT * FROM criteria WHERE task_id = {int(task_id)}", engine)
        alternatives = pd.read_sql(f"SELECT * FROM alternatives WHERE task_id = {int(task_id)}", engine)
        evaluations = pd.read_sql(
            f"""SELECT e.* FROM evaluations e
                JOIN alternatives a ON e.alternative_id = a.id
                WHERE a.task_id = {int(task_id)}""",
            engine,
        )

        if criteria.empty or alternatives.empty or evaluations.empty:
            return {
                "success": False,
                "message": "Недостатньо даних. Перевірте, що є критерії, альтернативи та повністю заповнена матриця оцінок.",
            }

        matrix = evaluations.pivot(index='alternative_id', columns='criterion_id', values='value')
        matrix = matrix.reindex(alternatives['id']).reset_index(drop=True)
        matrix.index = alternatives['name'].values
        criteria_types = criteria.set_index('id')['type'].loc[matrix.columns].values

        weights = weight_repo.get_weights_array(engine, task_id,
```

```
matrix.columns.tolist())
            if weights is None:
                weights = np.ones(len(criteria)) / len(criteria)

            return {
                "success": True,
                "ranking_table": calculate_from_matrix(matrix, weights, criteria_types),
                "matrix": matrix.to_dict(),
            }
```

Лістинг Б.2 — Модуль db_init.py (ініціалізація бази даних)

```
import os
from sqlalchemy import create_engine, text

_BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
_DB_URL = f"sqlite:///{os.path.join(_BASE_DIR, 'mcdm_decisions.db')}"

def get_engine():
    return create_engine(_DB_URL, echo=False)

def init_db(engine):
    with engine.connect() as conn:
        for sql in [
            """CREATE TABLE IF NOT EXISTS users (
                id            INTEGER PRIMARY KEY AUTOINCREMENT,
                username      TEXT NOT NULL UNIQUE,
                email         TEXT NOT NULL UNIQUE,
                password_hash TEXT NOT NULL,
```

```
    created_at    TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)""",
"""CREATE TABLE IF NOT EXISTS tasks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    description TEXT,
    user_id INTEGER REFERENCES users(id),
    created_at TEXT DEFAULT CURRENT_TIMESTAMP
)""",
"""CREATE TABLE IF NOT EXISTS criteria (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    task_id INTEGER,
    name TEXT NOT NULL,
    type TEXT CHECK(type IN ('benefit', 'cost')),
    unit TEXT,
    FOREIGN KEY(task_id) REFERENCES tasks(id)
)""",
"""CREATE TABLE IF NOT EXISTS alternatives (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    task_id INTEGER,
    name TEXT NOT NULL,
    FOREIGN KEY(task_id) REFERENCES tasks(id)
)""",
"""CREATE TABLE IF NOT EXISTS evaluations (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    alternative_id INTEGER,
    criterion_id INTEGER,
    value REAL NOT NULL,
    FOREIGN KEY(alternative_id) REFERENCES alternatives(id),
```

```
            FOREIGN KEY(criterion_id) REFERENCES criteria(id)
        )""",
        """CREATE TABLE IF NOT EXISTS weights (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            task_id INTEGER,
            criterion_id INTEGER,
            weight_value REAL NOT NULL,
            is_automatic BOOLEAN DEFAULT false,
            FOREIGN KEY(task_id) REFERENCES tasks(id)
        )""",
        """CREATE TABLE IF NOT EXISTS mcdm_results (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            task_id INTEGER,
            method_name TEXT NOT NULL,
            ranking TEXT,
            computed_at TEXT DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY(task_id) REFERENCES tasks(id)
        )""",
    ]:
        conn.execute(text(sql))
    conn.commit()

# Міграція: додати user_id до tasks якщо ще немає
from sqlalchemy.exc import OperationalError
try:
    with engine.connect() as conn:
        conn.execute(text("ALTER TABLE tasks ADD COLUMN user_id INTEGER
REFERENCES users(id)"))
        conn.commit()
```

```
    except OperationalError:
        pass
```

Лістинг Б.3 — Модуль repositories/users.py (автентифікація)

```
import hashlib
import os

import pandas as pd
from sqlalchemy import text

def _hash(password: str, salt: str | None = None) -> str:
    if salt is None:
        salt = os.urandom(16).hex()
    h = hashlib.pbkdf2_hmac("sha256", password.encode(), salt.encode(), 100_000)
    return f"{salt}:{h.hex()}"

def _verify(password: str, stored_hash: str) -> bool:
    salt = stored_hash.split(":")[0]
    return _hash(password, salt) == stored_hash

def register(engine, username: str, email: str, password: str) -> dict:
    username = username.strip()
    email = email.strip().lower()

    with engine.connect() as conn:
        existing = pd.read_sql(
            "SELECT id FROM users WHERE username = :u OR email = :e",
            conn, params={"u": username, "e": email},
```

```
        )
        if not existing.empty:
            # Визначаємо що саме зайнято
            by_name = pd.read_sql("SELECT id FROM users WHERE username = :u",
conn, params={"u": username})
            if not by_name.empty:
                raise ValueError(f"Ім'я користувача «{username}» вже зайнято.")
            raise ValueError(f"Email «{email}» вже зареєстровано.")

        conn.execute(
            text("INSERT INTO users (username, email, password_hash) VALUES (:u, :e,
:h)"),
            {"u": username, "e": email, "h": _hash(password)},
        )
        conn.commit()
        user_id = conn.execute(text("SELECT last_insert_rowid()")).scalar()

    return {"id": user_id, "username": username, "email": email}

def login(engine, username_or_email: str, password: str) -> dict | None:
    val = username_or_email.strip()
    with engine.connect() as conn:
        row = pd.read_sql(
            "SELECT id, username, email, password_hash FROM users "
            "WHERE username = :v OR email = :v",
            conn, params={"v": val},
        )
    if row.empty:
        return None
```

```
    user = row.iloc[0]
    if not _verify(password, user["password_hash"]):
        return None
    return {"id": int(user["id"]), "username": user["username"], "email": user["email"]}


def get_by_username(engine, username: str) -> dict | None:
    with engine.connect() as conn:
        row = pd.read_sql(
            "SELECT id, username, email FROM users WHERE username = :u",
            conn, params={"u": username},
        )
    if row.empty:
        return None
    u = row.iloc[0]
    return {"id": int(u["id"]), "username": u["username"], "email": u["email"]}
```

Лістинг Б.4 — Модуль repositories/weights.py (ваги з нормалізацією)

```
import numpy as np
import pandas as pd
from sqlalchemy import text


def save_weights(engine, task_id, criterion_weight_map):
    """Сохранить веса. criterion_weight_map = {criterion_id: weight_value}.
Нормализует до суммы 1."""
    total = sum(criterion_weight_map.values())
    if total == 0:
        return
    with engine.connect() as conn:
```

```
        conn.execute(text("DELETE FROM weights WHERE task_id = :t"), {"t":
int(task_id)})
        for crit_id, weight in criterion_weight_map.items():
            conn.execute(
                text(
                    "INSERT INTO weights (task_id, criterion_id, weight_value) "
                    "VALUES (:t, :c, :w)"
                ),
                {"t": int(task_id), "c": int(crit_id), "w": float(weight) / total},
            )
        conn.commit()


def get_weights_array(engine, task_id, criteria_ids):
    """Вернуть numpy-массив весов в порядке criteria_ids, или None если весов
нет."""
    df = pd.read_sql(
        f"SELECT criterion_id, weight_value FROM weights WHERE task_id =
{int(task_id)}",
        engine,
    )
    if df.empty:
        return None
    n = len(criteria_ids)
    weights = []
    for cid in criteria_ids:
        row = df[df['criterion_id'] == int(cid)]
        weights.append(float(row['weight_value'].iloc[0]) if not row.empty else 1.0 / n)
    return np.array(weights, dtype=float)
```

Лістинг Б.5 — Головний модуль main.py

```
import sys
import os

sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)))

import streamlit as st
import backend.db_init as db
from frontend.theme import apply_tesla_theme
from frontend import auth as auth_utils
from frontend.pages import (
    home,
    create_task,
    edit_task,
    calculations,
    compare_methods,
    whatif,
    history,
    auth as auth_page,
)

st.set_page_config(page_title="MCDM Decision Support System", layout="wide")
apply_tesla_theme()

@st.cache_resource
def get_engine():
    engine = db.get_engine()
    db.init_db(engine)
```

```
    return engine

engine = get_engine()

# Зчитуємо і одразу видаляємо флаг — щоб після закриття діалогу він не відкрився знову
_open_dialog = st.session_state.pop("_open_auth_dialog", False)

with st.sidebar:
    st.header("Навігація")
    page = st.radio(
        "Оберіть розділ",
        [
            "Головна",
            "Створити задачу",
            "Редагувати задачу",
            "MCDM-розрахунки",
            "Порівняння методів",
            "What-if аналіз",
            "Історія рішень",
        ],
        label_visibility="collapsed",
    )

    st.sidebar.divider()

    if auth_utils.check_session():
        user = auth_utils.get_current_user()
        st.sidebar.markdown(
```

```
            f"<div style='font-size:0.8rem;color:#999;margin-bottom:6px'>"
            f"Користувач</div>"
            f"<div style='font-size:0.95rem;color:#FFF;font-weight:600;"
            f"margin-bottom:10px'>{user['username']}</div>",
            unsafe_allow_html=True,
        )
        if st.sidebar.button("Вийти", use_container_width=True):
            auth_utils.logout()
            st.rerun()
    else:
        if st.sidebar.button("Увійти до системи", type="primary",
use_container_width=True):
            st.session_state["_open_auth_dialog"] = True
            st.rerun()

    st.sidebar.caption("Бакалаврська робота · Комп'ютерні науки")

# ── Не залогінений
─────────────────────────────────────────────────────────────────────────
──────────────

if not auth_utils.check_session():
    if page == "Головна":
        home.render(engine)
    else:
        auth_page.render(engine)

    if _open_dialog:
        auth_page.show_auth_dialog(engine)
```

```
    st.stop()

# ── Залогінений
──────────────────────────────────────────────────────────────────────────────

──────────────────────

pages = {
    "Головна": home,
    "Створити задачу": create_task,
    "Редагувати задачу": edit_task,
    "MCDM-розрахунки": calculations,
    "Порівняння методів": compare_methods,
    "What-if аналіз": whatif,
    "Історія рішень": history,
}

pages[page].render(engine)
```