МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 37.00.00.000 ПЗ

Група ІПм-23-3

Туз Віталій

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Туз Віталій Володимирович

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі, методи та алгоритми автоматизації процесів реалізації підтримки прийняття рішень

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Туз В.В.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Зікратий Сергій Вікторович, к.т.н., доцент

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

Допущено до захисту

Завідувач кафедри
доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль
доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Тузу Віталію Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи **“**Моделі, методи та алгоритми автоматизації процесів реалізації підтримки прийняття рішень**”**
керівник проекту (роботи) Зікратий Сергій Вікторович, к.т.н., доцент
затверджені наказом закладу вищої освіти від **“** 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Архітектура, опис та алгоритми систем підтримки прийняття рішень

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих рішень та огляд технологій, обраних для реалізації
2. Алгоритми підтримки прийняття рішень
3. Реалізація та документування власного програмного забезпечення для підтримки прийняття рішень

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Схема процесу прийняття рішень (рис. 2.1., ст. 39)
2. Вигляд головної сторінки додатку (рис. 3.5., ст. 64)
3. Початковий вигляд сторінки форми вводу даних для методу АНР (рис. 3.7., ст. 84)
4. Вигляд форми вводу даних для методу АНР (рис. 3.8, ст. 84)
5. Сторінка результатів методу АНР (рис. 3.10., ст. 85)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	20.09.2024	виконано
2	Аналіз існуючих рішень систем підтримки прийняття рішень	01.10.2024	виконано
3	Проектування власного рішення	12.10.2024	виконано
4	Реалізація власної системи підтримки прийняття рішень	25.10.20234	виконано
5	Виконання тестування та налагодження програмного забезпечення	05.11.2024	виконано
6	Оформлення пояснювальної записки	22.11.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 108 с., 21 рис., 1 табл., 43 джерела.

Тема: Моделі, методи та алгоритми автоматизації процесів реалізації підтримки прийняття рішень.

Об’єкт дослідження: сучасні інформаційні технології, що використовуються для веб-розробки та підтримки прийняття рішень.

Мета роботи: дослідження та впровадження сучасних веб-технологій для створення інформаційної системи, що підтримує прийняття рішень в умовах невизначеності.

Предмет дослідження: методи інтеграції алгоритмів прийняття рішень із веб-додатками для забезпечення їхньої гнучкості, ефективності та зручності використання.

Результати дослідження:

Виконано аналіз існуючих методів підтримки прийняття рішень і веб-технологій, на основі якого запропоновано власну архітектуру системи.

Висновок:

У результаті досліджень було створено систему підтримки прийняття рішень, що поєднує сучасні веб-технології та алгоритми аналізу. Вона забезпечує ефективність і зручність роботи з багатокритеріальними задачами, демонструючи високу адаптивність і функціональність для різних сфер застосування.

СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, ВЕБ-ТЕХНОЛОГІЇ, FLASK, PYTHON, БАГАТОКРИТЕРІАЛЬНИЙ АНАЛІЗ, АЛГОРИТМИ ПРИЙНЯТТЯ РІШЕНЬ.

ANNOTATION

Master's work: 108 p., 21 fig., 1 tab., 43 sources.

Topic: Models, methods and algorithms for automating decision support processes.

Object of research: modern information technologies used for web development and decision support.

Purpose: research and implementation of modern web technologies to create an information system that supports decision-making under conditions of uncertainty.

Subject of research: methods for integrating decision-making algorithms with web applications to ensure their flexibility, efficiency, and usability.

Research results:

Performed analysis of existing decision support methods and web technologies, and on its basis is proposed proprietary system architecture.

Conclusion:

As a result of the research, a decision support system was created that combines modern web technologies and analysis algorithms. It provides efficiency and convenience in working with multi-criteria tasks, demonstrating high adaptability and functionality for various fields of application.

DECISION SUPPORT SYSTEM, WEB TECHNOLOGIES, FLASK, PYTHON, MULTI-CRITERION ANALYSIS, DECISION-MAKING ALGORITHMS.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – програмний інтерфейс додатка, що дозволяє взаємодіяти між різними програмними компонентами.

DOM – об’єктна модель документа, що описує структуру HTML або XML документа як дерева об’єктів.

HTTP – протокол передачі гіпертексту для передачі даних у мережі Інтернет.

PDF – формат файлів для зберігання документів незалежно від програмного забезпечення та апаратного забезпечення.

MCDM – Multiple-criteria decision-making.

AHP – Analytic hierarchy process.

WSM – Weighted sum model.

ВСТУП

Актуальність роботи

У сучасному світі цифрових технологій створення ефективних веб-додатків та підтримка прийняття рішень в умовах невизначеності стають важливими аспектами різних сфер діяльності. Розвиток інформаційних технологій вимагає інтеграції потужних інструментів для обробки даних, візуалізації результатів та створення інтуїтивно зрозумілих інтерфейсів. Водночас, зростаюча складність управлінських завдань і необхідність приймати обґрунтовані рішення в умовах недостатньої інформації стимулюють дослідження ефективних алгоритмів підтримки прийняття рішень.

Використання сучасних технологій веб-розробки, таких як HTML, CSS, JavaScript, а також інструментів для обробки даних, включно з Python і фреймворком Flask, дає змогу створювати багатофункціональні системи, здатні обробляти великі обсяги інформації та забезпечувати гнучкість у прийнятті рішень. Актуальність дослідження зумовлена необхідністю розробки таких систем, які поєднують зручність користування з потужними обчислювальними можливостями і здатністю адаптуватися до різних сценаріїв прийняття рішень.

Порівняння роботи з відомими розв’язаннями проблеми

Існує безліч інструментів і методів для створення веб-додатків та підтримки прийняття рішень, зокрема такі фреймворки, як Django та бібліотеки для візуалізації даних. Однак більшість із них або надто складні для швидкого впровадження в невеликі проекти, або вимагають значних обчислювальних ресурсів. У цій роботі зроблено акцент на використанні легких і гнучких інструментів, таких як Flask і Chart.js, які забезпечують баланс між функціональністю та простотою. Важливою перевагою запропонованого підходу є інтеграція алгоритмів підтримки прийняття рішень із веб-інтерфейсом для швидкого доступу до результатів і забезпечення гнучкості в обробці даних.

Мета і задачі дослідження

Метою магістерської роботи є дослідження та впровадження сучасних веб-технологій для створення інформаційної системи, що підтримує прийняття рішень в умовах невизначеності. Основними задачами роботи є:

1) Огляд існуючих веб-технологій і алгоритмів підтримки прийняття рішень.

2) Аналіз критеріїв прийняття рішень і їхнього застосування в практичних завданнях.

3) Розробка інтегрованої моделі, що поєднує веб-технології та алгоритми прийняття рішень.

4) Програмна реалізація системи та її тестування на основі практичних сценаріїв.

Об’єктом дослідження є сучасні інформаційні технології, що використовуються для веб-розробки та підтримки прийняття рішень.

Предметом дослідження є методи інтеграції алгоритмів прийняття рішень із веб-додатками для забезпечення їхньої гнучкості, ефективності та зручності використання.

Методи дослідження

Для досягнення поставлених цілей було застосовано методи системного аналізу, порівняння існуючих технологій, моделювання алгоритмів прийняття рішень і експериментального тестування програмного забезпечення.

Наукова новизна одержаних результатів

У роботі запропоновано новий підхід до інтеграції веб-технологій із алгоритмами підтримки прийняття рішень, що дозволяє підвищити ефективність і гнучкість інформаційних систем. Розроблено модель, яка забезпечує швидкий доступ до результатів аналізу та підтримує адаптацію до різних умов роботи.

Практичне значення одержаних результатів

Розроблену систему можна використовувати для створення веб-додатків, що підтримують прийняття рішень у різних сферах: управлінні, економіці, науці. Вона забезпечує користувачів інтуїтивно зрозумілим інтерфейсом і потужними інструментами для аналізу даних.

Структура магістерської роботи

Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків. Робота викладена на 108 сторінках тексту, містить 21 рисунок.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОГЛЯД ТЕХНОЛОГІЙ, ОБРАНИХ ДЛЯ РЕАЛІЗАЦІЇ

1.1. Огляд існуючих систем підтримки прийняття рішень

Системи підтримки прийняття рішень відіграють важливу роль у сучасному аналізі даних та розробці стратегій, забезпечуючи ефективне вирішення складних багатокритеріальних задач. Такі системи дозволяють інтегрувати різноманітні джерела інформації, виконувати моделювання, оцінювати альтернативи та формувати обґрунтовані рекомендації для прийняття рішень. Застосування подібних технологій охоплює численні галузі, включаючи управління ресурсами, планування, економіку та інженерію.

Основною перевагою таких систем є здатність обробляти складні сценарії, враховуючи кількісні та якісні критерії, взаємозв’язки між ними, а також їхню чутливість до змін параметрів. Вони дозволяють знизити ризики та невизначеність, які часто виникають у процесі прийняття рішень, і забезпечують прозорість аналізу.

1.1.1. Огляд програмного забезпечення Definite/BOSDA

Програмне забезпечення Definite/BOSDA є одним із провідних інструментів для багатокритеріального прийняття рішень, розробленим для вирішення задач аналізу альтернатив у складних сценаріях. Його основна мета полягає у створенні платформи для прийняття обґрунтованих рішень, що враховують численні критерії та їхні вагомості. Definite широко застосовується у сфері екологічного планування, управління ресурсами та оцінки сталого розвитку, де потрібен гнучкий підхід до оцінки альтернатив з використанням декількох моделей.

Definite підтримує численні методи аналізу, включаючи AHP, ELECTRE, PROMETHEE, SMART, що дозволяє адаптувати програму до різних задач. Наприклад, метод AHP широко використовується для структурованого оцінювання складних рішень, а PROMETHEE забезпечує порівняння альтернатив на основі

переваг. Програма також пропонує функціональність для аналізу чутливості, інтеграцію з електронними таблицями Excel, а також інструменти для візуалізації результатів, що є важливим для ефективного представлення даних замовникам або аналітикам.

Інтерфейс програми представлено на рисунку 1.1. Вигляд інтерфейсу Definite/BOSDA.

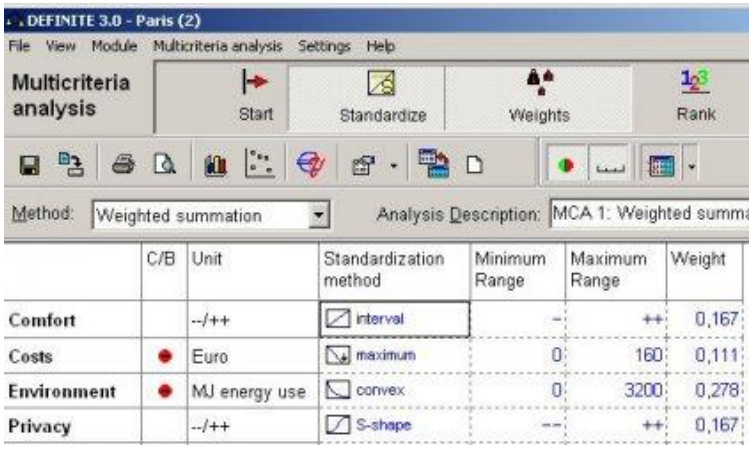


Рис.1.1. Вигляд інтерфейсу Definite/BOSDA

Попри значний набір функцій, програмне забезпечення має суттєві обмеження. Одним із них є висока вартість ліцензії, що робить його недоступним для невеликих організацій або окремих дослідників. Ціна варіюється від 750 євро для академічного використання до 1360 євро для комерційного застосування. Крім того, демоверсія, яка доступна для тестування, має обмежений функціонал і не дозволяє повною мірою оцінити всі можливості програми. Також робота з деякими функціями потребує попередньої технічної підготовки, що може створити труднощі для користувачів, які не мають досвіду роботи із подібними системами. Обмеження кастомізації під специфічні задачі користувача також є недоліком, оскільки деякі складні сценарії вимагають більшої адаптивності, ніж може запропонувати Definite.

1.1.2. Огляд програмного забезпечення MCDA Calculator

MCDA Calculator – це онлайн-інструмент для багатокритеріального аналізу рішень, що надає можливість оцінювати та ранжувати альтернативи на основі кількох

критеріїв. Він створений для спрощення складного процесу прийняття рішень завдяки доступному інтерфейсу та інтуїтивним функціям, що дозволяють легко взаємодіяти з даними. Основними методами, що підтримуються, є TOPSIS, PROMETHEE, MAVT, а також класичні підходи, як Weighted Sum Model (WSM) та інші подібні методи, орієнтовані на простоту і швидкість обчислень.

MCDA Calculator дозволяє користувачам визначати кількість критеріїв і альтернатив, призначати ваги для кожного критерію, а також оцінювати альтернативи відповідно до цих ваг. Результати аналізу подаються у вигляді інтерактивного ранжування альтернатив, що дає змогу легко зрозуміти оптимальний вибір. Інструмент також має опції для зміни ваг критеріїв і миттєвого оновлення результатів, що робить його зручним для сценаріїв із частими змінами умов. MCDA Calculator доступний через веб-браузер, що забезпечує доступність з будь-якого пристрою без потреби у встановленні програмного забезпечення.

Інтерфейс програми зображено на рисунку 1.2. Вигляд інтерфейсу MCDA Calculator.

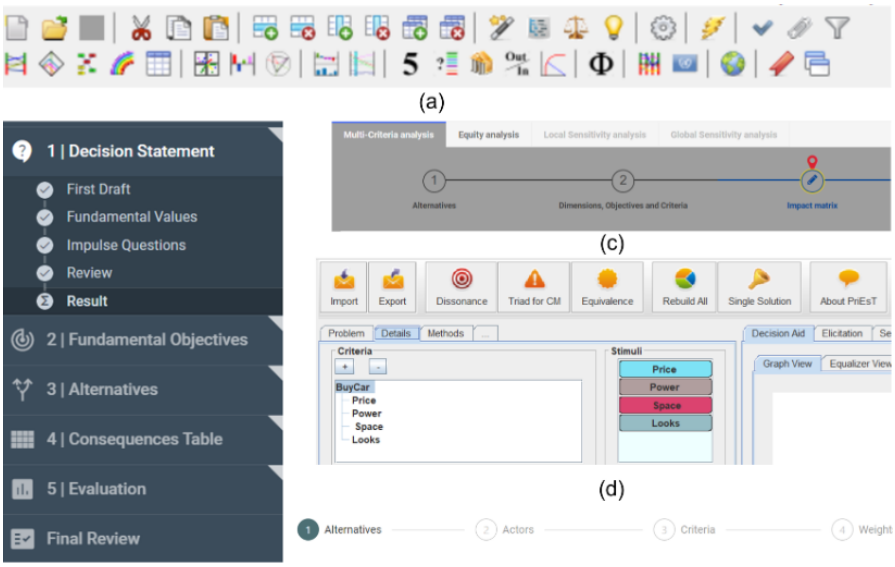


Рис.1.2. Вигляд інтерфейсу MCDA Calculator

Попри зручність і широкий функціонал, MCDA Calculator має свої обмеження. Інструмент підходить переважно для простих випадків, де критерії та альтернативи чітко визначені. Його структура не дозволяє кастомізувати алгоритми або інтегрувати

складніші методи, наприклад, нечітку логіку або багаторівневі ієрархії. Відсутність можливостей для групового прийняття рішень чи аналізу чутливості також обмежує його використання у складних проектах.

1.1.3. Огляд програмного забезпечення Expert Choice

Expert Choice – це одна з провідних програм для багатокритеріального прийняття рішень, що базується на методі аналізу ієрархій (AHP). Програма була створена в 1983 році Томасом Сааті та Ернестом Форманом і стала стандартом для багатокритеріального аналізу в таких галузях, як управління проектами, стратегічне планування, управління ризиками та оцінка капіталовкладень. Система використовується у виробництві, екологічному управлінні, суднобудуванні та аграрному секторі, забезпечуючи гнучкий і прозорий процес прийняття рішень.

Інтерфейс програми подано на рисунку 1.3. Вигляд інтерфейсу Expert Choice.

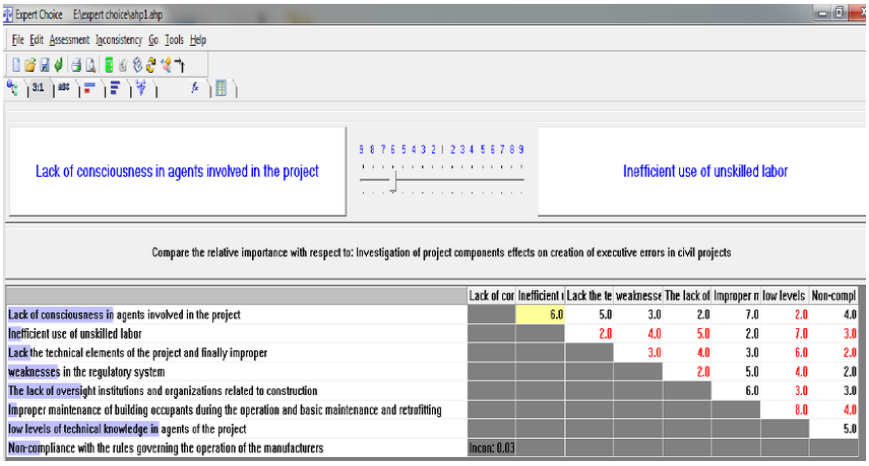


Рис.1.3. Вигляд інтерфейсу Expert Choice

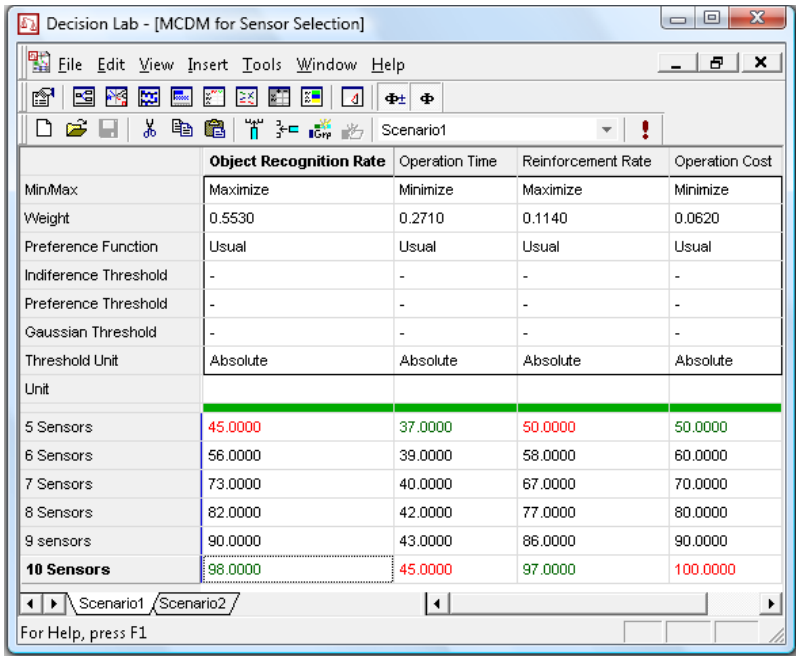
Програмне забезпечення Expert Choice дозволяє структурувати складні задачі у вигляді ієрархій, виконувати попарні порівняння критеріїв і альтернатив, а також аналізувати результати з використанням вбудованих засобів візуалізації. Основні функції включають розрахунок ваг, перевірку узгодженості матриць, аналіз чутливості, а також створення «що-якщо» сценаріїв для оцінки впливу змін на результати. Завдяки підтримці командної роботи програма полегшує колективне прийняття рішень, сприяючи інтеграції знань і пріоритетів усіх учасників.

Хоча програма має значний набір можливостей, вона також має свої недоліки. Для роботи з програмою часто потрібне навчання або технічна підготовка, оскільки інтерфейс містить складні функції. Крім того, система має обмежені можливості для кастомізації та інтеграції з іншими платформами, що може обмежувати її використання в нетипових або специфічних сценаріях.

1.1.4. Огляд програмного забезпечення Decision Lab

Decision Lab активно використовується для підтримки стратегічних рішень у різних галузях, таких як бізнес, екологія та управління ресурсами. Програма надає можливість моделювати складні рішення, враховуючи кілька критеріїв, оцінок і сценаріїв, з використанням таких методів, як PROMETHEE, ELECTRE та аналіз чутливості (sensitivity analysis). Її основною перевагою є інтуїтивно зрозумілий інтерфейс і широкий спектр інструментів для візуалізації даних, включаючи діаграми, GAIA-аналіз та інтерактивні графіки для оцінки результатів рішень.

Інтерфейс програми показано на рисунку 1.4. Вигляд інтерфейсу Decision Lab.



	Object Recognition Rate	Operation Time	Reinforcement Rate	Operation Cost
Min/Max	Maximize	Minimize	Maximize	Minimize
Weight	0.5530	0.2710	0.1140	0.0620
Preference Function	Usual	Usual	Usual	Usual
Indiference Threshold	-	-	-	-
Preference Threshold	-	-	-	-
Gaussian Threshold	-	-	-	-
Threshold Unit	Absolute	Absolute	Absolute	Absolute
Unit				
5 Sensors	45.0000	37.0000	50.0000	50.0000
6 Sensors	56.0000	39.0000	58.0000	60.0000
7 Sensors	73.0000	40.0000	67.0000	70.0000
8 Sensors	82.0000	42.0000	77.0000	80.0000
9 sensors	90.0000	43.0000	86.0000	90.0000
10 Sensors	98.0000	45.0000	97.0000	100.0000

Рис.1.4. Вигляд інтерфейсу Decision Lab

Програмне забезпечення дозволяє користувачам порівнювати альтернативи, проводити аналіз чутливості та досліджувати вплив змін у вагових коефіцієнтах на

ранжування критеріїв. Одна з ключових функцій – GAIA Plane, що візуалізує взаємозв’язки між критеріями, їхні конфлікти та вплив на альтернативи, допомагаючи користувачам краще розуміти структуру рішень. Крім того, функція Walking Weights дає змогу динамічно змінювати ваги критеріїв і миттєво спостерігати, як це впливає на результат.

Проте програма орієнтована на невелику кількість критеріїв та альтернатив, оскільки більші таблиці даних ускладнюють візуалізацію й аналіз. Деякі функції, наприклад GAIA Plane, можуть спричинити втрату інформації через проекцію багатовимірних даних на двовимірну площину. Крім того, складність певних алгоритмів та інтерфейсу вимагає додаткового навчання користувачів. Програма є комерційною і має обмеження щодо кастомізації для специфічних задач, а також інтеграції з іншими системами.

1.1.5. Огляд програмного забезпечення Super Decisions

Super Decisions базується на методах аналітичного ієрархічного процесу (AHP) і аналітичного мережного процесу (ANP). Прогрмне забезпечення розроблене під керівництвом Томаса Сааті, програма використовується для моделювання складних рішень, які потребують інтеграції як кількісних, так і якісних даних. Super Decisions особливо корисне для аналізу, де важливі фактори взаємопов’язані або мають зворотний зв’язок, наприклад, у бізнесі, управлінні проектами та державному секторі.

Програмне забезпечення дозволяє користувачам створювати ієрархічні та мережні моделі, виконувати попарні порівняння елементів і аналізувати результати за допомогою вбудованих засобів візуалізації. Однією з основних функцій є аналіз моделей типу BOCR (Benefits, Opportunities, Costs, Risks), що дозволяє враховувати різні аспекти прийняття рішень у складних сценаріях. Крім того, програма пропонує можливості для аналізу чутливості, що дозволяє оцінювати вплив змін у вагових коефіцієнтах на результати.

Завдяки гнучкому налаштуванню структури моделей програма підходить для аналізу складних задач, однак її багатофункціональність може ускладнювати процес

роботи з програмою для людей, які не знайомі з інтерфейсом.

Інтерфейс програми подано на рисунку 1.5. Вигляд інтерфейсу Super Decisions.

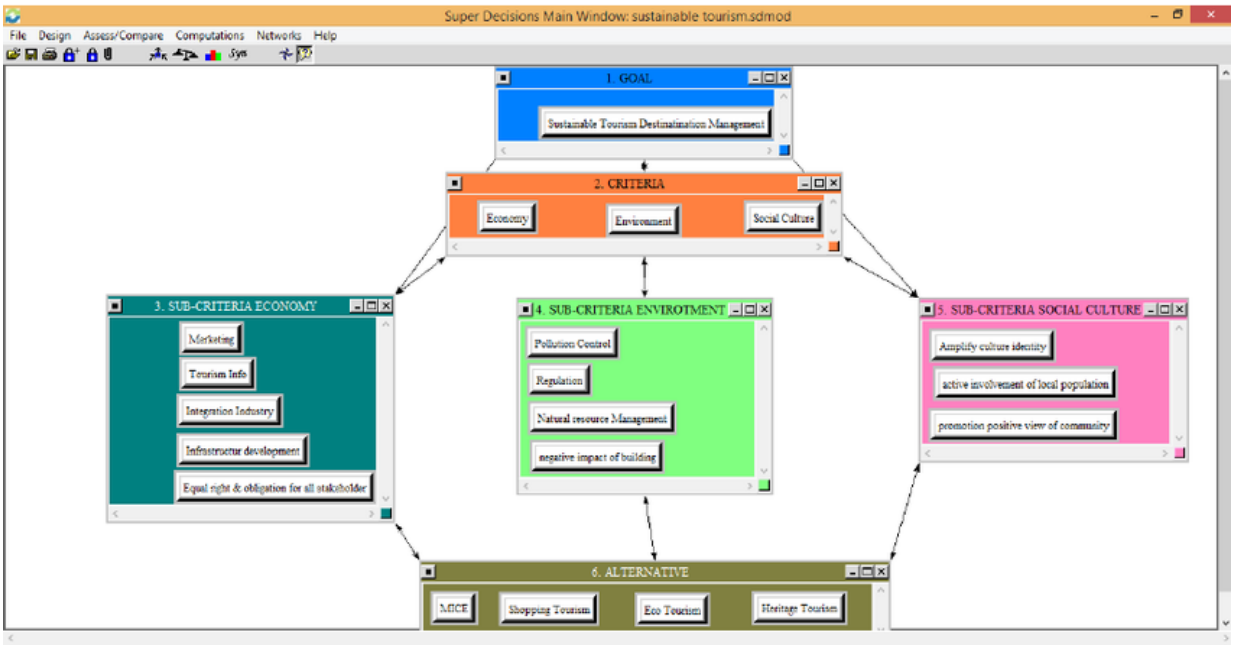


Рис.1.5. Вигляд інтерфейсу Super Decisions

Проте Super Decisions не позбавлена недоліків. Інтерфейс програми може бути складним для новачків і потребує навчання, щоб повністю використовувати її можливості. Відсутність гнучких опцій для інтеграції з іншими платформами також обмежує застосування в сучасних умовах. Ще одним недоліком є залежність від встановленого програмного забезпечення, що робить його менш адаптивним у порівнянні з веб-орієнтованими рішеннями.

1.1.6. Порівняння існуючих рішень

Аналіз п’яти програмних рішень для багатокритеріального прийняття рішень (MCDM) дозволяє виділити їхні особливості, переваги та недоліки. Definite/BOSDA відзначається широким вибором методів, таких як AHP, ELECTRE, і PROMETHEE, а також розвиненим інструментарієм для аналізу чутливості. Проте висока вартість та обмеження демоверсії знижують доступність для невеликих організацій чи окремих дослідників, що обмежує його застосування у практиці.

MCDA Calculator, навпаки, є безкоштовним та доступним у форматі веб-додатку, що робить його зручним для управлінських та економічних досліджень. Водночас інструмент має значні обмеження у кастомізації методів.

Expert Choice є потужним інструментом для колективного прийняття рішень, який активно використовується для стратегічного планування та оцінки ризиків. Його перевага полягає в підтримці сценаріїв «що-якщо» і складного аналізу чутливості. Однак висока вартість ліцензії та складність у використанні для новачків значно звужують потенційне коло користувачів.

Decision Lab зосереджується на інтерактивній візуалізації та аналізі залежностей за допомогою інструменту GAIA Plane. Це робить його зручним для стратегічного планування, але обмежена масштабованість і залежність від багатовимірних проекцій зменшують його ефективність при роботі з великими наборами даних.

Super Decisions вирізняється можливістю моделювання складних взаємозв’язків за допомогою методів AHP та ANP. Програма підтримує аналіз складних моделей, таких як BOCR, і забезпечує глибокий підхід до моделювання взаємопов’язаних критеріїв. Однак її інтерфейс є складним, і користувачі потребують значного часу для навчання, що знижує її зручність для широкого використання.

Порівняльна характеристика наведена в таблиці 1.1.

Таблиця 1.1

Порівняльна характеристика систем підтримки прийняття рішень

ПЗ	Основні методи	Особливості	Переваги	Недоліки
Definite/ BOSDA	AHP, ELECTRE, PROMETHEE, SMART	Інтерактивна візуалізація, аналіз чутливості, інтеграція з Excel, генерація звітів	Гнучкість вибору методів, широкі можливості для аналізу чутливості	Висока вартість, обмежена демоверсія, складність роботи з великими наборами даних

Продовження таблиці 1.1.

MCDA Calculator	TOPSIS, PROMETHEE, MAVT, WSM	Онлайн-доступ, динамічне оновлення результатів	Простота у використанні, доступність онлайн	Обмеження в кастомізації та інтеграції, залежність від інтернету
Expert Choice	AHP	Аналіз чутливості, колективне прийняття рішень, "що-якщо" сценарії	Широкий спектр інструментів для аналізу, підтримка колективних рішень	Висока вартість, складність для новачків
Decision Lab	PROMETHEE, ELECTRE, GAIA Plane	GAIA Plane для візуалізації залежностей, динамічне налаштування ваг	Інтуїтивний інтерфейс, інтерактивна візуалізація	Обмежена масштабованість, залежність від багатовимірної проекції
Super Decisions	AHP, ANP	Моделювання BOCR, аналіз ієрархій і мереж	Моделювання складних залежностей, детальний аналіз мереж	Складний інтерфейс, залежність від встановленого ПЗ

Аналіз існуючих рішень систем підтримки прийняття рішень показав, що багато з них мають обмеження, такі як складний інтерфейс та необхідність встановлення. Недостатня адаптивність до різних сценаріїв використання та неможливість інтеграції інших методів підтримки прийняття рішень також створюють труднощі для широкого впровадження таких систем. Ці недоліки підкреслюють необхідність розробки власного рішення, яке поєднує переваги кращих методів і надає гнучкість, доступність та зручність для користувачів. Розробка такої програми дозволить адаптувати інструмент під конкретні потреби та забезпечить ефективне використання новітніх технологій для багатокритеріального аналізу.

1.2. Огляд технологій, обраних для реалізації власного рішення

1.2.1. Мова розмітки HTML

HTML (HyperText Markup Language - мова розмітки гіпертексту) – це основний структурний елемент Інтернету. Вона визначає зміст і структуру веб-контенту. Інші технології, окрім HTML, зазвичай використовуються для опису зовнішнього вигляду веб-сторінки (CSS) або поведінки (JavaScript).

Веб-браузери отримують HTML-документи з веб-сервера або з локального сховища і перетворюють їх на мультимедійні веб-сторінки. HTML описує структуру веб-сторінки семантично і спочатку містить підказки для її зовнішнього вигляду.

HTML елементи є основними складовими веб-сторінок, слугуючи базою для створення їхньої структури. Вони дозволяють інтегрувати у відображувану сторінку зображення, інтерактивні форми та інші об’єкти. HTML забезпечує інструменти для побудови структурованих документів шляхом позначення семантичної структури тексту, включаючи заголовки, абзаци, списки, гіперпосилання, цитати та інші елементи.

Кожен HTML елемент визначається за допомогою тегів, що записуються у формі кутових дужок. Наприклад, теги на зразок <img> і <input> вставляють контент безпосередньо у веб-сторінку, тоді як теги, такі як та , окреслюють текст документа, надаючи йому семантичний контекст і можуть містити вкладені елементи. Браузери не відображають теги HTML, але використовують їх для інтерпретації вмісту сторінки.

Теги можуть також містити вкладені теги й текст, що формують дочірні елементи у межах батьківського елемента. Відкриваючий тег може включати атрибути, які надають додаткову інформацію, наприклад, ідентифікатори секцій, стилі або посилання на ресурси, як у випадку з тегом <img>: <img src="example.com/example.jpg">.

У багатьох тегах, таких як , закриваючий тег є необов’язковим, оскільки браузери можуть визначати завершення елемента на основі контексту та структурних правил HTML.

1.2.2. Мова стилів CSS

CSS (Cascading Style Sheets) – це мова стилів, яка визначає зовнішній вигляд і форматування документів, створених за допомогою мов розмітки, таких як HTML чи XML. Вона є однією з базових технологій Всесвітньої павутини разом із HTML та JavaScript.

CSS забезпечує відокремлення вмісту від його візуального оформлення, що включає макет, кольори, шрифти й інші стилістичні аспекти. Такий підхід спрощує створення доступного контенту, полегшує управління стилями та дозволяє використовувати окремі CSS-файли для спільного форматування кількох сторінок. Це, своєю чергою, зменшує складність коду, уникаючи дублювання, і прискорює завантаження сторінок завдяки кешуванню.

Ця технологія дозволяє адаптувати сторінки для різних середовищ відображення, таких як екрани, друковані носії, голосові браузери чи пристрої Брайля. CSS також підтримує правила для адаптивного дизайну на мобільних пристроях.

Назва "каскадні" пояснюється тим, що в CSS використовується пріоритетна схема, яка визначає застосування стилів у випадках конфлікту кількох декларацій. Ця схема є систематичною та передбачуваною.

Синтаксис CSS досить простий і використовує кілька англійських ключових слів для визначення назв різних властивостей стилю. Таблиця стилів складаються з певних правил. Кожне правило включає один або кілька селекторів (англ. selectors) та блок декларацій (англ. declaration block). Блок декларацій містить перелік властивостей, які визначають зовнішній вигляд елементів, і є укладеним у фігурні дужки. Кожна властивість у деклараційному блоці записується у форматі: назва властивості, двокрапка (:), значення властивості, крапка з комою (;).

До появи CSS майже всі презентаційні атрибути HTML-документів містилися в розмітці HTML. Усі кольори шрифтів, стилі фону, вирівнювання елементів, межі та розміри мали бути явно описані, часто багаторазово, в HTML. CSS дозволяє авторам перенести більшу частину цієї інформації в інший файл, таблицю стилів, що значно

спрощує HTML. Крім того, оскільки все більше пристроїв мають доступ до адаптивних веб-сторінок, починають з'являтися різні розміри екранів і макети.

1.2.3. Мова програмування JavaScript

JavaScript (JS) – це мова програмування, що є однією з основних технологій веб-розробки. 99% вебсайтів використовують JavaScript для взаємодії на стороні клієнта. Спеціалізовані рушії JavaScript виконують клієнтський код у веб-браузерах, а також використовуються на деяких серверах і в різних додатках.

JavaScript є мовою високого рівня, яка зазвичай компілюється під час виконання і відповідає стандарту ECMAScript. Вона є динамічно типізованою та об'єктно-орієнтованою. JavaScript має API для роботи з текстами, датами, регулярними виразами, стандартними структурами даних та Об'єктною моделлю документа (DOM).

Рушій JavaScript – це програмний компонент для виконання коду JavaScript. Спочатку це були прості інтерпретатори, але сучасні рушії використовують компіляцію «just-in-time» для підвищення продуктивності.

Рушії JavaScript розробляються виробниками браузерів, і кожен основний браузер має свій рушій. Вони працюють разом з рушієм рендерингу через об’єктну модель документа (DOM).

JavaScript є однопотоковою мовою програмування, що означає обробку повідомлень у черзі по одному за раз. Кожне нове повідомлення викликає відповідну функцію, створюючи стек викликів з аргументами функції та локальними змінними. Стек змінюється відповідно до необхідності функції. Після завершення виконання функції та порожнього стеку JavaScript переходить до обробки наступного повідомлення. Цей процес відомий як цикл обробки подій і описується принципом «виконання до завершення», оскільки кожне повідомлення обробляється повністю перед тим, як перейти до наступного.

Модель паралельності мови передбачає неблокуючий цикл подій, що дозволяє обробляти асинхронні завдання, як, наприклад, обробка подій клацання миші під час очікування на відповіді від бази даних.

JavaScript підтримує багато елементів синтаксису структурованого програмування з мови C, таких як умовні оператори (if), цикли (while, do while) та конструкції switch.

Подібно до C, JavaScript відрізняє вирази від операторів. Одна з синтаксичних відмінностей полягає в автоматичному вставленні крапок з комою, що дозволяє їх опускати в кінці операторів.

JavaScript є динамічно типізованою мовою, подібно до більшості інших сценарних мов. Тип зв’язується з конкретним значенням, а не з виразом. Наприклад, змінна, яка спочатку містить число, може бути переназначена як рядок. JavaScript також підтримує різні способи перевірки типу об'єктів, включаючи принцип "duck typing" (перевірка типу за поведінкою об’єкта).

Скрипти на JavaScript можуть реалізувати такі функції:

1) Завантаження нових даних без перезавантаження сторінки через Ajax або WebSocket.
2) Анімації веб-сторінок, такі як зміна розміру, переміщення та згасання елементів.
3) Керування потоковим медіа.
4) Спливаючі вікна або повідомлення.
5) Перевірка даних форми перед відправкою на сервер.
6) Реєстрація користувацької активності для аналітики та персоналізації.
7) Перенаправлення на інші сторінки.
8) Зберігання даних на пристрої користувача за допомогою стандартів зберігання або IndexedDB.

В JavaScript об’єкт є асоціативним масивом. Кожен ключ в об'єкті є назвою властивості, яку можна вказати двома способами: через крапкову нотацію (obj.x = 10) або через нотацію з дужками (obj['x'] = 10). Властивості можна додавати, змінювати або видаляти під час виконання. Більшість властивостей об'єкта, а також властивості, що належать до ланцюга спадкування його прототипу, можна перераховувати за допомогою циклу for...in.

JavaScript і DOM створюють можливість для зловмисників запускати шкідливі скрипти на клієнтському комп'ютері через Інтернет. З метою мінімізації цього ризику розробники браузерів впроваджують два ключові обмеження. По-перше, скрипти працюють у контрольованому середовищі («пісочниці»), де можуть виконувати лише специфічні для веб-середовища дії, уникаючи загальних завдань програмування, як-от створення файлів. По-друге, діє політика однакового походження, яка забороняє скриптам з одного веб-сайту доступ до даних (паролів, cookie тощо) іншого сайту. Більшість проблем безпеки JavaScript пов’язані з порушенням цих обмежень.

1.2.4. Бібліотека візуалізації даних Chart.js

Chart.js – це безкоштовна відкрита JavaScript-бібліотека для візуалізації даних, яка підтримує вісім типів діаграм: стовпчикові, лінійні, площинні, кругові, бульбашкові, радарні, полярні та точкові. Її створив лондонський розробник Нік Дауні у 2013 році, а на сьогоднішній день бібліотека підтримується спільнотою.

Chart.js є другою за популярністю бібліотекою для побудови графіків на GitHub після D3.js. Вона вважається значно простішою у використанні, хоча й менш налаштованою. Рендеринг відбувається за допомогою елемента HTML5 canvas. Це робить Chart.js високопродуктивним, особливо для роботи з великими наборами даних і складними візуалізаціями, які інакше потребували б тисячі SVG-вузлів у DOM-дереві. Водночас використання Canvas не дозволяє застосовувати стилізацію через CSS, тому для налаштування зовнішнього вигляду необхідно використовувати вбудовані опції або створювати власний плагін чи тип діаграми.

Chart.js часто розглядають як одну з найкращих бібліотек для візуалізації даних, і вона доступна на умовах ліцензії MIT.

1.2.5. Бібліотека для формування PDF-файлів pdfMake

Бібліотека pdfMake пропонує широкий набір функцій для створення PDF-документів, що робить її потужним інструментом для роботи з документами в JavaScript. Однією з ключових особливостей є можливість детального форматування тексту: розробники можуть змінювати шрифти, їхній розмір, стиль, кольори, а також

вирівнювати текст у документі. PdfMake також підтримує створення таблиць із розширеним функціоналом, наприклад, автоматичне перенесення заголовків таблиці на наступні сторінки документа. Це значно спрощує створення багатосторінкових PDF із великими таблицями.

Окрім цього, бібліотека дозволяє створювати марковані та нумеровані списки з гнучкими параметрами, що є особливо корисним для створення структурованих текстів або презентацій. Інтеграція зображень також є ключовою функцією, оскільки вона дозволяє вбудовувати зображення у форматах PNG та JPEG. У випадку використання на серверній стороні (з Node.js) можна додавати зображення із зовнішніх джерел.

Інтеграція шрифтів із зовнішніх джерел, таких як Google Fonts, або використання власних шрифтів дає змогу створювати документи, які відповідають специфічному дизайну. Завдяки цим функціям pdfMake стала популярним вибором серед розробників для створення PDF-документів із високим рівнем кастомізації.

1.2.6. Бібліотека html2canvas

Html2canvas – це бібліотека JavaScript, яка дозволяє генерувати «скріншоти» HTML-контенту безпосередньо в браузері користувача. На відміну від традиційних скріншотів, html2canvas читає структуру DOM і застосовані до елементів CSS стилі, перетворюючи їх на зображення в елементі canvas. Цей підхід дозволяє створювати зображення повністю на стороні клієнта без потреби у серверному рендерингу.

Бібліотека корисна для захоплення візуальних зображень веб-елементів і пропонує різні налаштування для коригування результатів рендерингу. Її також можна використати для завдань, таких як створення звітів, захоплення стану інтерфейсу або візуалізація DOM елементів як зображень.

1.2.7. Фреймворк Bootstrap

Bootstrap (раніше Twitter Bootstrap) – це безкоштовний CSS-фреймворк з відкритим вихідним кодом, призначений для адаптивної веб-розробки інтерфейсів, орієнтованих на мобільні пристрої. Він містить шаблони дизайну на основі HTML,

CSS та JavaScript для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу.

Основна мета додавання Bootstrap до веб-проекту – застосувати до нього вибір кольорів, розмірів, шрифтів та макетів Bootstrap. Таким чином, головним фактором є те, чи сподобаються ці варіанти розробникам.

Після додавання до проекту Bootstrap надає базові визначення стилів для всіх елементів HTML. Результатом є однаковий вигляд тексту, таблиць та елементів форм у всіх веб-браузерах. Крім того, розробники можуть скористатися класами CSS, визначеними в Bootstrap, щоб додатково налаштувати зовнішній вигляд свого вмісту. Наприклад, у Bootstrap передбачені світлі та темні таблиці, заголовки сторінок, більш помітні цитати та текст з підсвічуванням.

Bootstrap також включає декілька компонентів JavaScript, які не потребують інших бібліотек, таких як jQuery. Вони надають додаткові елементи інтерфейсу користувача, такі як діалогові вікна, підказки, індикатори виконання, навігаційні випадаючі списки та каруселі. Кожен компонент Bootstrap складається зі структури HTML, оголошень CSS і, в деяких випадках, супровідного коду JavaScript. Вони також розширюють функціональність деяких існуючих елементів інтерфейсу, включаючи, наприклад, функцію автозаповнення полів введення.

Найважливішими компонентами Bootstrap є компоненти верстки, оскільки вони впливають на всю веб-сторінку. Основний компонент макета називається контейнер (Container), оскільки в ньому розміщуються всі інші елементи сторінки.

Завдяки розгорнутій документації та великій спільноті розробників, Bootstrap став одним із найбільш популярних інструментів для веб-розробки, що дозволяє зменшити час на реалізацію проектів і покращити продуктивність команд розробників.

У порівнянні з іншими фреймворками, Bootstrap є зручним для початківців завдяки простоті у використанні та великій кількості вбудованих шаблонів і компонентів, але також достатньо гнучким для досвідчених розробників, які хочуть налаштувати дизайн під специфічні потреби.

1.2.8. Мова програмування Python

Python – це високорівнева мова програмування загального призначення. Її філософія проектування підкреслює читабельність коду з використанням значних відступів.

Python підтримує багато парадигм програмування, зокрема об’єктно-орієнтоване, структуроване, функціональне програмування, а також аспекти метапрограмування і аспектно-орієнтованого програмування. Мова використовує динамічну типізацію і комбінацію лічильника посилань та циклічного збирання сміття для управління пам'яттю. Вона також підтримує динамічне зв'язування імен, що дозволяє прив'язувати методи та змінні під час виконання програми, що забезпечує високу гнучкість.

Основна ідея Python полягає в тому, що це розширювана мова з невеликим ядром і великою стандартною бібліотекою. Це дозволяє Python бути зручним інструментом для інтеграції з іншими мовами і додатками. Python має на меті забезпечити «однозначний і очевидний» спосіб вирішення задачі, хоча на практиці існує кілька способів досягти однакових результатів.

Python відрізняється своєю орієнтованістю на читабельність коду. Форматування коду в Python є візуально впорядкованим, а для вираження операцій використовуються англійські ключові слова, що сприяє простоті розуміння. На відміну від багатьох інших мов програмування, Python не застосовує фігурні дужки для позначення блоків коду, а крапки з комою після операторів дозволені, але практично не використовуються. Крім того, синтаксис Python має менше виключень і спеціальних випадків порівняно з такими мовами, як C або Pascal.

У Python оператор присвоєння (=) не просто зберігає значення в змінній, як це може бути у деяких інших мовах програмування. Замість цього він прив’язує ім’я змінної до об'єкта, який створюється в пам’яті. Це означає, що змінну можна переназначити на інший об’єкт у будь-який момент. У Python змінна не має фіксованого типу даних, але завжди посилається на об’єкт із конкретним типом. Такий підхід називається динамічним типізуванням, на відміну від статично типізованих мов, де змінна може зберігати лише значення певного типу.

Попри динамічну типізацію, Python є строго типізованою мовою, тому невідповідні операції (наприклад, додавання числа до рядка) викликають помилки, а не виконуються автоматично.

Стандартна бібліотека Python, одна з ключових переваг цієї мови, забезпечує широкий спектр інструментів для різноманітних завдань. Вона підтримує стандартні формати та протоколи, зокрема MIME (Multipurpose Internet Mail Extensions) і HTTP (Hypertext Transfer Protocol), та включає модулі для створення графічних інтерфейсів, підключення до реляційних баз даних, генерування псевдовипадкових чисел, виконання арифметичних операцій із числами довільної точності, роботи з регулярними виразами та модульного тестування.

Широкий вибір інструментів робить Python зручним як для новачків, так і для досвідчених розробників у різних галузях, від веб-розробки до наукових досліджень.

Для створення та підтримки складних веб-застосунків Python пропонує різноманітні фреймворки, серед яких найбільш відомі Django, Flask, Pyramid, TurboGears, web2py, Tornado, Bottle та Zope. Кожен із них має свої унікальні функції для розробки спеціалізованих рішень.

У галузі штучного інтелекту та машинного навчання Python популярний завдяки таким бібліотекам, як TensorFlow, Keras, PyTorch, scikit-learn. Їх використовують для моделювання, аналізу даних і розробки інтелектуальних систем. Для обробки природної мови Python також добре підходить через простий синтаксис, модульну архітектуру та потужні інструменти для роботи з текстом, які роблять його ідеальним для задач лінгвістичного аналізу та генерації тексту.

Python також можна використовувати для створення графічного інтерфейсу користувача (GUI) за допомогою таких бібліотек, як Tkinter.

1.2.9. Вебфреймворк Flask

Flask — це легкий та гнучкий вебфреймворк для Python, часто описуваний як «мікрофреймворк» завдяки своєму мінімалістичному підходу. Він спроектований таким чином, щоб надавати лише основні компоненти, необхідні для веб-розробки, пропонуючи лише ті функції, які важливі для створення веб-додатків. Flask не

нав’язує структуру проекту, даючи розробникам повний контроль над архітектурою їхнього додатку. Такий мінімалізм дозволяє інтегрувати додаткові бібліотеки та компоненти за потреби, що робить його ідеальним для різних типів проектів – від невеликих додатків до великих і складних веб-сервісів, коли розробники хочуть мати більше контролю над структурою.

Основою Flask є інструментарій WSGI (Web Server Gateway Interface) від Werkzeug, який забезпечує сумісність з вебсерверами та ефективно обробляє HTTP-запити. Цей інструментарій включає в себе такі функції, як маршрутизація URL, обробка запитів і відповідей, а також управління сесіями. Flask також інтегрує Jinja2 – потужну систему шаблонів, яка дозволяє розробникам створювати динамічний HTML-контент, вбудовуючи вирази та логіку, подібну до Python, у HTML-файли. Це дозволяє створювати інтерактивні вебсторінки. Завдяки Jinja2 розробники можуть використовувати розширені функції, такі як цикли, умови та наслідування шаблонів, що робить Flask ефективним інструментом для розробки динамічних вебдодатків.

Flask є гнучким та мінімалістичним. На відміну від інших фреймворків, які накладають суворі обмеження, Flask дозволяє розробникам створювати додатки, які відповідають їхнім конкретним потребам і вподобанням. Він не вимагає дотримання певних директорій або попередньо визначених структур файлів, що робить його ідеальним для тих розробників, які хочуть побудувати індивідуальну архітектуру свого додатку. Ця гнучкість означає, що Flask можна адаптувати під різні випадки, незалежно від того, наскільки складну систему потрібно реалізувати.

Модульна архітектура Flask є однією з його головних переваг, оскільки розробники можуть інтегрувати розширення та сторонні бібліотеки відповідно до специфічних вимог проекту. Наприклад, Flask підтримує використання Flask-SQLAlchemy для роботи з базами даних, Flask-WTF для обробки веб-форм і Flask-Login для автентифікації користувачів. Цей підхід дозволяє вибирати лише необхідні функції, підтримуючи легкість додатка та надаючи йому індивідуальність. Flask також надає вбудований сервер для розробки та налагодження, що спрощує процес розробки та тестування завдяки миттєвому зворотному зв’язку під час змін у коді.

Незважаючи на свою простоту, Flask здатний ефективно масштабуватися, обробляючи як малі проєкти, так і великі складні системи. Мінімалістичний дизайн фреймворку гарантує, що додатки залишаються легкими навіть при збільшенні їх складності. Модульна природа Flask означає, що додаток може розвиватися разом із проєктом, дозволяючи додавати функціональність без порушення структури.

Офіційна документація є вичерпною для користувача. Вона надає численні ресурси для розробників різного рівня. Крім того, спільнота Flask створила безліч розширень та бібліотек, які ще більше розширюють можливості фреймворку. Екосистема, створена навколо Flask, гарантує, що розробники мають всі необхідні інструменти для побудови складних додатків. Широке використання фреймворку в індустрії та його постійний розвиток забезпечують, що Flask залишається надійним і актуальним варіантом для веб-розробки.

1.2.10. Редактор коду Visual Studio Code

Visual Studio Code (VS Code) – це багатофункціональний редактор коду з відкритим вихідним кодом, розроблений корпорацією Microsoft. Він поєднує легкість використання з потужними інструментами для професійної розробки, що робить його популярним серед розробників усіх рівнів.

VS Code доступний для Windows, macOS і Linux, забезпечуючи однаковий досвід користування на всіх основних операційних системах. Його компактність гарантує швидке встановлення та запуск навіть на малопотужних пристроях. Базові функції включають підсвічування синтаксису для багатьох мов програмування, IntelliSense для інтелектуальних підказок, вбудований термінал і глибоку інтеграцію з Git для контролю версій. Ці можливості значно прискорюють розробку проєктів будь-якої складності.

Особливу увагу заслуговує інтегрована функція налагодження. Розробники можуть ставити точки зупину, переглядати виклики стеків і відслідковувати значення змінних у реальному часі. Така інтеграція зменшує залежність від зовнішніх інструментів і спрощує пошук помилок як у клієнтському, так і серверному коді.

Головною перевагою VS Code є його розширюваність через велику кількість плагінів. Вони охоплюють підтримку мов програмування, засоби аналізу коду, інтеграцію з хмарними платформами, інструменти для контейнеризації (Docker, Kubernetes) тощо.

На відміну від традиційних інтегрованих середовищ розробки (IDE), таких як Eclipse або Visual Studio, VS Code пропонує ідеальний баланс між функціональністю та продуктивністю. Він забезпечує передові можливості, притаманні IDE, без зайвого навантаження на систему. Активна спільнота розробників також сприяє постійному вдосконаленню редактора, забезпечуючи доступ до багатьох навчальних ресурсів і рекомендацій.

1.2.11. Система контролю версій Git

Git – це розподілена система контролю версій, призначена для ефективного відстеження змін у файлах, зокрема у вихідному коді, що використовується в командній розробці програмного забезпечення. Вона була створена Лінусом Торвальдсом у 2005 році для управління розробкою ядра Linux.

Git створений із фокусом на швидкість, цілісність даних і підтримку розподілених нелінійних робочих процесів. Система дозволяє працювати з тисячами паралельних гілок, які можуть існувати на різних комп’ютерах, забезпечуючи гнучкість у командній роботі. На відміну від централізованих систем, Git використовує локальні копії репозиторіїв, які включають повну історію змін, що дозволяє працювати автономно, без доступу до мережі чи центрального сервера.

Можливості Git значно розширюються завдяки платформам для хостингу репозиторіїв, таким як GitHub, GitLab, Bitbucket і SourceForge. Ці сервіси пропонують інструменти для командної роботи, автоматизації процесів CI/CD (безперервна інтеграція та доставка), а також управління проектами, що робить Git невід’ємною частиною сучасного процесу розробки.

Git є важливим не тільки як технічний інструмент, але і як механізм, що сприяє співпраці у сфері розробки програмного забезпечення. Його вплив охоплює проекти

різного масштабу: від відкритих ініціатив до корпоративних програм, роблячи його незамінною складовою екосистеми розробки.

1.3. Висновок до розділу

У цьому розділі було проведено всебічний аналіз сучасних програмних рішень у сфері багатокритеріального прийняття рішень, а також виконано огляд технологій, що лежать в основі їхньої розробки. Здійснено огляд існуючих підходів до створення систем підтримки прийняття рішень, з акцентом на їхню здатність забезпечувати структурований і гнучкий процес вибору в умовах складних багатокритеріальних задач.

Особливу увагу було приділено характеристикам і функціональності цих систем, їх перевагам та обмеженням. У ході аналізу було виявлено ключові тенденції в розробці таких рішень, що демонструють важливість інтеграції сучасних методів аналізу та візуалізації даних. Окрім цього, розглянуто можливості використання сучасних технологій, таких як Python, Flask, Bootstrap та інших, що сприяють створенню доступних і адаптивних інструментів для широкого кола користувачів.

РОЗДІЛ 2
АЛГОРИТМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

2.1. Основні поняття теорії прийняття рішень

Теорія прийняття рішень описує структурований процес, що включає кілька ключових етапів:

- визнання необхідності прийняття рішення;
- визначення цілей;
- складання списку можливих варіантів;
- аналіз позитивних і негативних наслідків кожного варіанту;
- оцінка бажаності цих наслідків;
- визначення ймовірності їх реалізації;
- інтеграція отриманої інформації.

Весь процес відбувається в контексті певної ситуації, яка впливає на доступні варіанти та їхні наслідки.

У реальному житті люди часто ухвалюють рішення інтуїтивно, керуючись звичками чи традиціями, без систематичного проходження цих етапів. Соціальний тиск або обмеження в часі можуть заважати ретельному аналізу варіантів і наслідків. Емоційний стан, недостатність інформації чи навичок також здатні негативно впливати на якість рішень. Навіть коли є достатньо часу та даних, люди нерідко роблять помилки у розумінні ймовірностей наслідків або надають перевагу особистому досвіду, а не статистичним даним.

Рішення – це результат вибору з кількох альтернатив, який фіксується у певній формі. На початкових етапах може бути відсутній чітко визначений набір альтернатив, між якими необхідно зробити вибір. Наприклад, у випадку обміну квартири потрібно спершу визначити потенційні варіанти. Множина альтернатив спочатку є аморфною, тобто без структури, що ускладнює оцінку їхньої якості. Формування структури множини дозволяє забезпечити більш ефективний вибір.

Категорії рішень включають різні типи, які відрізняються за характером проблем і підходами до їх розв’язання. Організаційні рішення стосуються виконання посадових обов’язків, спрямованих на досягнення цілей організації. Їхня ефективність вимірюється внеском у кінцевий результат.

Програмовані рішення базуються на чітких алгоритмах чи стандартних процедурах, що забезпечує зменшення помилок і економію часу. Такі рішення характерні для технічних задач, які мають повторюваний характер. В свою чергу, непрограмовані рішення використовуються в ситуаціях, які є новими або пов’язаними з невизначеністю, що вимагає від осіб, які приймають рішення, ініціативності та високого рівня компетенції.

Інтуїтивні рішення є результатом суб’єктивних відчуттів і часто застосовуються у складних ситуаціях, де логічний аналіз може бути ускладнений. Натомість рішення, засновані на міркуваннях, використовують знання та досвід, накопичений у схожих обставинах, дозволяючи прогнозувати результати вибору на основі попередніх успішних прикладів. Раціональні рішення, на відміну від обґрунтованих на міркуваннях, базуються на об’єктивному аналізі даних, а не на минулому досвіді.

Рішення можна поділити на групи за різними критеріями:

За масштабом об’єкта розрізняють глобальні рішення, які охоплюють широкий спектр проблем на рівні організації чи галузі, та локальні, що стосуються окремих підрозділів чи обмежених завдань.

За характером мети рішення поділяються на стратегічні, спрямовані на досягнення довгострокових цілей, та тактичні, орієнтовані на короткострокові завдання.

За джерелом виникнення виділяють рішення за розпорядженням (наказом керівництва), ініціативні (запропоновані співробітниками), та рішення за замовленням, виконувані за запитом сторонніх осіб чи організацій.

За способом доведення рішення можуть бути усними, що передаються безпосередньо під час комунікації, та письмовими, оформленими у вигляді документів.

За суб’єктом прийняття рішення можуть бути індивідуальним (прийнятим однією особою), колективним (група осіб, які приймають рішення разом), або колегіальним, де рішення ухвалюються спеціальною радою чи комісією.

За ступенем новизни виділяють традиційні рішення, що ґрунтуються на попередньому досвіді, та новаційні, які вносять суттєві зміни у процеси чи структуру організації.

За методами розробки рішення можуть бути кількісними, побудованими на використанні математичних методів і моделей, або евристичними, які спираються на інтуїцію та творчі підходи.

За інформаційною базою рішення можуть бути визначеними (за умови повної інформації), ймовірнісними (з частковою інформацією), або невизначеними (за умов відсутності точних даних).

За цільовою спрямованістю рішення поділяються на одноцільові, що спрямовані на досягнення єдиної мети, та багатоцільові, орієнтовані на одночасне досягнення кількох цілей.

За змістом рішення можуть бути економічними, технічними, соціальними чи організаційними, що охоплюють різні аспекти діяльності.

За тривалістю дії виділяють довготривалі рішення, які впливають на організацію в перспективі, та оперативні, що реалізуються негайно.

За станом свідомості рішення можуть бути усвідомленими (прийнятими на основі повного розуміння ситуації), малоусвідомленими (з частковим аналізом) та неусвідомленими (імпульсивними чи інтуїтивними).

Процес прийняття рішення вимагає врахування різних елементів, таких як проблемна ситуація, наявність варіантів, обмеження ресурсів і зовнішнього середовища. Кожен із цих факторів впливає на результат і формує основу для вибору оптимального підходу до розв’язання проблеми.

Процес прийняття рішень можна поділити на кілька ключових етапів. Вони взаємопов’язані і вимагають чіткого розуміння контексту задачі та доступних ресурсів. Структурований підхід до прийняття рішень дозволяє не лише знайти оптимальне рішення, але й підвищити прозорість та обґрунтованість обраної

стратегії.Схема процесу прийняття рішень зображена на рис. 2.1. Основні етапи процесу прийняття рішень.

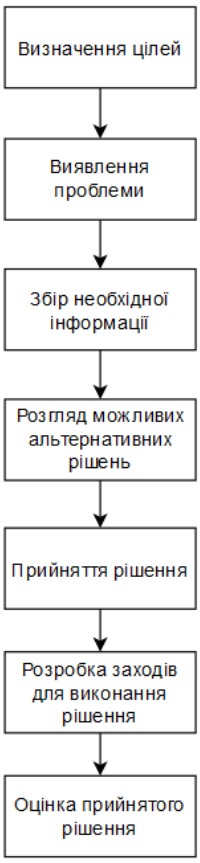


Рис. 2.1. Основні етапи процесу прийняття рішень.

Першим кроком є визначення мети, тобто формулювання завдання, яке необхідно виконати. Метою може бути збільшення прибутку, розширення масштабів виробництва, скорочення витрат чи покращення якості продукції. Чітке формулювання мети слугує основою для подальших дій.

Наступний етап – виявлення проблеми, тобто аналіз перешкод, які заважають досягненню мети.

Для пошуку шляхів вирішення проблеми необхідно здійснити збір інформації про причини виникнення проблеми та можливі шляхи їх усунення. Збір даних потребує витрат часу і ресурсів, які мають бути відповідати потенційній вигоді від досягнення мети.

Етап розгляду альтернатив базується на аналізі зібраної інформації та формулюванні кількох варіантів вирішення проблеми. Наприклад, у разі необхідності зниження витрат можна запропонувати різні підходи, такі як вдосконалення процесів або оптимізація ресурсів.

Наступним кроком є прийняття рішення, тобто вибір найкращої альтернативи. Цей етап пов’язаний із труднощами, такими як нестача інформації чи необхідність сміливості для прийняття відповідальності. Процес ухвалення рішень залежить від особистих якостей людини, що приймає рішення.

Розробка заходів для виконання рішення передбачає планування дій, розподіл обов'язків і ресурсів.

Останнім етапом є оцінка результатів ухваленого рішення. Аналіз фактичних результатів допомагає визначити ефективність вибраного підходу та внести корективи для майбутніх рішень. Наприклад, оцінка може включати аналіз собівартості продукції, продуктивності праці чи прибутковості.

2.2. Класичні критерії прийняття рішень та їх застосування

2.2.1. Критерій Вальда

Критерій Вальда, також відомий як критерій максиміна, є одним із ключових методів прийняття рішень в умовах невизначеності. Цей критерій орієнтований на песимістичну стратегію, де метою є мінімізація максимально можливих втрат або, відповідно, максимізація мінімального виграшу. Він підходить для ситуацій, коли доступна інформація про ймовірності результатів обмежена або відсутня, і приймаюча рішення особа прагне уникнути найгіршого сценарію.

Основою критерію Вальда є принцип, який полягає в обчисленні мінімального значення виграшу або найгіршого результату для кожної альтернативи. Серед цих мінімальних значень обирається альтернатива, яка має найбільше значення.

Це можна описати формулою:

$$К_{мм} = \max \min a_{ij} \qquad (2.1)$$

Критерій фокусується на найгіршому можливому сценарії для кожної альтернативи, що робить його корисним у ситуаціях із високою невизначеністю або обмеженими ресурсами для додаткового аналізу. Цей підхід є консервативним і підходить для рішень, де ризики дуже високі, і помилки можуть мати серйозні наслідки. Метод не вимагає складних обчислень або інформації про ймовірності.

Однак в критерію Вальда є свої недоліки. Він може не враховувати вигоди, які можна отримати за сприятливих обставин. Також він може бути надто обережним у ситуаціях, де ризик є прийнятним.

Критерій Вальда часто використовується в економіці, фінансах, управлінні ризиками та інших сферах, де рішення мають прийматися за відсутності повної інформації про майбутнє.

2.2.2. Критерій азартного гравця

Критерій азартного гравця, або максимаксний критерій, є одним із підходів до прийняття рішень в умовах невизначеності. Його основна ідея полягає у виборі альтернативи, яка має потенціал забезпечити найбільший можливий виграш у найкращих умовах. Цей підхід ґрунтується на припущенні, що зовнішнє середовище сприятиме досягненню максимально можливого результату для кожної з альтернатив.

Для реалізації критерію азартного гравця створюється таблиця (матриця), де рядки представляють альтернативи, а стовпці – можливі стани середовища. Елементи матриці відображають виграші або результати відповідної альтернативи за конкретного стану середовища. Для кожного рядка матриці визначається максимальне значення виграшу. Це максимальний виграш, який може бути досягнутий у найсприятливішому стані зовнішнього середовища. Серед знайдених максимальних виграшів обирається найбільше значення. Відповідна альтернатива вважається оптимальною згідно з критерієм азартного гравця. Формула для цього критерію має наступний вигляд:

$$К_{мм} = \max \min a_{ij} \qquad (2.2)$$

Даний критерій ідеально підходить для ситуацій, коли особа, яка приймає рішення, орієнтується на досягнення максимального результату й налаштована оптимістично.

Критерій максимаксу добре підходить для рішень у ситуаціях, де позитивний результат є критичним, а невдача менш важлива. Метод є інтуїтивно зрозумілим і легко застосовним навіть у великих задачах, якщо є доступ до матриці виграшів.

Проте використовуючи критерій азартного гравця, варто пам’ятати, що при виборі враховується лише найкращий сценарій, а можливі негативні наслідки ігноруються. Це може призвести до вибору ризикованих альтернатив. Якщо виграші в інших станах суттєво відрізняються, ризик програшу значно зростає. Також критерій не враховує ймовірності реалізації різних станів зовнішнього середовища, що може знизити його практичну цінність у деяких випадках.

Хоча цей критерій є простим і ефективним у використанні, його недоліки, пов’язані з ігноруванням ризиків і варіабельності результатів, роблять його менш підходящим у ситуаціях із високим ступенем невизначеності.

2.2.3. Критерій Гурвіца

Критерій Гурвіца є компромісним підходом до прийняття рішень в умовах невизначеності, який поєднує елементи песимізму та оптимізму. Його мета – збалансувати між найгіршими та найкращими можливими результатами кожної альтернативи, забезпечуючи обґрунтований вибір навіть за відсутності ймовірнісної інформації про стани зовнішнього середовища.

В першу чергу при реалізації критерію Гурвіца складається таблиця, де рядки відповідають альтернативам, а стовпці – можливим станам середовища. Елементи таблиці відображають виграші для кожної альтернативи залежно від стану зовнішнього середовища. Для кожного рядка матриці визначається найменше значення виграшу, визначається найбільше значення виграшу. Вводиться коефіцієнт оптимізму α ($0 \leq \alpha \leq 1$), який відображає ступінь схильності до оптимістичного чи песимістичного підходу. Якщо $\alpha=1$, вибір повністю оптимістичний (максимакс). Якщо $\alpha=0$, вибір повністю песимістичний (максимін).

Після цього для кожної альтернативи обчислюється значення критерію за формулою:

$$H_i = \alpha \cdot \max a_{ij} + (1 - \alpha) \cdot \min a_{ij} \quad (2.3)$$

Максимальне з обчислених значень вважається оптимальним згідно з критерієм Гурвіца.

Урахування як найкращих, так і найгірших результатів забезпечує рівновагу між ризиком і вигодою. Коефіцієнт α дозволяє адаптувати критерій до різних уподобань і рівнів ризикованості особи, що приймає рішення. Обчислення не потребує складної математики та є зрозумілим навіть для початківців у теорії прийняття рішень.

Однак, результат залежить від суб'єктивного вибору коефіцієнта, що може викликати суперечки або невизначеність. Також ігноруються ймовірності станів середовища, що зменшує точність у ситуаціях, де така інформація доступна.

2.2.4. Критерій Лапласа

Критерій Лапласа, або принцип рівноймовірності, є одним із основних методів прийняття рішень в умовах невизначеності. Він ґрунтується на припущенні, що всі стани середовища мають однакову ймовірність, якщо інша інформація про їх розподіл відсутня. Цей підхід дозволяє уникнути суб'єктивних оцінок, концентруючись на обчисленні середніх виграшів для кожної альтернативи.

Алгоритм побудови критерію Лапласа полягає у складанні таблиці, у якій рядки відповідають альтернативам, а стовпці – можливим станам середовища. Елементи таблиці відображають виграші для кожної альтернативи залежно від стану зовнішнього середовища. Для кожного рядка (альтернативи) обчислюється середнє арифметичне всіх виграшів, що можна описати формулою:

$$L_i = \frac{1}{m} \sum_{j=1}^{m} a_{ij} \quad (2.4)$$

де m – кількість можливих станів системи.

Знову ж таки, як і з критерієм Гурвіца, вибирається максимальне з обчислених значень, яке вважається оптимальним за критерієм Лапласа.

Критерій Лапласа базується на ідеї, що в умовах повної невизначеності кожен стан середовища має рівну ймовірність. Такий підхід узгоджується з принципом максимальної ентропії, який передбачає вибір моделі з найменшою упередженістю за браком достовірної інформації. У цьому сенсі критерій Лапласа є максимально об’єктивним методом оцінювання альтернатив.

Даний критерій має ряд переваг. Він уникає суб’єктивних оцінок і використовує припущення про рівноймовірність. Математичні обчислення є прямолінійними та не потребують додаткових даних.

Однак, він не позбавлений недоліків. Критерій не враховує можливих негативних наслідків найгірших сценаріїв. Рівноймовірність станів середовища рідко відповідає реальним умовам, тому застосування обмежується задачами, де рівноймовірність можна обґрунтовано припустити.

Критерій Лапласа широко використовується в різних галузях, зокрема:

– Економіка – оцінювання інвестицій за відсутності інформації про ринки.
– Менеджмент – вибір стратегій розвитку підприємства.
– Соціальні науки – аналіз політичних або соціальних сценаріїв.

Незважаючи на обмеження, цей метод залишається важливим засобом у теорії прийняття рішень.

2.2.5. Критерій Севіджа

Критерій Севіджа, відомий також як критерій мінімального ризику, є одним із підходів до прийняття рішень в умовах невизначеності. Він базується на аналізі ризиків, пов’язаних із кожною альтернативою, та спрямований на мінімізацію максимально можливого програшу (ризику), який може виникнути при обранні певного рішення. Цей підхід зазвичай застосовується, коли особа або організація прагне уникнути найгірших наслідків.

В основі цього критерію лежить ідея оцінки ризиків у вигляді «упущеного виграшу», тобто різниці між найкращим можливим виграшем у кожному стані середовища та виграшем, який забезпечується вибраною альтернативою. У цьому підході вивчається, наскільки кожна альтернатива відстає від оптимального сценарію, і вибирається та, яка мінімізує максимальний ризик.

Під час реалізації цього критерію створюється таблиця, де рядки відповідають альтернативам, а стовпці – можливим станам середовища. Елементи таблиці представляють виграші кожної альтернативи в певному стані. Для кожного стовпця матриці виграшів визначається максимальне значення. Від максимального значення у стовпці віднімається виграш для кожного рядка (альтернативи). З отриманих результатів формується матриця ризиків. Після цього обчислюється максимальний ризик у рядку матриці ризиків. Оптимальною є альтернатива з найменшим максимальним ризиком.

Метод дозволяє оцінити, наскільки вибір може бути невдалим у найгіршому випадку, що важливо для обережних осіб або організацій. Також забезпечує глибоке розуміння упущених виграшів, допомагаючи визначити слабкі місця альтернатив. Критерій Севіджа підходить для задач, де найгірший сценарій є критичним фактором.

Проте в цього критерію теж є недоліки. Побудова матриці ризиків може бути громіздкою для великих задач із численними альтернативами та станами. Також метод концентрується виключно на найгірших сценаріях, не враховуючи середніх показників виграшу.

Критерій Севіджа застосовується в економіці для вибору стратегій інвестування з мінімальними ризиками, менеджменті для розробки планів, які уникають найгірших сценаріїв та в соціальних науках для прийняття рішень за умов соціальної або політичної невизначеності.

2.2.6. Критерій зважених сум

Критерій зважених сум (Weighted Sum Model, WSM) є одним із найпоширеніших методів мультикритеріального прийняття рішень, що використовується для вибору найкращої альтернативи з урахуванням кількох

критеріїв. Цей метод базується на підході, який передбачає оцінку кожної альтернативи за всіма критеріями з подальшим агрегуванням цих оцінок у єдиний показник за допомогою вагових коефіцієнтів.

Критерій WSM ґрунтується на припущенні, що кожен критерій має свою вагу, яка відображає його значущість для ухвалення рішення. Ваги є числовими величинами, які зазвичай нормалізуються, щоб їхня сума дорівнювала одиниці. Оцінка альтернативи визначається як зважена сума значень за всіма критеріями, що можна описати формулою:

$$S_i = \sum_{j=1}^{n} \omega_j \cdot a_{ij} \qquad (2.5)$$

де ω_j – вага j-го критерію.

Процес реалізації WSM починається з формування критеріїв та альтернатив, за якими оцінюватимуться варіанти. Усі альтернативи повинні бути проаналізовані за кожним із визначених критеріїв. Після цього кожному критерію надається вага, яка відображає його значущість у процесі прийняття рішення. Визначення ваг може здійснюватися експертним шляхом або за допомогою аналітичних методів.

Якщо критерії виражені в різних одиницях виміру, їх потрібно нормалізувати, щоб забезпечити коректне порівняння. Після нормалізації для кожної альтернативи обчислюється зважена сума, яка є агрегованою оцінкою варіанта. Цей розрахунок здійснюється шляхом множення значень критеріїв на відповідні ваги та підсумовування отриманих результатів.

На завершальному етапі проводиться порівняння оцінок усіх альтернатив, і обирається найкраща, тобто та, яка має найвищу загальну оцінку.

Переваги критерію:

- Простота реалізації та розуміння.
- Можливість чіткого врахування ваг критеріїв.
- Застосовність до широкого спектра задач.

Недоліки:

– Метод вимагає чіткого визначення ваг, що може бути суб’єктивним.

– Не враховує взаємозв’язків між критеріями.

– Не підходить для ситуацій, де критерії мають складні, нелінійні залежності.

Критерій зважених сум (WSM) широко застосовується в економіці, бізнесі, інженерії, медицині, транспорті, екології та освіті для вирішення багатокритеріальних задач. Він використовується для вибору інвестиційних проектів, оптимізації бюджетів, визначення пріоритетів, оцінки впливу на довкілля, вибору маршрутів чи медичних методів. Завдяки простоті реалізації та можливості враховувати вагомість критеріїв, WSM дозволяє інтегрувати різні показники ефективності, витрат чи результативності, що робить його універсальним інструментом у багатьох галузях.

2.3. Багатокритеріальні методи прийняття рішень

2.3.1. Метод аналізу ієрархій

Метод аналізу ієрархій (Analytic hierarchy process, AHP) є потужним багатокритеріальним підходом до прийняття рішень, запропонованим Томасом Сааті у 1970-х роках. Цей метод широко застосовується для вирішення складних задач, які вимагають систематичного аналізу та порівняння численних альтернатив за декількома критеріями. AHP забезпечує структуровану основу для організації, аналізу та вимірювання багатовимірних задач, дозволяючи приймати раціональні рішення.

Замість того, щоб визначати «правильне» рішення, метод AHP допомагає знайти найкраще рішення, яке відповідає цілям осіб, що приймають рішення та їх розумінню проблеми. Метод пропонує всебічну і раціональну основу для структуризації проблеми, представлення та кількісної оцінки її елементів, їх зв’язку з загальними цілями та оцінки альтернативних рішень.

Перший етап в методі полягає в структуризації проблеми у вигляді ієрархії. Це означає розбиття проблеми на декілька рівнів, починаючи від загальних аспектів і

переходячи до більш конкретних, і організацію цих аспектів у відповідності до методу AHP. В процесі побудови ієрархії учасники отримують глибше розуміння проблеми, її контексту, а також точок зору один одного.

Ієрархія – це система, де елементи розташовані за рівнями, причому кожен елемент, за винятком верхнього, підпорядкований іншим елементам. І хоча концепція ієрархії є інтуїтивно зрозумілою, вона також може бути описана математично. Зазвичай ієрархічні діаграми мають вигляд піраміди, де на вершині знаходиться єдиний елемент, але сама структура не обов'язково повинна бути пірамідальною.

Прикладом ієрархії у комп'ютерній системі може бути блок живлення, який знаходиться на вершині ієрархії, а монітор, клавіатура та миша розташовані нижче.

У процесі набуття знань ієрархії використовуються для розбиття складних тем на менші, більш керовані компоненти. Наприклад, студенти-медики вивчають анатомію ієрархічно, аналізуючи системи, такі як м'язова, кровоносна та нервова системи, на різних рівнях: від загальних категорій до детальних компонентів, таких як м'язи, кістки та клітини. Цей процес покращує загальне розуміння, допомагаючи зосередитись на одному елементі за раз.

Подібним чином, коли ми стикаємося зі складними проблемами прийняття рішень, ієрархічна структура дозволяє інтегрувати велику кількість інформації, надаючи чіткіше та більш всебічне уявлення про ситуацію. За допомогою систематизації інформації таким чином, приймаючі рішення можуть сформувати більш обґрунтоване розуміння проблеми.

В ієрархії AHP кожен елемент є структурованим способом моделювання прийняття рішення. Вона включає загальну мету, набір альтернативних варіантів для досягнення цієї мети та набір критеріїв, які пов'язують ці альтернативи з метою. Кожен критерій може бути розбитий на підкритерії, і так до необхідної глибини.

Процес побудови ієрархії AHP залежить не лише від самої проблеми, а й від знань, суджень та цінностей учасників процесу прийняття рішень. Створення такої ієрархії зазвичай включає детальні обговорення та дослідження. Після початкової побудови ієрархія може змінюватися, щоб включати нові критерії або альтернативи.

Цей процес дає змогу ефективно структурувати складні рішення, зменшити невизначеність і забезпечити більш обґрунтовані та ефективні результати на основі порівняння різних варіантів за допомогою заданих критеріїв.

Після побудови ієрархії учасники процесу прийняття рішень проводять її аналіз через серію парних порівнянь, що дозволяють отримати числові масштаби вимірювання для кожного вузла ієрархії. Критерії порівнюються парно з метою оцінки їх важливості щодо поставленої мети. Альтернативи, у свою чергу, порівнюються з кожним критерієм для визначення їх переваг. Всі ці порівняння обробляються математично, після чого для кожного вузла в ієрархії визначаються пріоритети.

У AHP застосовується шкала 1 до 9, що дозволяє більш детально та чітко оцінювати різницю в пріоритетах між варіантами. Цей підхід забезпечує детальну та об’єктивну оцінку елементів, дозволяючи приймати більш обґрунтовані рішення на основі конкретних порівнянь.

Пріоритети – це числові значення, які асоціюються з вузлами ієрархії AHP. Вони представляють відносні ваги елементів у будь-якій групі. Вони є абсолютними числами, які знаходяться між нулем і одиницею, не мають одиниць вимірювання або вимірів.

Пріоритети розподіляються по ієрархії в залежності від її структури, і їх значення залежать від інформації, яку вводять користувачі процесу. Пріоритети для мети, критеріїв та альтернатив тісно взаємопов’язані, але кожен з них має бути оцінений окремо.

Згідно з визначенням, пріоритет мети завжди дорівнює 1.000. Пріоритети альтернатив завжди сумуються до 1.000. У разі наявності кількох рівнів критеріїв ситуація може ускладнюватися, але якщо є тільки один рівень, їхні пріоритети також складають 1.000.

Процес визначення пріоритетів в AHP починається з пріоритетів за замовчуванням, але згодом вони змінюються на основі порівнянь парних альтернатив, введених учасниками прийняття рішень.

Хоча AHP може бути використаний окремими особами для вирішення простих завдань, цей метод найбільш корисний у випадках, коли групи людей працюють над складними проблемами, пов’язаними з людськими сприйняттями та судженнями, і рішення яких матимуть довгострокові наслідки.

Метод AHP має численні переваги, що робить його корисним інструментом для прийняття рішень. Однією з основних переваг є його інтуїтивність, що спрощує процес прийняття рішень. Це дозволяє спеціалістам зосередитися на важливих аспектах, не витрачаючи надмірно багато часу на складні математичні обчислення. Крім того, AHP дозволяє обробляти велику кількість критеріїв, що робить цей метод гнучким та ефективним у вирішенні складних завдань.

Ще однією важливою перевагою AHP є забезпечення постійності результатів. Використання програмного забезпечення для цього методу дозволяє перевіряти та коригувати вхідні дані, що дає змогу отримувати надійні та обґрунтовані результати. Важливою характеристикою методу є також його здатність зменшувати упередженість при прийнятті рішень. Всі учасники мають можливість внести свої оцінки, що сприяє колективному обговоренню та сприяє ухваленню оптимального рішення. Завдяки цьому, AHP забезпечує ефективне використання навичок і знань фахівців, що працюють над вирішенням проблеми, і дозволяє отримати найбільш обґрунтоване та раціональне рішення для конкретної ситуації.

2.3.2. Методи нечіткого багатокритеріального прийняття рішень

Нечітке багатокритеріальне прийняття рішень (Fuzzy MCDM) є еволюцією класичних методів MCDM (Multiple-criteria decision-making) і використовує принципи нечіткої логіки для обробки невизначеності в процесі прийняття рішень. У Fuzzy MCDM ухвалення рішень відбувається за допомогою нечітких множин, що дозволяє приймати рішення в умовах, де оцінки критеріїв або альтернатив не є точними або чітко визначеними.

Нечітка логіка (Fuzzy Logic), що була запропонована Лотфі Заде в 1965 році, дозволяє працювати з невизначеними або нечіткими даними. На відміну від класичної логіки, де кожен елемент належить до чіткої категорії (істина або хиба), нечітка логіка

дозволяє елементам набувати значень часткової істинності, коли значення істинності може коливатися від повністю істинного до повністю хибного. Це дозволяє більш гнучко та точно моделювати реальні ситуації.

Основний процес Fuzzy MCDM полягає в оцінці альтернатив на основі декількох критеріїв, де кожен критерій описується за допомогою нечіткої функції належності. Під час цього процесу застосовуються різні методи.

Першим кроком є визначення нечітких множин для критеріїв, які представляють різні рівні важливості або переваги. Це дозволяє моделювати оцінки, коли точні значення важко визначити або коли вони залежать від суб'єктивних факторів.

Далі нечіткі оцінки присвоюються альтернативам на основі їхніх характеристик і переваг за кожним критерієм. Ці оцінки можуть бути отримані за допомогою експертних оцінок або аналізу даних.

Після присвоєння нечітких оцінок наступним кроком є агрегація. Це можна зробити за допомогою різних методів, таких як обчислення середнього значення, оптимізація або застосування спеціальних операцій для об'єднання нечітких чисел.

Після агрегації оцінок альтернативи порівнюються, і найкраща альтернатива визначається на основі відповідних критеріїв. Рішення може ґрунтуватися на різних принципах, включаючи евристику або інші методи аналізу нечітких даних.

Такий підхід дозволяє особам, які приймають рішення, враховувати невизначеність і неточність у процесі оцінювання, забезпечуючи більшу гнучкість у складних сценаріях прийняття рішень.

Метод Fuzzy MCDM включає кілька важливих технік, які часто використовуються для вирішення специфічних проблем прийняття рішень. Однією з найпомітніших технік є Fuzzy Analytic Hierarchy Process (FAHP), яка поєднує принципи методу аналізу ієрархій (AHP) з нечіткою логікою. У FAHP ієрархічна структура проблеми прийняття рішень включає нечіткі оцінки для критеріїв і альтернатив, що дозволяє гнучко враховувати невизначеність при порівнянні варіантів. FAHP є особливо ефективним, коли потрібно враховувати кілька критеріїв,

як це відбувається, наприклад, при виборі постачальника або оцінці інвестиційних проектів.

Іншим важливим методом є техніка упорядкування переваг за подібністю до ідеального рішення (Technique for Order Preference by Similarity to Ideal Solution), який оцінює альтернативи на основі їх відстані від ідеального рішення. В контексті нечітких даних, Fuzzy TOPSIS враховує неточність у визначенні найбільш прийнятного варіанту. Ключовим етапом цього процесу є обчислення відстані до ідеального та найгіршого можливого рішення для кожної альтернативи.

Елімінація і вибір, що перетворює реальність (Elimination and Choice Translating Reality) – це ще одна техніка, яка призначена для вибору найкращих альтернатив шляхом порівняння та усунення непридатних варіантів. Завдяки інтеграції нечітких чисел цей метод може справлятися з комплексними, суперечливими критеріями, які часто зустрічаються в реальних сценаріях. Він корисний, коли рішення потрібно приймати не тільки на основі ідеальних варіантів, а й з урахуванням найбільш прийнятних рішень в межах певних обмежень.

Метод Fuzzy MCDM має кілька переваг, серед яких варто виділити його здатність ефективно враховувати невизначеність у даних та оцінках критеріїв. Завдяки використанню нечіткої логіки, цей метод дозволяє моделювати ситуації, коли точні значення важко визначити. Це дає можливість приймати більш реалістичні рішення, що враховують різноманітні аспекти, де інформація є нечіткою або неповною. Крім того, методи Fuzzy MCDM дуже гнучкі у застосуванні і можуть бути використані для вирішення широкого спектра проблем у різних галузях, таких як бізнес, інженерія, екологія, медицина тощо.

Однак, існують і певні обмеження. Одним з них є складність реалізації методів Fuzzy MCDM, оскільки вони можуть вимагати значних обчислювальних ресурсів та складних математичних моделей для обробки нечітких даних і їх подальшої агрегації. Крім того, деякі техніки вимагають використання експертних оцінок, що може призвести до суб’єктивності і, як наслідок, до упередженості або невизначеності в результатах.

Методи Fuzzy MCDM широко використовуються в різних сферах, де потрібно приймати рішення в умовах невизначеності та складності. У бізнесі та менеджменті ці методи застосовуються для вибору стратегій, інвестиційних рішень або постачальників, коли необхідно врахувати різноманітні критерії, включаючи якісні та кількісні аспекти. В екологічному плануванні вони дозволяють оцінювати вплив на навколишнє середовище та вибирати сталий розвиток, зважаючи на непевність у довгострокових наслідках.

У медицині методи Fuzzy MCDM можуть бути використані для діагностики або вибору методів лікування, коли ситуація не завжди чітко визначена, а також для обробки суб'єктивних оцінок лікарів. В інженерії ці методи допомагають оптимізувати проектні рішення, вибір матеріалів або оцінку ризиків, зважаючи на множинність варіантів і необхідність у врахуванні багатьох параметрів.

Загалом, застосування Fuzzy MCDM допомагає приймати більш точні та обґрунтовані рішення в умовах, де традиційні методи не завжди дають бажаний результат через складність або неточність даних.

Fuzzy MCDM є потужним інструментом для прийняття рішень у складних і невизначених ситуаціях, що робить його корисним для вирішення широкого спектру багатокритеріальних задач у різних галузях.

2.3.3. Метод зважених сум

Метод зважених сум (Weighted Sum Model, WSM) виступає практичною реалізацією теоретичного підходу, описаного в рамках критерію зважених сум. Цей критерій був описаний в пункті 2.2.6.

На відміну від критерію, який зосереджений на концептуальному визначенні важливості критеріїв та їх ваг, метод представляє алгоритм обчислення, що дозволяє інтегрувати ці ваги у процес аналізу й оцінки альтернатив.

Цей метод є ефективним інструментом, який поєднує математичну простоту з високою практичністю. Використовуючи ваги критеріїв, метод обчислює загальну оцінку кожного варіанту через процедуру агрегації, яка дозволяє враховувати як кількісні, так і якісні аспекти оцінювання. Зокрема, WSM має на меті надати чітке

кількісне уявлення про ступінь відповідності альтернатив певній сукупності критеріїв, що робить його незамінним для задач, де є необхідність у порівнянні великої кількості варіантів.

У методі зважених сум також приділяється увага нормалізації даних. Якщо оцінки критеріїв представлені в різних одиницях вимірювання, вони підлягають перетворенню до єдиної шкали, що дозволяє уникнути спотворення підсумкових результатів. Цей процес є ключовим для забезпечення коректності результатів і рівнозначності критеріїв при їх подальшому порівнянні.

Крім того, метод активно використовується у багатокритеріальному аналізі для підтримки прийняття рішень у таких сферах, як інвестиції, проектування систем, управління ресурсами, вибір постачальників і навіть у соціально-економічному плануванні. Завдяки своїй прозорості та зручності у використанні, WSM став основою для багатьох сучасних інструментів прийняття рішень, зокрема тих, що інтегруються у програмні засоби автоматизації аналізу даних.

Таким чином, метод зважених сум, хоча і ґрунтується на принципах критерію зважених сум, є самостійною реалізацією, яка розширює ці принципи, дозволяючи не лише враховувати важливість критеріїв, але й забезпечувати аналітичну інтеграцію результатів у зручній для практичного застосування формі.

2.4. Висновок до розділу

У цьому розділі детально розглянуто основні принципи та методи підтримки прийняття рішень, включаючи як традиційні підходи, так і сучасні алгоритми, що дозволяють вирішувати складні багатокритеріальні задачі. Зокрема, описано такі методи, як критерії Вальда, Гурвіца, Лапласа та Севіджа, які є основою класичних підходів до прийняття рішень за умов невизначеності. Особливу увагу приділено універсальним методам, таким як метод аналізу ієрархій (AHP), метод зважених сум (WSM) і методи нечіткого багатокритеріального прийняття рішень (Fuzzy MCDM), які широко використовуються у сучасних системах.

Розкрито теоретичні основи кожного з підходів, їх математичні моделі, а також проаналізовано практичні аспекти їх застосування в різних сценаріях. Показано, як кожен із методів дозволяє обробляти множинні критерії, враховуючи їх вагомість, а також справлятися з невизначеностями та суперечностями між альтернативами. Окрім того, наведено порівняння між різними підходами, підкреслено їх сильні сторони, недоліки та обмеження, що дозволяє глибше зрозуміти, у яких умовах їх доцільно застосовувати.

Увагу також приділено актуальним питанням щодо ефективності цих методів, їх адаптивності до складних багатокритеріальних задач та необхідності створення більш гнучких і адаптивних підходів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ДОКУМЕНТУВАННЯ ВЛАСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

3.1. Опис проєкту

Програмне забезпечення, що розробляється, призначене для підтримки процесів багатокритеріального прийняття рішень. Програма інтегрує три методи: Аналітичний ієрархічний процес (AHP), метод нечіткого багатокритеріального прийняття рішень (Fuzzy MCDM) та метод зважених сум (WSM). Реалізація цих методів дає змогу проводити аналіз альтернатив із врахуванням різних критеріїв і їхньої ваги, що є надзвичайно важливим для підтримки управлінських, технічних і стратегічних рішень.

Метою програми є створення зручного та ефективного інструменту для аналізу складних багатокритеріальних задач у різних сферах діяльності. Вона спрямована на підтримку прийняття рішень, де необхідно врахувати декілька взаємопов’язаних критеріїв, а також забезпечити користувачів доступним інструментом для оцінки, порівняння та вибору оптимальних альтернатив. Особливу увагу приділено можливості адаптації програми до потреб користувача, що включає варіативність у виборі підходів до вирішення задач.

Програма базується на сучасних підходах до програмування та обробки даних, що забезпечує використання оптимізованих алгоритмів і бібліотек, дозволяє виконувати обчислення навіть для складних систем з великим числом критеріїв та альтернатив. Реалізована архітектура програми дозволяє додавати нові методи прийняття рішень, інтегрувати нові модулі для обробки даних або модифікувати існуючі.

У ході розробки використовувалися такі технології та інструменти:

– Python як основна мова програмування для реалізації алгоритмів методів.

– Flask для створення веб-інтерфейсу, що забезпечує інтерактивність і доступність програми через браузер.

– Бібліотеки для роботи з числовими обчисленнями, такими як NumPy, що використовувалися для реалізації математичних розрахунків.

– Інструменти для генерації звітів у форматі PDF, зокрема pdfMake і html2canvas, які дозволяють автоматизувати створення структурованих документів із результатами аналізу.

– HTML, CSS, і JavaScript для побудови клієнтської частини, що забезпечує динамічний і функціональний користувацький інтерфейс.

– Bootstrap для розробки адаптивного дизайну, який сприяє зручному відображенню програми на різних пристроях.

Програма також відповідає вимогам документування та прозорості результатів: усі результати аналізу можуть бути представлені у вигляді звітів у форматі PDF, що генеруються автоматично. Це забезпечує можливість подальшого використання результатів для ухвалення рішень, їх обговорення у командах або збереження для архівування.

Таким чином, програма спрямована на задоволення потреб як індивідуальних, так і корпоративних користувачів, пропонуючи комплексний, адаптивний та ефективний підхід до розв’язання багатокритеріальних задач.

3.2. Проєктування програмного забезпечення

У процесі проектування програмного забезпечення було розроблено чітку структуру додатка, що забезпечує зручну навігацію та ефективну організацію функціоналу. Програма складається з головної сторінки, сторінок для вводу даних для кожного методу, а також сторінок для відображення результатів аналізу.

Головна сторінка слугує точкою входу та містить вибір одного з трьох методів прийняття рішень: AHP, Fuzzy MCDM і WSM. Вона створена із використанням адаптивного дизайну на базі Bootstrap, що дозволяє однаково зручно працювати на пристроях різних типів.

Для кожного методу передбачено окрему сторінку для введення даних. На цих сторінках користувачі мають змогу вказати критерії, альтернативи та вагові

коефіцієнти. Наприклад, для AHP користувач вводить парні порівняння, для Fuzzy MCDM – нечіткі оцінки у вигляді трійки значень, а для WSM – числові значення критеріїв та їхні ваги.

Усі введені дані проходять перевірку як на рівні клієнтської частини, так і на серверному рівні. Перевірка введених даних на клієнтській та серверній сторонах є важливим компонентом для забезпечення надійності та безпеки веб-додатка. У програмі клієнтська валідація реалізована за допомогою JavaScript, що дозволяє швидко виявляти базові помилки, такі як відсутність обов'язкових полів або некоректні формати, без зайвих запитів до сервера.

Серверна валідація на Flask гарантує обробку навіть потенційно некоректних даних, що могли обійти клієнтську перевірку. Це включає перевірку коректності, відповідності вимогам.

Об'єднання клієнтської та серверної перевірки створює двошарову систему захисту, яка підвищує стабільність роботи додатка. Клієнтська перевірка підвищує зручність роботи користувача, а серверна гарантує, що навіть у разі помилок або зловмисних дій збереження коректності даних не порушується. Цей підхід також знижує навантаження на сервер, оскільки багато помилок перехоплюються ще на етапі клієнта, дозволяючи обробляти лише валідні запити.

Поєднання обох підходів створює ефективну систему, де клієнтська валідація підвищує зручність, а серверна забезпечує надійність.

Після виконання обчислень програма перенаправляє користувача на відповідну сторінку виводу результатів. Кожна сторінка результатів надає структуровану інформацію у вигляді таблиць та графіків, які допомагають краще зрозуміти підсумки аналізу. Окрім цього, користувач може сформувати звіт у форматі PDF завдяки інтеграції інструментів pdfMake і html2canvas. Це дозволяє автоматично генерувати документ, який містить усі необхідні деталі, включно з графічними візуалізаціями.

Навігація між сторінками здійснюється через маршрути, реалізовані у Flask. Користувач має можливість повернутися до головної сторінки для повторного вибору методу або розпочати новий аналіз з іншими параметрами. Такий підхід забезпечує

послідовність дій, інтуїтивність у використанні програми та можливість легкого розширення її функціоналу в майбутньому.

Маршрутизація у Flask є одним із ключових компонентів веб-додатків, що дозволяє обробляти запити до різних URL-адрес і забезпечує взаємодію користувача з додатком. Flask використовує декоратори для визначення маршрутів, що значно спрощує налаштування зв'язку між URL і функціями, які виконують відповідну логіку.

Інтеграція сучасних веб-технологій, таких як HTML, CSS, JavaScript та Bootstrap, дозволяє створити адаптивний і зручний інтерфейс, орієнтований на користувача будь-якого рівня технічної підготовки.

3.3. Розробка проєкту

3.3.1. Встановлення засобів розробки

Visual Studio Code (VS Code) був обраний для розробки через свою багатофункціональність, доступність і підтримку великої кількості розширень, що робить його зручним інструментом для розробників різного рівня.

Для встановлення Visual Studio Code необхідно спочатку завантажити інсталятор з офіційного вебсайту. Для цього потрібно перейти на сторінку https://code.visualstudio.com, де можна вибрати версію програми, яка відповідає операційній системі – рисунок 3.1. Завантаження VS Code. Після завантаження інсталяційного файлу потрібно запустити його, що ініціює процес інсталяції.

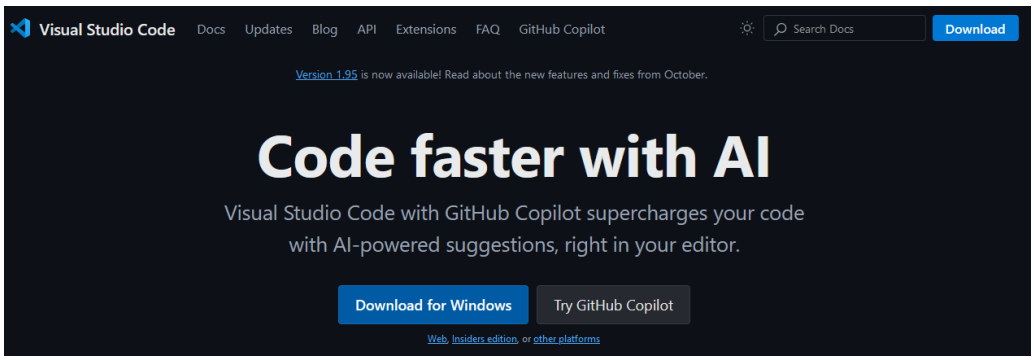


Рис.3.1. Завантаження VS Code

Під час встановлення користувач має можливість налаштувати додаткові параметри, зокрема, вибрати опцію додавання Visual Studio Code до контекстного меню провідника файлів або налаштування для запуску через командний рядок. Ці опції сприяють зручнішій роботі з редактором у майбутньому.

Після завершення інсталяції, користувач може запустити програму та налаштувати середовище для роботи, зокрема встановити розширення для підтримки різних мов програмування. Це можна зробити через панель розширень, відкривши її через відповідну іконку на боковій панелі програми. Додавши необхідні плагіни, Visual Studio Code буде готовий до використання для розробки програмних проєктів.

Python був обраний для розробки через свою простоту в освоєнні, багатофункціональність та активну підтримку в широкому спектрі галузей, включаючи наукові обчислення, веб-розробку, автоматизацію, обробку даних та штучний інтелект. Однією з основних переваг Python є його зручний і зрозумілий синтаксис, що дозволяє швидко розпочати роботу, навіть для початківців у програмуванні. Цей аспект робить Python ідеальним вибором для розробки прототипів та автоматизованих рішень, де важлива швидкість розробки.

Для встановлення Python потрібно завантажити інсталятор з офіційного сайту python.org і вибрати версію. Після запуску інсталятора обов’язково потрібно вибрати опцію "Add Python to PATH" для зручності використання Python через командний рядок.

Після завершення інсталяції перевяємо коректність встановлення, виконавши команду python --version або python3 --version у командному рядку – рисунок 3.2. Перевірка версії Python.

Рис.3.2. Перевірка версії Python

Наступним кроком є встановлення системи контролю версій Git. Для цього необхідно запустити наступну команду в PowerShell:

```
winget install --id Git.Git -e --source winget
```

Flask був обраний для розробки веб-додатку з кількох причин. По-перше, це легкий і мінімалістичний веб-фреймворк для Python, що дозволяє швидко створювати веб-застосунки без зайвих налаштувань та складних конфігурацій. Flask є чудовим вибором для малих та середніх проєктів, де важливі гнучкість і простота, а також можливість інтеграції з іншими бібліотеками та компонентами без значних обмежень. Крім того, він забезпечує високу ступінь контролю над процесом розробки, дозволяючи розробникам вирішувати, які компоненти та функції вони хочуть використовувати в проєкті.

Щоб встановити Flask потрібно відкрити командний рядок та виконати команду, зображену на рисунку 3.3. Встановлення Flask.

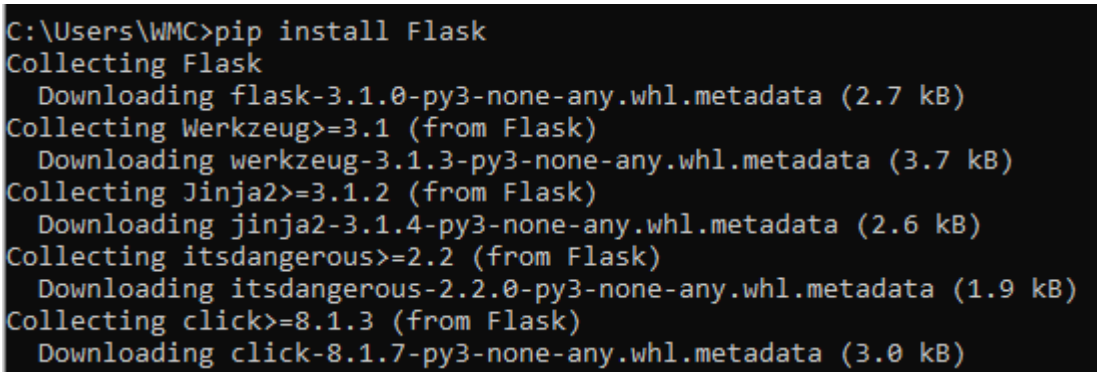


Рис.3.3. Встановлення Flask.

Після цього Flask буде встановлений і можна починати розробку веб-додатку.

3.3.2. Створення проєкту та головної сторінки

Для створення проєкту Flask достатньо створити папку проєкту та всередині створити папки templates і static та файл app.py.

В файлі app.py необхідно створити змінну app, в якій буде зберігатися об’єкт типу Flask. Це є основним об’єктом для нашого додатку.

Далі необхідно створити декоратор, який визначає маршрут для головної сторінки додатку. Після цих кроків код файлу app.py буде виглядати як показано в лістингу 3.1.

Лістинг 3.1.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Далі потрібно створити файл index.html, яка буде формуватися за допомогою функції render_template в файлі app.py. В цьому файлі буде реалізована основна розмітка головної сторінки. Його потрібно створити в папці templates.

Після формування стандартної структури каркасу HTML, яка включає створення основних елементів сторінки (head, body та інші), необхідно підключити Bootstrap до нашої сторінки, це можна зробити, додавши код, поданий в лістингу 3.2., всередину тегу head:

Лістинг 3.1.

```
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.
min.css" rel="stylesheet">
```

Це дозволить нам застосовувати стилі Bootstrap для елементів сторінки, що полегшить та пришвидшить розробку адаптивного інтерфейсу користувача.

Основний контейнер body містить заголовок «Алгоритм для обчислень» та три картки (по одній для кожного з методів: AHP, Fuzzy MCDM та WSM).

Кожна картка містить заголовок з назвою методу, короткий опис методу та кнопка, що веде на відповідну сторінку для введення даних.

Картки побудовані за допомогою компонентів Bootstrap, які дозволяють легко створювати відформатовані елементи з тінями та кнопками.

Клас row-cols-1 row-cols-md-3 визначає кількість колонок для карток в залежності від ширини екрану (одна колонка на малих екранах та три на більших).

Після реалізації вище описаного код файлу index.html наведено в лістингу 3.2.

Лістинг 3.2.

```
<body class="bg-light">
    <div class="container py-5">
        <h1 class="text-center mb-5">Алгоритм для обчислень</h1>
        <div class="row row-cols-1 row-cols-md-3 g-4">
            <!-- AHP Card -->
            <div class="col">
                <div class="card h-100 text-center border-0
shadow-lg">
                    <div class="card-body">
                        <h5 class="card-title">Метод AHP</h5>
                        <p class="card-text">Аналітичний
ієрархічний процес для багатокритерійного вибору.</p>
                        <a href="/ahp_input" class="btn btn-
primary btn-lg">Перейти</a>
                    </div>
                </div>
            </div>
            <!-- Fuzzy MCDM Card -->
            <div class="col">
                <div class="card h-100 text-center border-0
shadow-lg">
                    <div class="card-body">
                        <h5 class="card-title">Метод Fuzzy
MCDM</h5>
                        <p class="card-text">Використання
нечіткої логіки для ухвалення рішень.</p>
                        <a href="/fuzzy_input" class="btn btn-
success btn-lg">Перейти</a>
                    </div>
                </div>
            </div>
            <!-- WSM Card -->
            <div class="col">
                <div class="card h-100 text-center border-0
shadow-lg">
                    <div class="card-body">
                        <h5 class="card-title">Метод WSM</h5>
                        <p class="card-text">Простий метод
зважених сум для вибору альтернатив.</p>
                        <a href="/wsm_input" class="btn btn-
warning btn-lg">Перейти</a>
                    </div>
                </div>
            </div>
        </div>
        <!-- Bootstrap JS -->
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bu
ndle.min.js"></script>
    </body>
```

Тепер ми можемо запустити локальний сервер за допомогою Python, щоб мати можливість переглянути створену сторінку – рисунок 3.4. Запуск локального сервера додатку.

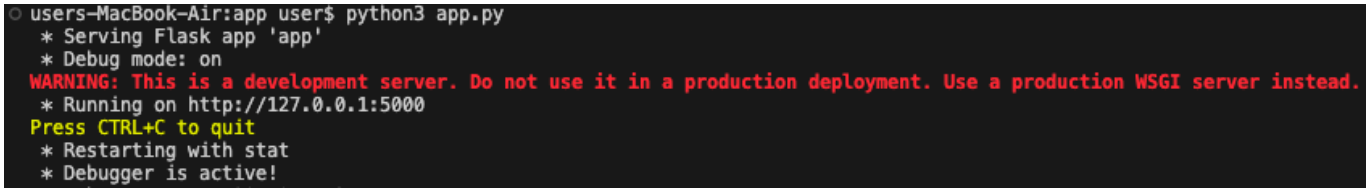


Рис.3.4. Запуск локального сервера додатку

Тепер в браузері переходимо за посиланням, вказаним у командному рядку на рис. 3.4. Відкриється головна сторінка нашого додатку, вона зображена на рисунку 3.5. Головна сторінка додатку.

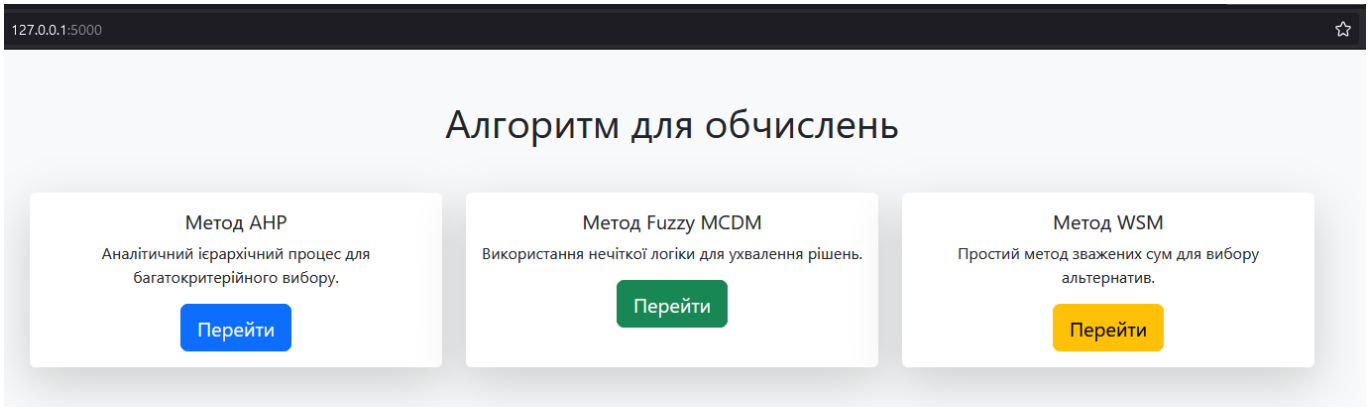


Рис.3.5. Головна сторінка додатку

На цьому створення проєкту та головної сторінки можна вважати завершеним.

3.3.3. Реалізація алгоритму методу аналізу ієрархій

Наступним кроком є створення алгоритму AHP для обчислення ваг критеріїв, оцінок альтернатив і визначення найкращої альтернативи з множини варіантів.

Потрібно створити папку utils, в якій будуть зберігатися всі необхідні алгоритми мовою Pyhton в нашому додатку. В цій папці створюємо файл ahp_algorithm.py.

В файлі ініціалізується функція ahp, яка приймає чотири основні параметри:

– criteria: список критеріїв, які використовуються для оцінки альтернатив.

– options: список альтернатив (варіантів).

– comparisons: словник, що містить попарні порівняння критеріїв.

– alternatives: словник, що містить оцінки альтернатив за кожним критерієм, у форматі попарних порівнянь.

Вхідні дані, які буде вводити користувач, перетворюватимуться в потрібний формат перед передачею їх в нашу функцію.

Далі ініціалізується матриця порівнянь для критеріїв, яка спочатку заповнюється одиницями, оскільки кожен критерій порівнюється з собою, і всі критерії вважаються рівнозначними. Потім матриця заповнюється на основі значень з словника comparisons, де зазначено порівняння між двома критеріями. Якщо критерій і є більш важливим за критерій j, то матриця отримує відповідне значення, а для позиції j, і записується обернене значення.

Наступним кроком розраховуються ваги критеріїв. Для цього використовуються власні значення та власні вектори матриці порівнянь. Функція numpy.linalg.eig знаходить ці значення та вектори, після чого вибирається вектор, що відповідає найбільшому власному значенню. Цей вектор нормалізується так, щоб сума всіх його елементів дорівнювала 1, що дає фінальні ваги для кожного критерію.

Після цього створюється матриця оцінок альтернатив, подібно до матриці порівнянь критеріїв. Ця матриця заповнюється значеннями, які показують ефективність кожної альтернативи щодо кожного критерію, і ці значення отримуються зі словника alternatives, де зазначені парні порівняння альтернатив для кожного критерію.

Коли матриця оцінок альтернатив заповнена, підсумкові оцінки кожної альтернативи обчислюються за допомогою множення матриці оцінок на вектор ваг критеріїв. Це дає зважену суму для кожної альтернативи, що дозволяє оцінити її загальну ефективність.

Нарешті, найкраща альтернатива визначається за допомогою вибору тієї, яка

має найвищу оцінку. Функція повертає словник, що містить критерії, їх ваги, альтернативи, їх оцінки та найкращу альтернативу.

Функція ahp наведена в лістингу 3.3.

Лістинг 3.3.

```
def ahp(criteria, options, comparisons, alternatives):
    n_criteria = len(criteria)
    if n_criteria == 0 or len(options) == 0:
        raise ValueError("Критерії та альтернативи не можуть бути порожніми.")
    comparison_matrix = np.ones((n_criteria, n_criteria))
    for key, value in comparisons.items():
        i, j = map(int, key.split('_')[1:])
        comparison_matrix[i, j] = value
        comparison_matrix[j, i] = 1 / value
    eigvals, eigvecs = np.linalg.eig(comparison_matrix)
    max_index = np.argmax(eigvals.real)
    criteria_weights = eigvecs[:, max_index].real
    criteria_weights /= np.sum(criteria_weights)
    alternative_matrix = np.ones((len(options), n_criteria))
    for key, value in alternatives.items():
        i, j = map(int, key.split('_')[1:])
        alternative_matrix[i, j] = value

    final_scores = alternative_matrix.dot(criteria_weights)

    best_alternative_index = np.argmax(final_scores)
    best_alternative = options[best_alternative_index]
    return {
        "criteria": criteria,
        "criteria_weights": criteria_weights.tolist(),
        "alternatives": options,
        "alternative_scores": final_scores.tolist(),
        "best_alternative": best_alternative,
    }
```

Створена функція в подальшому буде імпортуватися в необхідні файли та викликатися для опрацювання введених користувачем даних.

3.3.4. Реалізація алгоритму методу нечіткого багатокритеріального прийняття рішень

В папці utils створюємо файл fuzzy_mcdm_algorithm.py, в якому буде реалізований алгоритм мовою Python. Метод Fuzzy MCDM застосовується для оцінки

альтернатив на основі нечіткої логіки. Він дозволяє враховувати невизначеність та нечіткість при оцінці критеріїв і виборі альтернатив.

В файлі створюється функція fuzzy_mcdm. Вхідними параметрами є:

– criteria – список критеріїв, які використовуються для оцінки альтернатив.

– options – список альтернатив, які оцінюються.

– weights – список ваг для кожного критерію. Кожна вага представлена як трійка чисел – нижня, середня та верхня межі нечіткої ваги для критерію.

– evaluations – список оцінок альтернатив за кожним критерієм. Для кожної альтернативи є трійка оцінок: нижня, середня та верхня оцінка.

Спочатку функція перевіряє вхідні дані, переконуючись, що кількість критеріїв і альтернатив не є нульовою, що кількість критеріїв збігається з кількістю ваг, а також що кожна альтернатива має оцінки для кожного критерію.

Далі здійснюється нормалізація ваг критеріїв, де сума середніх значень усіх ваг дорівнює одиниці. Це дозволяє привести ваги до єдиної шкали, що дає можливість порівнювати різні критерії. Оцінки для кожної альтернативи обчислюються через добуток кожної оцінки на відповідну нормалізовану вагу, після чого ці добутки підсумовуються для кожної альтернативи, даючи трійки оцінок (нижня, середня, верхня).

Після цього відбувається дефазифікація, тобто обчислення скалярного значення для кожної альтернативи, використовуючи центр ваги для кожної трійки оцінок. Це значення дозволяє звести нечітку оцінку до чіткої.

Останнім кроком є вибір найкращої альтернативи, яка отримала найбільше скалярне значення. Функція повертає результат у вигляді словника, який містить критерії, ваги, альтернативи, нечіткі оцінки, скалярні оцінки та найкращу альтернативу. Це дозволяє автоматизувати процес прийняття рішень у умовах нечіткої інформації, надаючи можливість враховувати варіативність оцінок і ваг.

Реалізація вище описаного алгоритму представлена в лістингу 3.4.

Лістинг 3.4.

```
def fuzzy_mcdm(criteria, options, weights, evaluations):
```

```
        if len(criteria) == 0 or len(options) == 0:
            raise ValueError("Критерії та альтернативи не можуть бути
порожніми.")
        if len(criteria) != len(weights):
            raise ValueError("Кількість критеріїв повинна збігатися з
кількістю ваг.")

        if len(evaluations) != len(options) or any(len(ev) !=
len(criteria) for ev in evaluations):
            raise ValueError("Кількість альтернатив або оцінок не
відповідає кількості критеріїв.")

        weight_sums = np.sum([w[1] for w in weights])  # Сума середніх
значень
        normalized_weights = [
            (w[0] / weight_sums, w[1] / weight_sums, w[2] /
weight_sums)
            for w in weights
        ]

        final_scores = []
        for option_evaluations in evaluations:
            score = [0, 0, 0]  # lower, mean, upper
            for i, (crit_evaluation, crit_weight) in
enumerate(zip(option_evaluations, normalized_weights)):
                score = [
                    score[0] + crit_evaluation[0] * crit_weight[0],
                    score[1] + crit_evaluation[1] * crit_weight[1],
                    score[2] + crit_evaluation[2] * crit_weight[2],
                ]
            final_scores.append(tuple(score))

        def defuzzify(fuzzy_score):
            l, m, u = fuzzy_score
            return (l + 4 * m + u) / 6

        scalar_scores = [defuzzify(score) for score in final_scores]
        best_alternative_index = np.argmax(scalar_scores)
        best_alternative = options[best_alternative_index]

        return {
            "criteria": criteria,
            "weights": weights,
            "alternatives": options,
            "fuzzy_scores": final_scores,
            "scalar_scores": scalar_scores,
            "best_alternative": best_alternative,
        }
```

Створена функція в подальшому буде викликатися для опрацювання введених користувачем даних на відповідній формі вводу.

3.3.5. Реалізація алгоритму методу зважених сум

Створюємо ще один файл в папці utils з назвою wsm_algorithm.py. В ньому буде реалізований алгоритм методу зважених сум.

В файлі створюємо функцію wsm за наступними вхідними параметрами:

– criteria – список критеріїв для оцінки.

– options – можливі варіанти вибору.

– weights – числові значення важливості критеріїв.

– evaluations – матриця оцінок альтернатив за кожним критерієм.

Алгоритм починається з перевірки вхідних даних, щоб переконатися, що критерії, альтернативи та оцінки не порожні, а також що кількість критеріїв співпадає з кількістю ваг. Якщо ці умови не виконуються, викликається помилка.

Далі йде етап нормалізації ваг. Для цього сума всіх ваг критеріїв обчислюється, і кожну вагу ділять на цю суму. Це гарантує, що сума всіх ваг дорівнює 1, і кожен критерій має пропорційний вплив на загальний результат.

Наступний етап – це розрахунок підсумкових оцінок для кожної альтернативи. Для кожної альтернативи підсумкова оцінка обчислюється як сума добутків оцінок цієї альтернативи за кожним критерієм і відповідної нормалізованої ваги цього критерію. Це дозволяє отримати підсумкову оцінку, яка враховує важливість кожного з критеріїв.

Після цього алгоритм визначає найкращу альтернативу як ту, що має найвищу підсумкову оцінку. Це здійснюється за допомогою функції argmax, яка вибирає індекс максимальної оцінки.

Результатом роботи алгоритму є словник, що містить список критеріїв, ваг, альтернатив, підсумкових оцінок для кожної альтернативи та найкращу альтернативу.

Реалізація вище описаного алгоритму представлена в лістингу 3.5.

Лістинг 3.5.

```
def wsm(criteria, weights, options, evaluations):
    if len(criteria) == 0 or len(options) == 0:
        raise ValueError("Критерії та альтернативи не можуть бути порожніми.")
```

```
        if len(criteria) != len(weights):
            raise ValueError("Кількість критеріїв повинна збігатися з
кількістю ваг.")

        if len(evaluations) != len(options) or any(len(ev) !=
len(criteria) for ev in evaluations):
            raise ValueError("Кількість альтернатив або оцінок не
відповідає кількості критеріїв.")

        weight_sum = sum(weights)
        normalized_weights = [w / weight_sum for w in weights]

        final_scores = []
        for option_evaluations in evaluations:
            score = 0
            for i, (crit_evaluation, crit_weight) in
enumerate(zip(option_evaluations, normalized_weights)):
                score += crit_evaluation * crit_weight
            final_scores.append(score)

        best_alternative_index = np.argmax(final_scores)
        best_alternative = options[best_alternative_index]

        return {
            "criteria": criteria,
            "weights": weights,
            "alternatives": options,
            "final_scores": final_scores,
            "best_alternative": best_alternative,
        }
```

Цей метод є простим і ефективним для багатокритеріального вибору, де важливо врахувати різну значимість критеріїв при порівнянні альтернатив.

3.3.6. Реалізація сторінки вводу даних для методу AHP

Наступним кроком є створення сторінки вводу даних для методу аналізу ієрархій. Для цього необхідно в папці templates створити файл ahp_input_form.html. В цьому файлі буде реалізована форма вводу даних для методу. Введені дані після цього будуть опрацьовуватися в інших частинах програми та передаватися у вже реалізовані алгоритми.

HTML-код форми для методу AHP надає зручний веб-інтерфейс для введення даних, необхідних для виконання аналізу ієрархічного процесу ухвалення рішень. Він дозволяє користувачеві послідовно вводити критерії, альтернативи, попарні

порівняння критеріїв, а також оцінки альтернатив. Інтерфейс оформлений за допомогою Bootstrap, що забезпечує сучасний адаптивний дизайн.

Форма складається з кількох секцій. У першій вводяться критерії, які додаються динамічно за допомогою кнопки, використовуючи JavaScript, яктй буде реалізований пізніше. Попарні порівняння критеріїв вводяться у вигляді числових оцінок у спеціальному контейнері.

Далі передбачено секцію для додавання альтернатив, де користувач вводить можливі варіанти вибору, які будуть оцінюватися. Завершує структуру введення оцінок альтернатив за кожним із критеріїв, що дозволяє сформувати повний набір даних для розрахунків.

Після заповнення форми користувач може надіслати її на сервер для обробки за допомогою відповідної кнопки. Дані обробляються з використанням методу AHP для обчислення ваг критеріїв, аналізу альтернатив і вибору найкращого варіанта. JavaScript забезпечує динамічне додавання елементів, а серверна частина (через POST-запит) виконує розрахунки.

Реалізація основної частини файлу подана в лістингу 3.6.

Лістинг 3.6.

```
<body class="bg-light">
    <h1 class="text-center text-primary py-4">Метод AHP - Введення критеріїв та варіантів</h1>
    <form method="POST" action="/calculate" class="container p-4 bg-white border rounded shadow-sm">
        <div id="criteria-container" class="mb-4 d-flex flex-column gap-1">
        </div>
        <button type="button" class="btn btn-outline-secondary btn-sm mb-4" onclick="addCriterion()">Додати критерій</button>

        <h3 class="text-secondary mb-3">Попарні порівняння критеріїв</h3>
        <div id="comparison-container" class="mb-4 d-flex flex-column gap-1">
<!-- Динамічні попарні порівняння будуть додаватися сюди -->
        </div>

        <h3 class="text-secondary mb-3">Варіанти</h3>
        <div id="options-container" class="mb-4 d-flex flex-column gap-1">
```

```
            </div>
            <button type="button" class="btn btn-outline-secondary
btn-sm mb-4" onclick="addOption()">Додати варіант</button>

            <h3 class="text-secondary mb-3">Оцінки варіантів</h3>
            <div id="alternatives-container" class="mb-4 d-flex flex-
column gap-1">
      <!-- Динамічно додаватимуться оцінки для альтернатив -->
            </div>

            <button type="submit" class="btn btn-success btn-lg w-
100">Розрахувати</button>
        </form>
    </body>
```

Для реалізації функцій addCriterion та addOption створюємо файл ahp_scripts.js в папці static. Ці функції забезпечать динамічне управління формою введення даних для методу AHP.

Для коректної роботи всіх функцій необхідно створити змінні criteriaCount та optionsCount для збереження інформації про кількість доданих критеріїв та альтернатив.

Функція addCriterion додає новий критерій до форми. Вона створює HTML-елемент з текстовим полем для введення назви критерію і кнопкою видалення. Після додавання критерію функція викликає updateComparisonFields і updateAlternativeFields, щоб оновити поля для попарних порівнянь та оцінок відповідно до нової кількості критеріїв. Код функції поданий в лістингу 3.7.

Лістинг 3.7.

```
    function addCriterion() {
        const criteriaContainer = document.getElementById('criteria-
container');
        const newCriterionDiv = document.createElement('div');
        newCriterionDiv.classList.add('criterion');
        const criterionId = `criteria_${criteriaCount}`;
        newCriterionDiv.innerHTML = `
            <label for="${criterionId}">Критерій ${criteriaCount +
1}:</label>
            <input type="text" id="${criterionId}"
name="criteria_${criteriaCount}" placeholder="Введіть критерій"
required>
            <button type="button" class="btn btn-danger btn-sm"
onclick="removeCriterion(this)">Видалити</button>
```

```
        `;

        criteriaContainer.appendChild(newCriterionDiv);
        criteriaCount++;

        updateComparisonFields();
        updateAlternativeFields();
    }
```

Для оновлення попарних порівнянь створена функція updateComparisonFields. Вона генерує матрицю парних порівнянь для критеріїв. Вона перебирає всі пари критеріїв, створює відповідні поля введення числових значень і додає їх до контейнера. Цей процес виконується щоразу, коли змінюється кількість критеріїв. Код функції подано в лістингу 3.8.

Лістинг 3.8.

```
    function updateComparisonFields() {
        const comparisonContainer =
document.getElementById('comparison-container');
        comparisonContainer.innerHTML = '';
        for (let i = 0; i < criteriaCount; i++) {
            for (let j = i + 1; j < criteriaCount; j++) {
                const comparisonDiv = document.createElement('div');
                comparisonDiv.classList.add('comparison');
                comparisonDiv.innerHTML = `
                    <label for="compare_${i}_${j}">Порівняти критерій
${i + 1} з критерієм ${j + 1}:</label>
                    <input type="number" name="comparison_${i}_${j}"
min="1" max="9" placeholder="Вага" required>
                `;
                comparisonContainer.appendChild(comparisonDiv);
            } } }
```

Видалення критеріїв реалізовано в функції removeCriterion, яка видаляє вибраний критерій із форми та оновлює кількість критеріїв. Вона також викликає оновлення полів для попарних порівнянь і оцінок, щоб забезпечити відповідність форми поточному стану. Код функції подано в лістингу 3.9.

Лістинг 3.9.

```
    function removeCriterion(button) {
        const criterionDiv = button.parentElement;
```

```
            criterionDiv.remove();
            criteriaCount--;
            updateComparisonFields();
            updateAlternativeFields();
        }
```

Наступною була реалізована функція додавання альтернатив addOption. Вона додає новий варіант (альтернативу) до форми. Функція створює блок HTML з текстовим полем для введення назви варіанта і кнопкою видалення. Як і з критеріями, після додавання альтернативи викликається функція updateAlternativeFields. Код функції подано в лістингу 3.10.

Лістинг 3.10.

```
        function addOption() {
            const optionsContainer = document.getElementById('options-
container');
            const newOptionDiv = document.createElement('div');
            newOptionDiv.classList.add('option');
            const optionId = `option_${optionsCount}`;
            newOptionDiv.innerHTML = `
                <label for="${optionId}">Варіант ${optionsCount +
1}:</label>
                <input type="text" id="${optionId}"
name="option_${optionsCount}" placeholder="Введіть варіант" required>
                <button type="button" class="btn btn-danger btn-sm"
onclick="removeOption(this)">Видалити</button>
            `;
            optionsContainer.appendChild(newOptionDiv);
            optionsCount++;
            updateAlternativeFields();
        }
```

Вище згадана функція updateAlternativeFields генерує секцію для введення оцінок кожної альтернативи за кожним критерієм. Вона перебирає всі альтернативи та критерії, створюючи відповідні поля введення числових значень, які додаються до контейнера. Код функції подано в лістингу 3.11.

Лістинг 3.11.

```
        function updateAlternativeFields() {
            const alternativesContainer =
document.getElementById('alternatives-container');
```

```
        alternativesContainer.innerHTML = '';
        const n_criteria = criteriaCount;
        const n_options = optionsCount;
        for (let i = 0; i < n_options; i++) {
            const alternativeDiv = document.createElement('div');
            alternativeDiv.classList.add('alternative');
            alternativeDiv.innerHTML = `<h4>Оцінки для варіанту ${i +
1}:</h4>`;
            for (let j = 0; j < n_criteria; j++) {
                const inputId = `alternative_${i}_${j}`;
                alternativeDiv.innerHTML += `
                    <label for="${inputId}">Оцінка за критерієм ${j +
1}:</label>
                    <input       type="number"       id="${inputId}"
name="alternative_${i}_${j}"   min="1"   max="9"   placeholder="Оцінка"
required>
                `;
            }
            alternativesContainer.appendChild(alternativeDiv); } }
```

Видалення альтернатив реалізоване подібно до видалення критеріїв. Функція removeOption видаляє обрану альтернативу з форми та оновлює кількість альтернатив. Вона також оновлює поля для введення оцінок альтернатив. Код функції подано в лістингу 3.12.

Лістинг 3.12.

```
function removeOption(button) {
    const optionDiv = button.parentElement;
    optionDiv.remove();
    optionsCount--;
    updateAlternativeFields();
}
```

Ці функції забезпечують інтерактивність форми, дозволяючи користувачам зручно вводити дані, потрібні для методу AHP. Вони автоматизують багато аспектів роботи з формою, мінімізуючи можливість помилок при введенні та полегшуючи налаштування структури моделі.

3.3.7. Реалізація сторінки виводу результатів для методу AHP

Для початку реалізації цього кроку створюємо файл results.html в папці templates.

У блоці head необхідно підключити зовнішні бібліотеки для стилізації (Bootstrap), побудови графіків (Chart.js), генерування PDF-файлів (pdfmake) та створення зображень з HTML-елементів (html2canvas). Ці бібліотеки дозволяють додавати необхідні функціональні можливості для відображення графіків і генерації звітів. Код блоку подано в лістингу 3.13.

Лістинг 3.13.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Результати AHP</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.
min.css"              rel="stylesheet"              integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.68/pdfmake.min
.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.68/vfs_fonts.j
s"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/1.4.1/html2can
vas.min.js"></script>
    <script src="{{ url_for('static', filename='ahp_scripts.js')
}}"></script>
</head>
```

Після цього потрібно реалізувати сам вивід результатів, які будуть передаватися функцією, яка формуватиме сторінку по заданому шаблону.

Формується таблиця з вагами критеріїв, яка відображає назву кожного критерію та його вагу, яку було розраховано на сервері. Веб-сторінка використовує шаблонну мову Jinja, щоб динамічно вставляти дані з об’єкта results, що містить критерії та їхні ваги. Цей блок генерує таблицю, де кожен рядок відповідає одному з критеріїв, і для кожного критерію виводиться його назва та вага, округлена до трьох знаків після коми.

Подібним чином відображаються оцінки альтернатив. Для кожної альтернативи виводиться її назва і оцінка, яку вона отримала на основі результатів методу AHP. Код блоку подано в лістингу 3.14.

Лістинг 3.14.

```
<!-- Таблиця з вагами критеріїв -->
        <h2 class="text-secondary mb-3">Ваги критеріїв</h2>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <th>Критерій</th>
                    <th>Вага</th>
                </tr>
            </thead>
            <tbody>
                {%      for      criterion,      weight      in
zip(results['criteria'], results['criteria_weights']) %}
                <tr>
                    <td>{{ criterion }}</td>
                    <td>{{ weight | round(3) }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <!-- Таблиця з вагами альтернатив -->
        <h2 class="text-secondary mb-3">Оцінки альтернатив:</h2>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <th>Альтернатива</th>
                    <th>Оцінка</th>
                </tr>
            </thead>
            <tbody>
                {%       for       option,       score       in
zip(results['alternatives'], results['alternative_scores']) %}
                <tr>
                    <td>{{ option }}</td>
                    <td>{{ score | round(3) }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
```

Для відображення розподілу ваг критеріїв у вигляді графіків використовується бібліотека Chart.js. Графік будується на основі отриманих даних за допомогою

функції drawCriteriaChart, яка отримує дані з сервера, що передаються через шаблонну мову.

Ще один графік, створений за допомогою Chart.js, відображає оцінки альтернатив. Він отримує дані про альтернативи та їх оцінки і генерує графік для візуалізації результатів.

Внизу сторінки розташовані кнопки для повернення на головну сторінку та для завантаження результатів у форматі PDF. Генерація PDF відбувається через функцію generatePDF, яка використовує бібліотеки для створення документів. Код блоку подано в лістингу 3.15.

Лістинг 3.15.

```
            <div class="my-4">
                <h2    class="text-secondary    mb-3">Розподіл    ваг
критеріїв</h2>
                <canvas   class="container   my-3"   style="max-width:
900px; max-height: 400px;" id="criteriaChart"></canvas>

                <script>
                    const         criteriaLabels         =         {{
results['criteria']|tojson }};
                    const         criteriaWeights         =         {{
results['criteria_weights']|tojson }};
                    console.log(criteriaLabels, criteriaWeights);
                    drawCriteriaChart(criteriaLabels,
criteriaWeights);
                </script>
            </div>

            <div class="my-3">
                <h2 class="text-secondary mb-3">Альтернативи</h2>
                <canvas  class="container"  style="max-width:  900px;
max-height: 400px;" id="alternativesChart" ></canvas>
                <script>
          const alternatives = {{ results['alternatives'] | tojson }};
          const scores = {{ results['alternative_scores'] | tojson }};
                    drawAlternativesChart(alternatives, scores);
                </script>
            </div>
        </div>
        <div class="text-center my-4">
            <a href="/" class="btn btn-secondary ">
                На головну
            </a>
    <button    onclick="generatePDF(results1)"    class="btn    btn-
primary">Завантажити PDF</button>
```

```
        </div>
        <br>
        <script>
            const results1 = {{ results | tojson }};
        </script>
```

Після цього в файлі ahp_scripts.js потрібно реалізувати функції, які викликаються на сторінці результатів, а саме – drawCriteriaChart, drawAlternativesChart та generatePDF.

Функція drawCriteriaChart відповідає за створення кругової діаграми (pie chart) для відображення ваг критеріїв. В змінну записується елемент canvas, створений в HTML-файлі, після цього створюється новий графік, якому задаються характеристики, такі як масиви даних та назв критеріїв, кольори, товщина ліній, тощо. Код функції подано в лістингу 3.16.

Лістинг 3.16.

```
    function drawCriteriaChart(criteriaLabels, criteriaWeights) {
        const ctx = document.getElementById('criteriaChart').getContext('2d');
        new Chart(ctx, {
            type: 'pie',
            data: {
                labels: criteriaLabels,
                datasets: [{
                    label: 'Ваги критеріїв',
                    data: criteriaWeights,
                    backgroundColor: [
                        'rgba(75, 192, 192, 0.2)',
                    ],
                    borderColor: [
                        'rgba(153, 102, 255, 1)',
                    ],
                    borderWidth: 1
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                scales: {
                }
            }
        });
    }
```

Принцип роботи функції побудови графіків альтернатив drawAlternativesChart подібна до функції побудови критеріїв, за винятком деяких специфічних налаштувань графіку альтернатив.

Функція generatePDF створює PDF-документ, який містить результати методу AHP, включаючи таблиці з вагами критеріїв і оцінками альтернатив, а також зображення графіків. Спочатку графіки конвертуються в зображення за допомогою бібліотеки html2canvas. Зображення зберігаються в змінних у форматі Base64. Для створення PDF використовується бібліотека pdfMake, яка формує структуру документа, що включає таблиці з даними (ваги критеріїв і оцінки альтернатив) та зображення графіків. Документ генерується та завантажується у форматі PDF. Код функції подано в лістингу 3.17.

Лістинг 3.17.

```
      function generatePDF(results) {
      html2canvas(document.getElementById('criteriaChart')).then(funct
ion(canvas) {
              const chart1Image = canvas.toDataURL();
      html2canvas(document.getElementById('alternativesChart')).then(f
unction(canvas) {
              const chart2Image = canvas.toDataURL();
              const docDefinition = {
                  content: [
                     { text: 'Результати методу AHP', style: 'header' },
                     { text: 'Ваги критеріїв', style: 'subheader' },
                     { table: {
                                 body: [
                                 ['Критерій', 'Вага'],
                  ...results.criteria.map((criterion, index) => [
                                    criterion,
                      results.criteria_weights[index].toFixed(3)
                                    ]) ] } },
                     { text: 'Оцінки альтернатив', style: 'subheader' },
                     { table: {
                            body: [
                                 ['Альтернатива', 'Оцінка'],
                     ...results.alternatives.map((option, index) => [
                                       option,
results.alternative_scores[index].toFixed(3)
                                    ]) ] } },
                   {        text:         `Найкраща         альтернатива:
${results.best_alternative}`, style: 'subheader' },
                  { text: 'Розподіл ваг критеріїв', style: 'subheader' },
                  { image: chart1Image,
```

```
                            width: 500 },
                        { text: 'Альтернативи', style: 'subheader' },
                        { image: chart2Image,
                            width: 500 } ],
                styles: {
                header: { fontSize: 18, bold: true, alignment: 'center' },
        subheader: { fontSize: 14, bold: true, margin: [0, 10, 0, 5] } }
};
            pdfMake.createPdf(docDefinition).download('results.pdf');
                });
            });
        }
```

3.3.8. Поєднання створених компонентів для методу AHP

Для того, щоб всі вище описані компоненти працювали, необхідно додати логіку в основний файл app.py. В ньому потрібно створити декоратор, який визначить маршрути для сторінки вводу даних та виводу результатів.

Декоратор сторінки вводу даних для методу AHP направляє користувача на стоворену сторінку ahp_input_form.html.

Функція направлення користувача на сторінку результатів працює слкданіше. Вона обробляє дані, що були відправлені з форми вводу AHP. За допомогою циклів витягуються критерії, варіанти, попарні порівняння та оцінки альтернатив, перевіряючи ключі у формі, які починаються з певних префіксів (наприклад, criteria_, option_, comparison_, alternative_). Ці дані передаються в функцію ahp, яка виконує необхідні обчислення. Якщо під час обчислень виникає помилка, відображається повідомлення про помилку. Якщо все проходить успішно, результати передаються в шаблон results.html, де вони відображаються. Код функцій подано в лістингу 3.18.

Лістинг 3.18.

```
@app.route("/ahp_input", methods=["GET", "POST"])
def ahp_input():
    return render_template("ahp_input_form.html")

@app.route('/calculate', methods=['POST'])
def calculate_ahp():
    criteria = []
    for key in request.form:
        if key.startswith("criteria_"):
```

```
                criteria.append(request.form[key])
        options = []
        for key in request.form:
            if key.startswith("option_"):
                options.append(request.form[key])
        comparisons = {}
        for key in request.form:
            if key.startswith("comparison_"):
                comparisons[key] = float(request.form[key])
        alternatives = {}
        for key in request.form:
            if key.startswith("alternative_"):
        try:
            results    =    ahp(criteria,    options,    comparisons,
alternatives)
        except Exception as e:
            return f"Помилка: {e}"
        return    render_template("results.html",    results=results,
zip=zip)
```

Таким чином, метод аналізу ієрархій можна вважати повністю реалізованим.

3.3.9. Спільні аспекти реалізації інтерфейсів вводу та виводу в методах AHP, Fuzzy MCDM та WSM

Логіка реалізації сторінок вводу та виводу для методів Fuzzy MCDM та (WSM) є аналогічною тій, що використовується для методу AHP. Основна відмінність між цими методами полягає лише в специфікаціях полів вводу та результатах виводу, що відповідають конкретним вимогам кожного з алгоритмів, а також у перетворенні даних всередині файлу app.py для передачі їх у функцію алгоритму. Наприклад, для методу Fuzzy MCDM користувач вводить нечіткі оцінки для альтернатив, а для WSM – ваги критеріїв та оцінки альтернатив за кожним критерієм. Обробка цих даних в app.py здійснюється відповідно до особливостей кожного методу, однак сам підхід до збору введених значень з форм та їх передачі в алгоритм є однаковим.

3.4. Тестування проєкту

Наступним після розробки йде етап тестування. Мета тестування розробленої програми полягає в забезпеченні її коректної роботи, відповідності функціональних

можливостей вимогам проєкту та гарантуванні стабільності в різних умовах.

Крім того, тестування допомагає переконатися, що програма є інтуїтивно зрозумілою для користувача, навіть за умови некоректного вводу чи помилкових дій. Перевірка всіх етапів, від введення критеріїв і ваг до генерації графіків і PDF-звітів, гарантує, що система працює так, як було заплановано, і відповідає заданим цілям.

3.4.1. Тестування головної сторінки додатку

Для перевірки роботи головної сторінки необхідно запустити локальний сервер та перейти за посиланням, вказаним у командному рядку. Після цього в браузері ми зможемо побачити головну сторінку додатку.

На сторінці реалізована проста анімація при наведенні курсору на блок з назвою методу.

Перевіряємо правильність роботи цієї функції – рисунок 3.6. Перевірка роботи анімації при наведенні курсору.

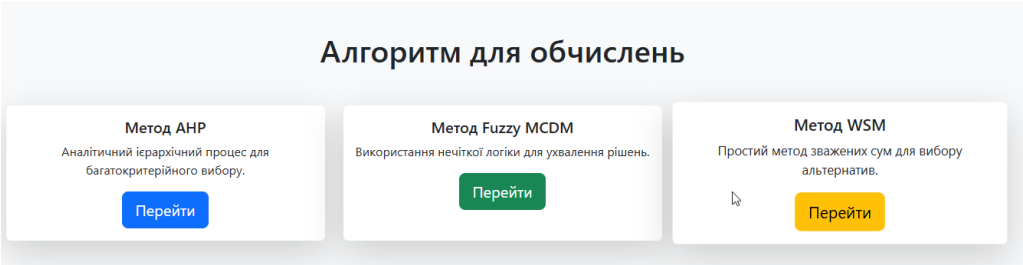


Рис.3.6. Перевірка роботи анімації при наведенні курсору

Після цього потрібно перевірити сам функціонал кнопок. Вони повинні відкривати сторінки з формами вводу відповідного методу.

Даний етап тестування був успішно пройдений, кнопки переходу на відповідні форми вводу даних для обислень працюють коректно.

3.4.2. Тестування вводу даних та отримання результатів для методу аналізу ієрархій

Щоб перевірити правильність роботи форми вводу для методу AHP необхідно перейти на відповідну сторінку за допомогою кнопки на головній сторінці. Після

цього повинна відкритися сторінка зі правильною формою – рисунок 3.7. Вигляд сторінки «Форма вводу даних для методу AHP».

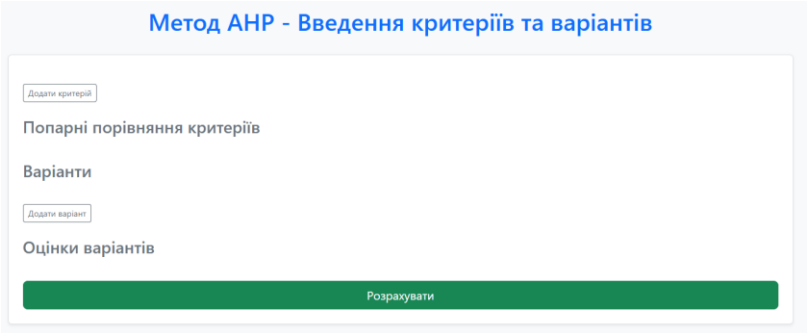


Рис.3.7. Вигляд сторінки «Форма вводу даних для методу AHP»

На сторінці можна побачити три кнопки: «Додати критерій», «Додати варіант» та «Розрахувати». При натисканні кнопки додавання критерію додається нове поле вводу, куди можна вписати назву критерію. Біля цього поля вводу присутня кнопка видалення критерію. Також під написом «Попарні порівняння критеріїв» з'являється поле для вводу порівняльних оцінок для кожного критерію.

Після натискання кнопки додавання варіанту, на сторінці з'являється поле для вводу назви альтернативи. Біля неї також є кнопка для видалення з відповідним написом. Також під написом «Оцінки варіантів» з'являється поле для вводу оцінок доданих альтернатив за введеними критеріями. Форма з доданими полями для критеріїв та альтернатив зображена на рисунку 3.8. Форма вводу для методу AHP з доданими полями.

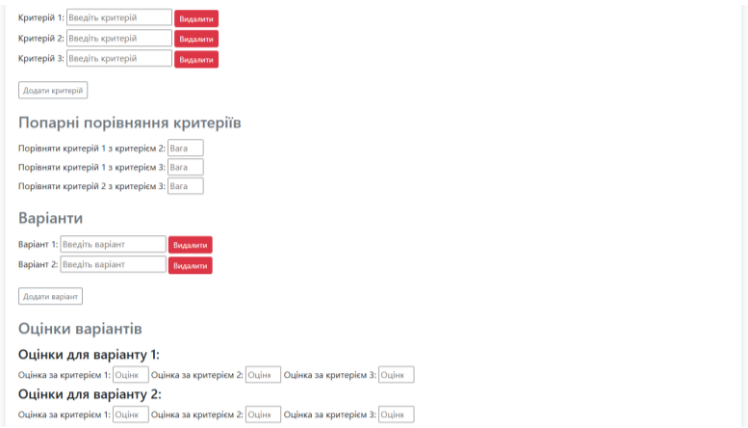


Рис.3.8. Форма вводу для методу AHP з доданими полями.

При спробі вписати літери в поля, призначені для введення оцінок критеріїв або альтернатив, з’являється підказка, яка повідомляє, що в дане поле можна ввести тільки числові значення.

Повідомлення про некоректність даних зображено на рисунку 3.9.

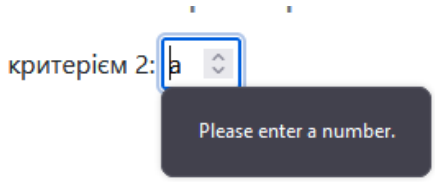


Рис.3.9. Повідомлення про некоректність даних

Після заповнення полів відповідними тестовими даними та натискання кнопки «Розрахувати» сервер опрацьовує їх та повертає рехультати обчислення алгоритму. Після цього користувач перенаправляється на сторінку виведення результатів – рисунок 3.10. Сторінка результатів методу AHP.

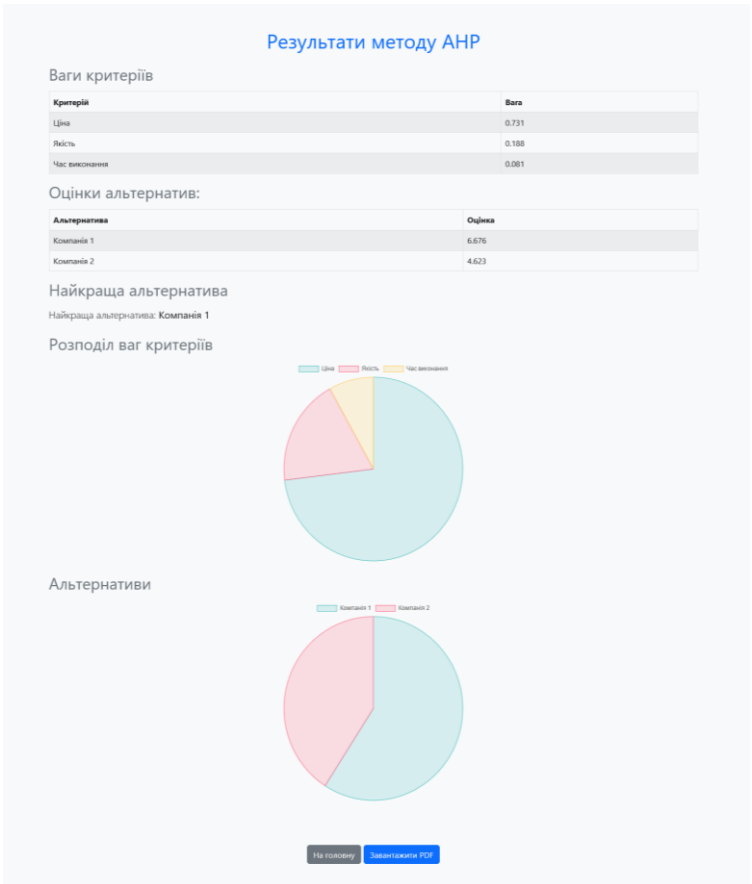


Рис.3.10. Сторінка результатів методу AHP

На сторінці можна побачити таблиці з вагами критеріїв та оцінками альтернатив, найкращу альтернативу за результатами обчислень та графіки розподілу ваг критеріїв та альтарнатив.

Внизу сторінки розміщені кнопки «На головну» та «Завантажити PDF». Кнопки були перевірені та зроблений висновок, що вони працюють коректно. Після натискання кнопки «На головну» відкрилася головна сторінка з вибором методу.

Після натискання на кнопку для завантаження PDF у вікні завантажень браузера з’явився файл з результатами обчислень, які зображені на поточній сторінці – рисунок 3.11. Завантаження PDF-файлу з результатами.

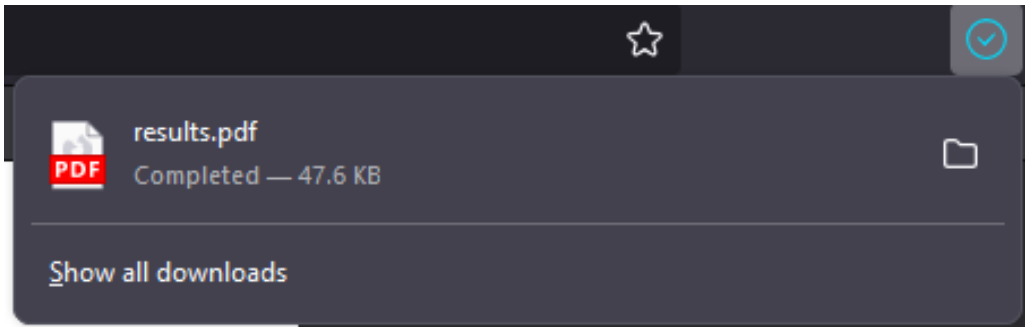


Рис.3.11. Завантаження PDF-файлу з результатами

Зміст файлу відповідає інформації на стоірнці, де він був сформований.

Після тестування роботи додатку з методом AHP був зроблений висновок, що всі функції на формі введення даних (додавання критеріїв та альтернатив, їх видалення, введеня оцінок альтернатив та ваг критеріїв) та на сторінці результатів працюють правильно.

3.4.3. Тестування вводу даних та отримання результатів для методу Fuzzy MCDM

Після переходу з головної сторінки на форму введення даних для методу Fuzzy MCDM відкривається відповідна форма. Основний функціонал, такий як додавання критеріїв та альтернатив, введення ваг критеріїв та оцінок альтернатив, працює подібно до сторінки введення даних для методу AHP.

Однак формат даних відрізняється, оскільки даних метод працює з нечіткими даними. Тому для кожного критерію існує три поля вводу для введення нижньої межі, середнього значення та верхньої межі. Внаслідок цього альтернативи також оцінюються за цими тьома параметрами для кожного критерію.

При введенні некоректних даних також з’являється підказка з повідомленням про це.

Після перевірки кнопок додавання різних елементів та підтвердження їх правильної роботи можна заповнити форму тестовими даними – рисунок 3.12. Заповнення форми вводу Fuzzy MCDM тестовими даними.

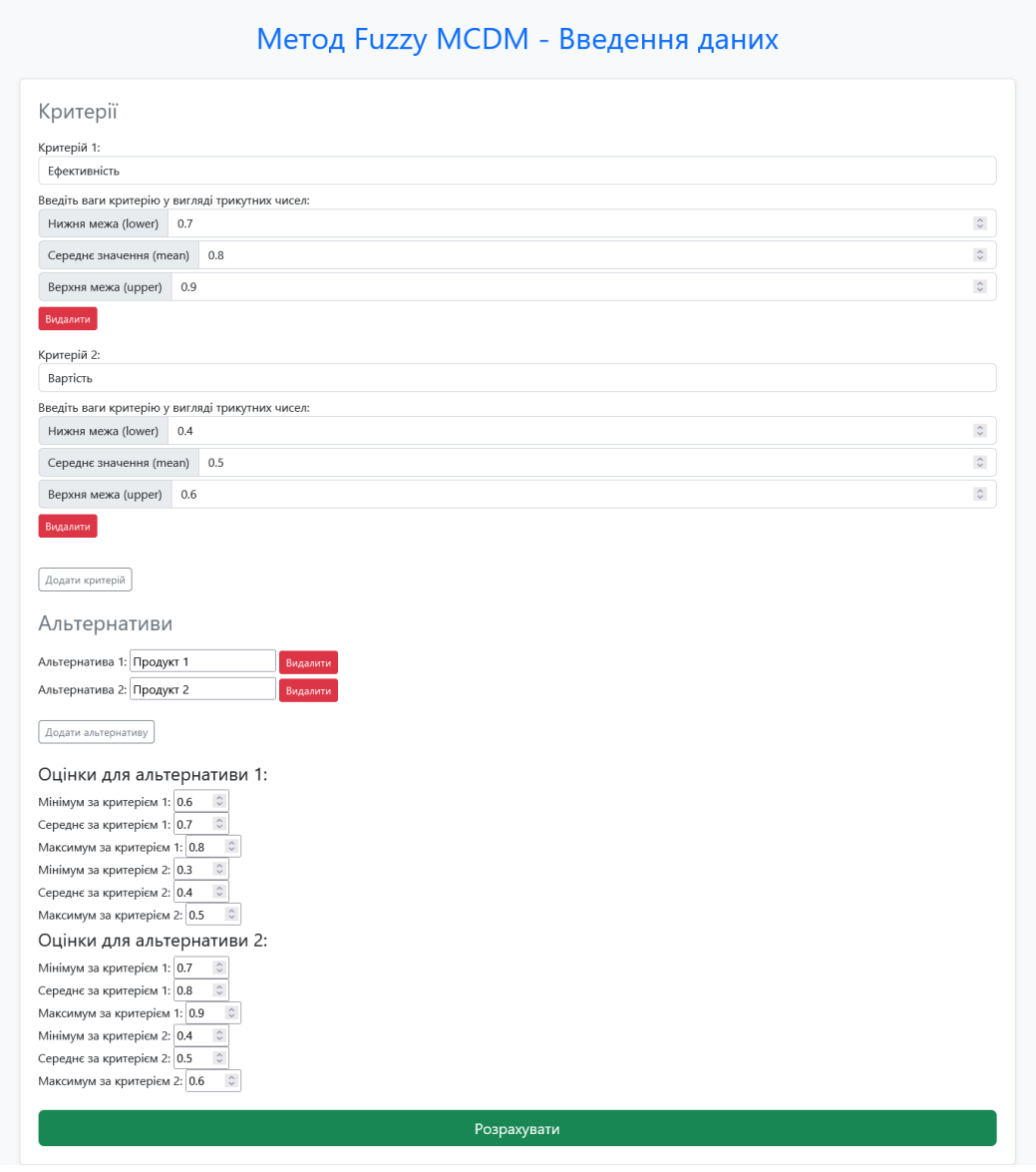


Рис.3.12. Заповнення форми вводу Fuzzy MCDM тестовими даними

Після натискання кнопки «Розрахувати» система використовує відповідний алгоритм для обчислення результатів та перенаправляє користувача на сформовану сторінку з результатами для методу Fuzzy MCDM.

На сторінці результатів, як і у випадку з методом AHP, можна побачити ваги критеріїв, оцінки альтернатив та найкращу альтернативу за результатами обчислень.

Оскільки в кожного критерію є три значення (верхня межа, середнє значення та нижня межа), графік розподілу ваг критеріїв оформлений у вигляді стовпчикової діаграми. Графік альтернатив розміщений нижче.

Вигляд сторінки результатів зображено на рисунку 3.13. Сторінка результатів методу Fuzzy MCDM.

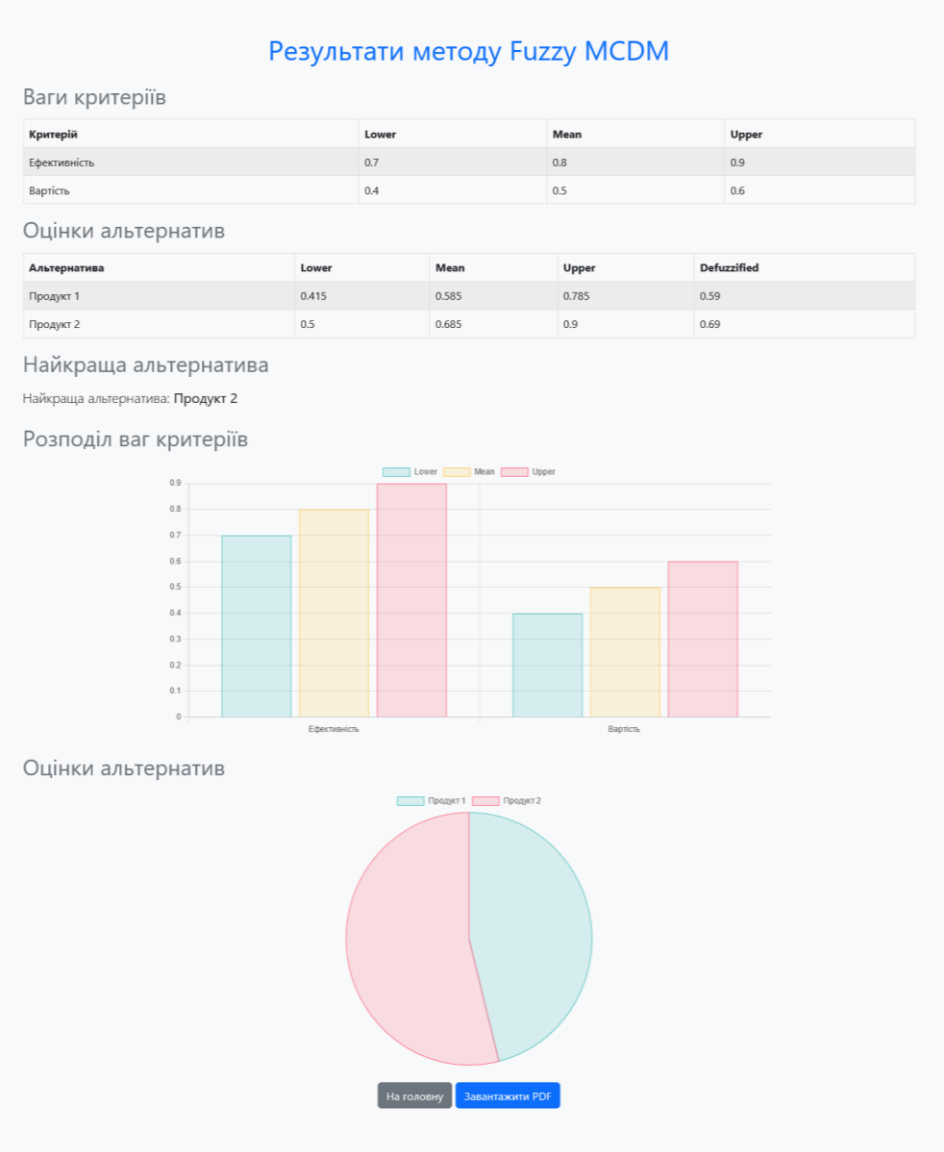


Рис.3.13. Сторінка результатів методу Fuzzy MCDM

Кнопки для повернення на головну сторінку та формування PDF-файлу протестовані та зроблений висновок, що вони працюють правильно. Зміст згенерованого файлу відповідає результатам на сторінці.

3.4.4. Тестування вводу даних та отримання результатів для методу WSM

Після переходу з головної сторінки на форму введення даних для методу Fuzzy MCDM відкривається відповідна форма. . Основний функціонал, такий як додавання критеріїв та альтернатив, введення ваг критеріїв та оцінок альтернатив, працює подібно до сторінки введення даних для методів, описаних вище.

При введенні некоректних даних також з’являється підказка з повідомленням про це. Після успішного тестування кнопок для додавання різних елементів та підтвердження їх коректної роботи, можна перейти до заповнення форми тестовими значеннями – Рисунок 3.14. Заповнення форми вводу WSM тестовими даними.

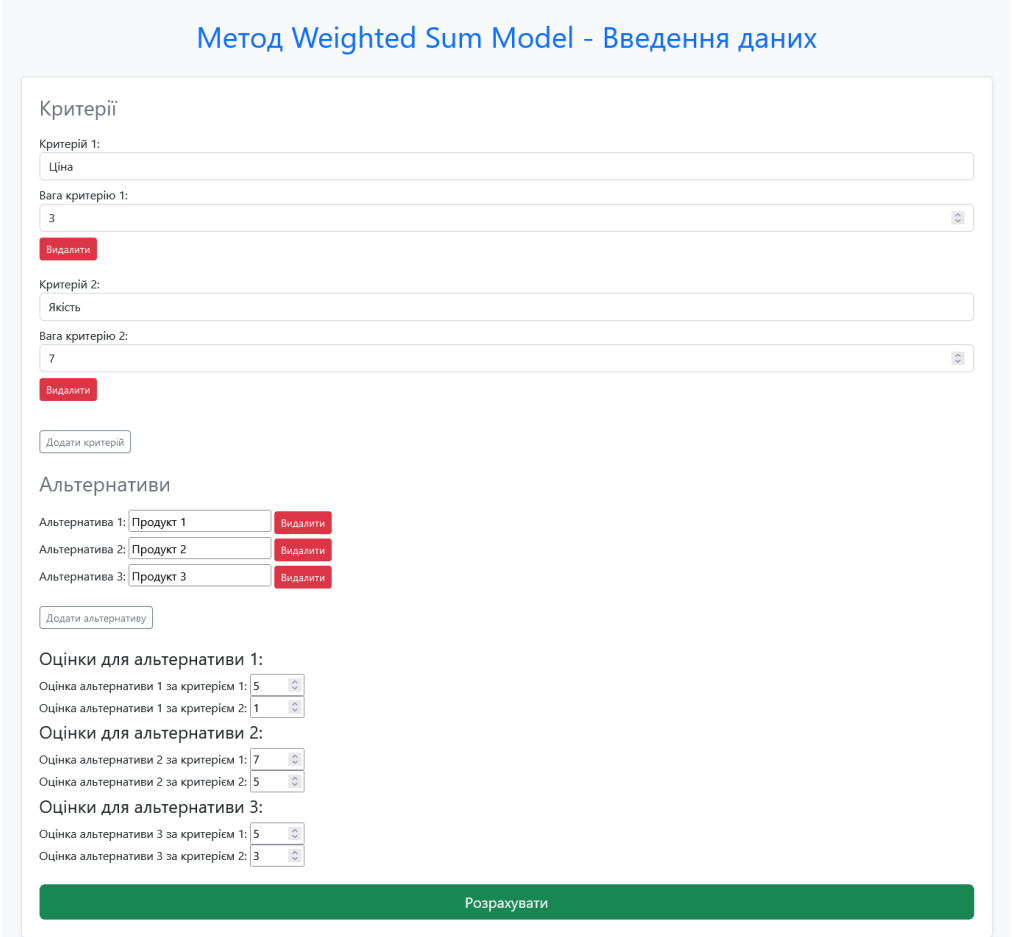


Рис.3.14. Заповнення форми вводу WSM тестовими даними

Після натискання кнопки «Розрахувати» введені дані обробляються за допомогою відповідного алгоритму та сервер перенаправляє користувача на стрінку з результатами обчислень для методу зважених сум.

На сторінці результатів, подібно до двох вище описаних методів, зображені ваги критеріїв, оцінки альтернатив та найкращу альтернативу за результатами обчислень. Графіки ваг критеріїв та альтернатив оформлені у вигляді секторних діаграм.

Вигляд сторінки результатів зображено на рисунку 3.15. Сторінка результатів методу WSM.

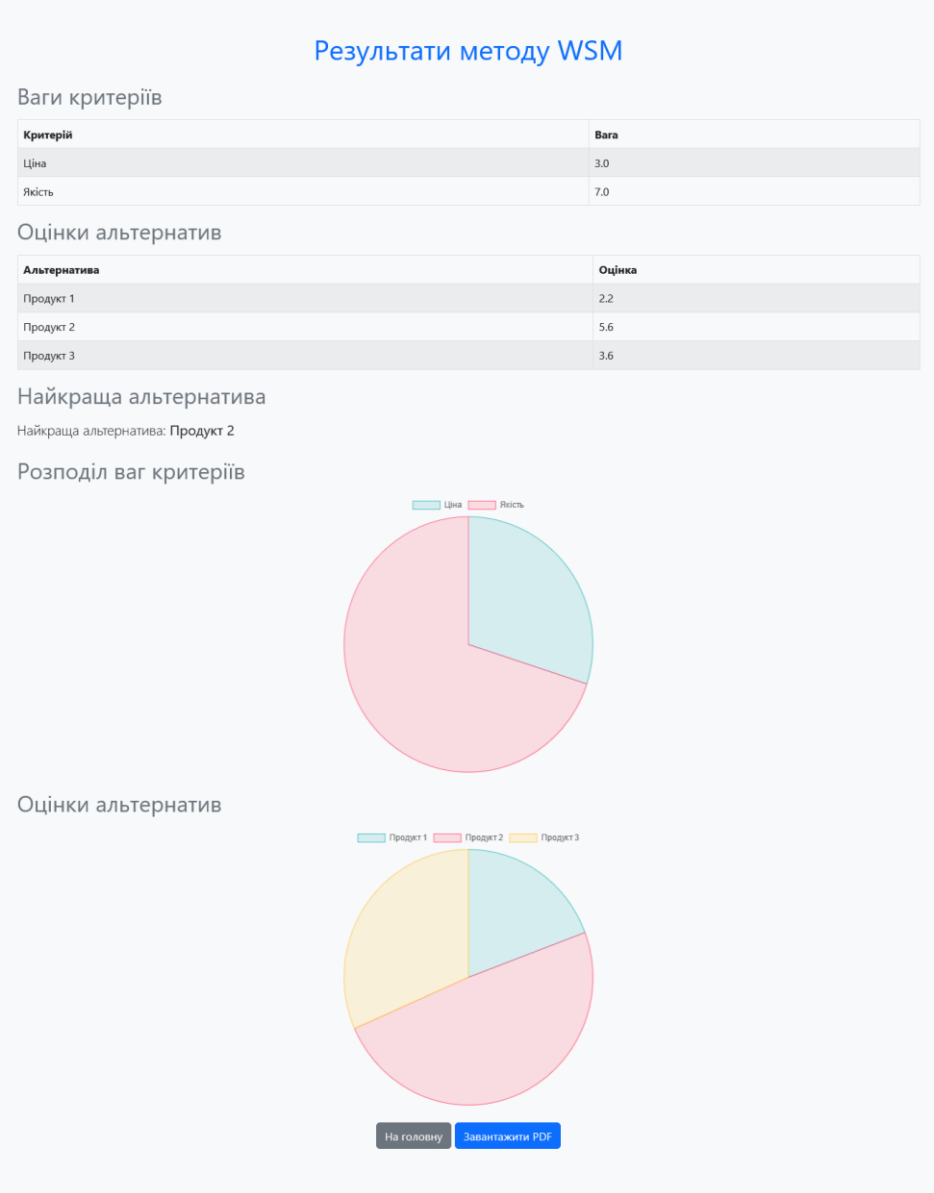


Рис.3.15. Сторінка результатів методу WSM

Функціональність кнопок для повернення на головну сторінку та створення PDF-файлу була перевірена, і підтверджено, що вони працюють належним чином. Зміст згенерованого PDF-файлу повністю відповідає відображеним на сторінці результатам.

Повні знімки всіх сторінок додатку та згенерованих PDF-файлів наведені в додатку А.

3.5. Завантаження проєкту на GitHub

За допомогою програмного забезпечення Visual Studio Code на спеціальній вкладці авторизуємось на свій обліковий запис, створюємо новий репозиторій, додаємо туди необхідні файли, та за допомогою спеціальної кнопки завантажуємо проект на GitHub.

Завантажений проєкт доступний за посиланням: https://github.com/VitaliyTuz/diploma_m.

3.6. Висновок до розділу

У розділі детально описано процес розробки програмного забезпечення для підтримки багатокритеріального прийняття рішень, яке інтегрує методи AHP, Fuzzy MCDM та WSM. Було проведено аналіз вимог, сформульовано архітектуру системи, реалізовано алгоритми обчислень та створено інтерфейси користувача.

Особливу увагу приділено інтеграції сучасних веб-технологій, таких як Flask, HTML, CSS, JavaScript та Bootstrap, для створення інтуїтивно зрозумілого й адаптивного інтерфейсу. Реалізовані рішення включають динамічні форми вводу, які дозволяють користувачам вводити дані про критерії та альтернативи, здійснювати попарні порівняння та генерувати результати у вигляді таблиць, графіків і PDF-звітів.

Ключовим елементом розділу є опис інтеграції алгоритмів у загальну структуру програми. В алгоритмах реалізовані сучасні підходи до обробки даних, що дозволяють ефективно працювати з великим числом критеріїв і альтернатив.

Зазначено можливості розширення системи, що дозволяє додавати нові методи та модулі.

Проведено тестування основних функціональних компонентів програми, включаючи коректність обчислень, генерацію звітів та інтерактивність інтерфейсу. Результати тестування підтвердили стабільність і ефективність роботи розробленої системи, що забезпечує її готовність до використання у реальних задачах.

ВИСНОВКИ

У даній магістерській роботі було досліджено сучасні веб-технології, алгоритми підтримки прийняття рішень та їх інтеграцію для створення ефективних інформаційних систем. У роботі приділено значну увагу основним інструментам веб-розробки, зокрема мові розмітки HTML, мові стилів CSS та мові програмування JavaScript. Розглянуто їхні особливості, структуру, а також взаємодію між собою для створення динамічного та адаптивного веб-контенту. Проаналізовано бібліотеки Chart.js, pdfMake і html2canvas, які використовуються для візуалізації даних, створення PDF-документів та захоплення вмісту сторінок у вигляді зображень. Було продемонстровано, що ці інструменти значно спрощують процес створення інтерактивних веб-сторінок.

Було проаналізовано основні класичні критерії, такі як критерії Вальда, Гурвіца, Лапласа та максимакс. Критерій Вальда орієнтований на мінімізацію ризиків, пропонуючи песимістичний підхід до прийняття рішень. Критерій Гурвіца дозволяє балансувати між песимізмом і оптимізмом за допомогою коефіцієнта α, що робить його більш гнучким у порівнянні з іншими методами. Критерій Лапласа, орієнтований на рівноймовірний розподіл можливих результатів, виявився корисним у випадках відсутності додаткової інформації про ймовірності. Метод максимакс акцентує увагу на досягненні найвищих можливих виграшів і є оптимістичним підходом, що може бути ефективним у ситуаціях з низьким рівнем ризику.

Досліджено всі ключові етапи процесу прийняття рішень: визначення цілей, аналіз проблем, пошук і оцінка альтернатив, вибір оптимального рішення та оцінка його ефективності. Було показано, що правильне структурування процесу прийняття рішень дозволяє підвищити точність і обґрунтованість обраних варіантів, що є критично важливим для складних проектів і систем з обмеженими ресурсами або високим рівнем невизначеності.

Практична цінність роботи полягає у створенні бази знань, яку можна застосовувати для розробки інформаційних систем, що підтримують прийняття рішень у різних сферах, зокрема в управлінні, економіці та технічних дисциплінах.

Отримані результати можуть бути використані для розробки веб-додатків, які не лише забезпечують зручний інтерфейс користувача, а й підтримують ефективний аналіз даних і прийняття рішень. Проведене дослідження демонструє можливості інтеграції сучасних веб-технологій з теорією прийняття рішень для створення інноваційних рішень, що відповідають актуальним викликам і потребам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Анікін В.К., Крилов Є.В., Пасько В.П. Теорія прийняття рішень. Навчальний посібник. Київ, 2023 – 134 с.

2. Madzík P, Falát L (2022) State-of-the-art on analytic hierarchy process in the last 40 years: Literature review based on Latent Dirichlet Allocation topic modelling. PLoS ONE 17(5): e0268777. https://doi.org/10.1371/journal.pone.0268777

3. Dymova, L.; Kaczmarek, K.; Sevastjanov, P.; Kulawik, J. A Fuzzy Multiple Criteria Decision Making Approach with a Complete User Friendly Computer Implementation. Entropy 2021, 23, 203. https://doi.org/10.3390/e23020203

4. HTML – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/HTML

5. HTML Documentation – Mozilla Developer Network – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML

6. HTML Tutorial – W3Schools. – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/html/

7. CSS – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/CSS

8. CSS Documentation – Mozilla Developer Network – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS

9. CSS Tutorial – W3Schools. – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/css/

10. JavaScript – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/JavaScript

11. JavaScript Documentation – Mozilla Developer Network. – [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

12. JavaScript Tutorial – W3Schools. – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/js/

13. Chart.js – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Chart.js

14. Chart.js Documentation – Chart.js. – [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/docs/latest/

15. Plot with Chart.js – W3Schools. – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/graphics/plot_chartjs.asp

16. PDFMake Repository. – GitHub. – [Електронний ресурс]. – Режим доступу: https://github.com/bpampuch/pdfmake

17. PDFMake Documentation. – [Електронний ресурс]. – Режим доступу: https://pdfmake.github.io/docs/0.3/

18. Html2Canvas Repository. – GitHub. – [Електронний ресурс]. – Режим доступу: https://github.com/niklasvh/html2canvas

19. Html2Canvas Documentation. – [Електронний ресурс]. – Режим доступу: https://html2canvas.hertzen.com/documentation

20. Bootstrap – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

21. Bootstrap Introduction – GetBootstrap. – [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/

22. Python – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Python_(programming_language)

23. Python Documentation. – [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/

24. Python Tutorial – W3Schools. – [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/python/

25. Visual Studio Code – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Visual_Studio_Code

26. Visual Studio Code Documentation. – [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs

27. Introduction to Visual Studio Code – Microsoft. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/training/modules/introduction-to-visual-studio-code/1-introduction

28. Visual Studio vs Visual Studio Code – Dev.to. – [Електронний ресурс]. – Режим доступу: https://dev.to/angelocodes/visual-studio-vs-visual-studio-code-an-in-depth-comparison-2eon

29. Git – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Git

30. Branching and Merging – Git. – [Електронний ресурс]. – Режим доступу: https://git-scm.com/about/branching-and-merging

31. Git Documentation. – [Електронний ресурс]. – Режим доступу: https://git-scm.com/docs

32. Decision-Making – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Decision-making

33. Дмитрієнко В. Д. Засоби та алгоритми прийняття рішень / Д 53 В. Д. Дмитрієнко, О. Ю. Заковоротний, В. І. Носков: навчально-методичний посібник до практичних занять. –Х.: НТМТ, 2013. – 76 с.

34. Analytic hierarchy process – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Analytic_hierarchy_process

35. The AHP Method: Definition and Example. – Indeed.com. – [Електронний ресурс]. – Режим доступу: https://www.indeed.com/career-advice/career-development/ahp-method

36. What is the Analytic Hierarchy Process (AHP)? – 1000minds – [Електронний ресурс]. – Режим доступу: https://www.1000minds.com/decision-making/analytic-hierarchy-process-ahp

37. Multiple-criteria decision analysis – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Multiple-criteria_decision_analysis

38. Ta-Chung Chu, Yichen Lin, An extension to fuzzy MCDM, Computers & Mathematics with Applications, Volume 57, Issue 3, 2009, Pages 445-454, ISSN 0898-1221, https://doi.org/10.1016/j.camwa.2008.10.076.

39. Definite / BOSDA – Spatial Information Laboratory – [Електронний ресурс]. – Режим доступу: https://spinlab.vu.nl/support/tools/definite-bosda/

40. MCDA Calculator: A Streamlined Decision Support System for Multi-Criteria Decision Analysis – [Електронний ресурс]. – Режим доступу: https://mcda-calculator.psi.ch/assets/MCDA_Calculator_v302.pdf

41. Expert Choice – Wikipedia, the free encyclopedia. – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Expert_Choice

42. Geldermann, Jutta & Zhang, Kejing. (2001). Software review: “Decision Lab 2000”. Journal of Multi-Criteria Decision Analysis. 10. 317 - 323. 10.1002/mcda.311.

43. Super Decisions – [Електронний ресурс]. – Режим доступу: https://www.superdecisions.com/

ДОДАТКИ

Додаток А

Знімки готового додатку

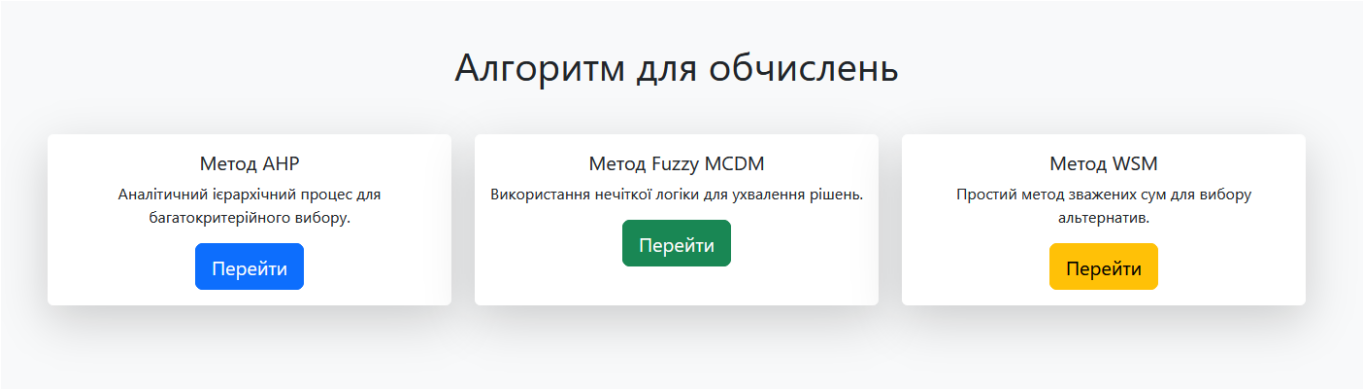


Рис. А.1. Головна сторінка додатку

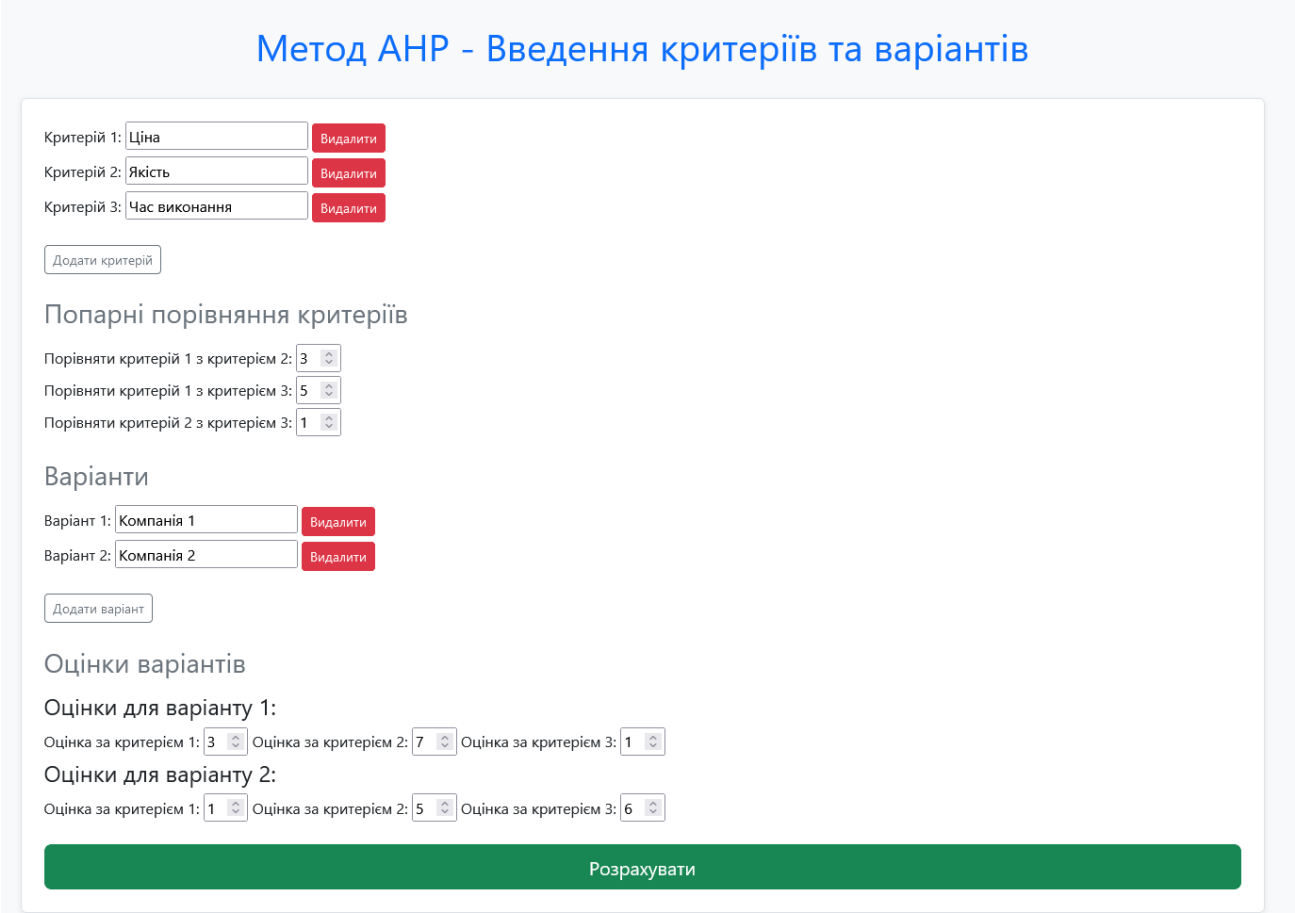


Рис. А.2. Сторінка «Метод AHP - Введення критеріїв та варіантів»

Результати методу AHP

Ваги критеріїв

Критерій	Вага
Ціна	0.659
Якість	0.185
Час виконання	0.156

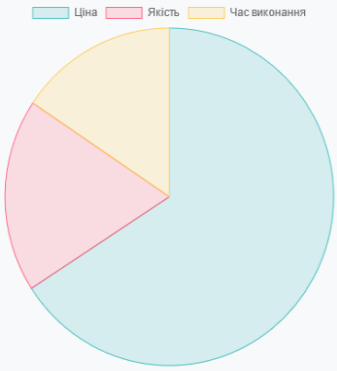
Оцінки альтернатив:

Альтернатива	Оцінка
Компанія 1	3.428
Компанія 2	2.522

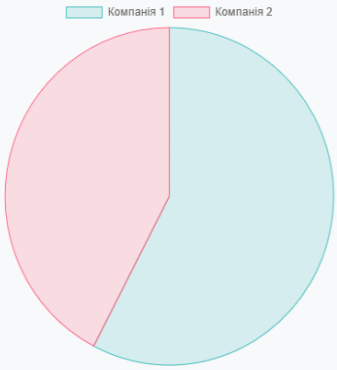
Найкраща альтернатива

Найкраща альтернатива: Компанія 1

Розподіл ваг критеріїв



Альтернативи



На головну Завантажити PDF

Рис. А.3. Сторінка результатів для методу AHP

Результати методу AHP

Ваги критеріїв

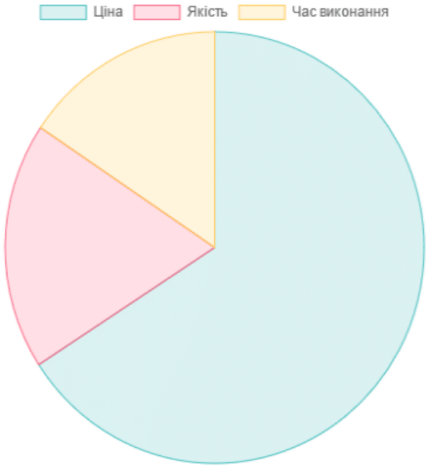
Критерій	Вага
Ціна	0.659
Якість	0.185
Час виконання	0.156

Оцінки альтернатив

Альтернатива	Оцінка
Компанія 1	3.428
Компанія 2	2.522

Найкраща альтернатива: Компанія 1

Розподіл ваг критеріїв



Альтернативи

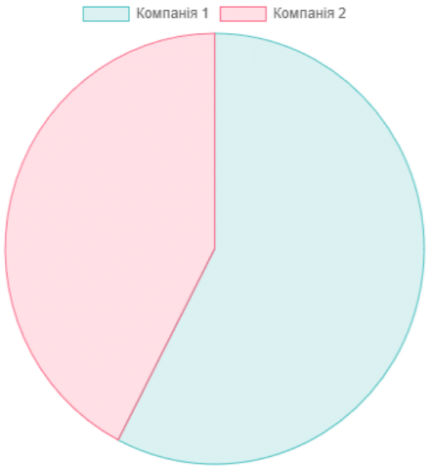


Рис. А.4. Вигляд згенерованого PDF-файлу для результатів методу AHP

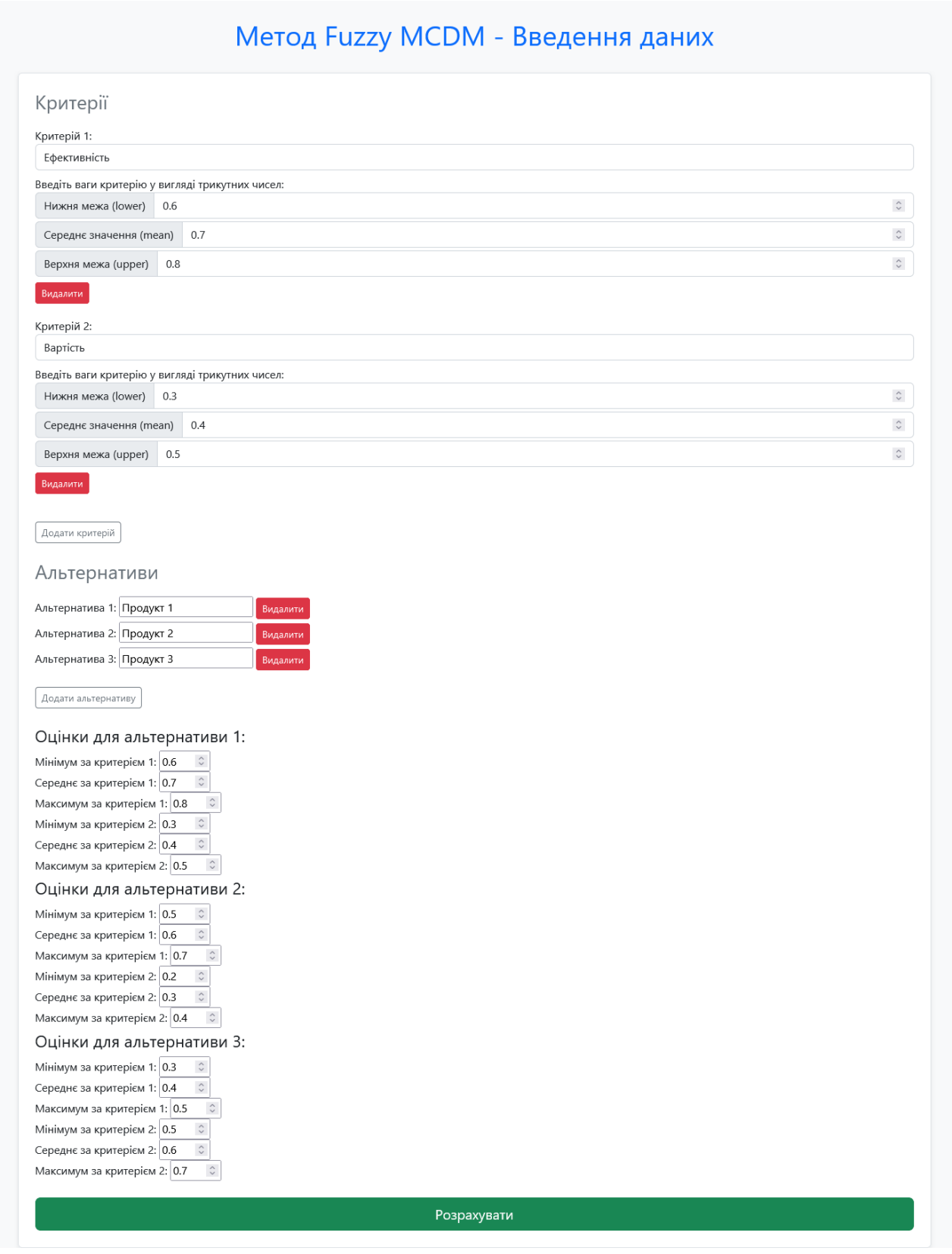


Рис. А.5. Сторінка «Метод Fuzzy MCDM - Введення даних»

Результати методу Fuzzy MCDM

Ваги критеріїв

Критерій	Lower	Mean	Upper
Ефективність	0.6	0.7	0.8
Вартість	0.3	0.4	0.5

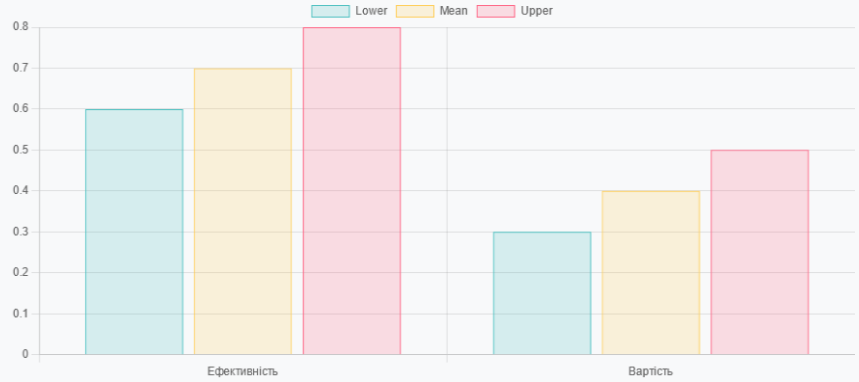
Оцінки альтернатив

Альтернатива	Lower	Mean	Upper	Defuzzified
Продукт 1	0.409	0.591	0.809	0.597
Продукт 2	0.327	0.491	0.691	0.497
Продукт 3	0.3	0.473	0.682	0.479

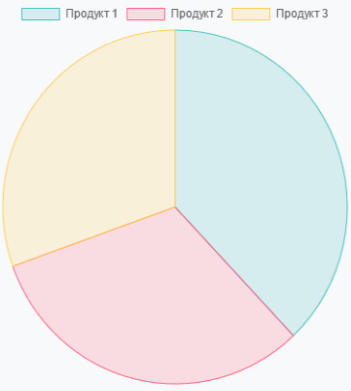
Найкраща альтернатива

Найкраща альтернатива: **Продукт 1**

Розподіл ваг критеріїв



Оцінки альтернатив



На головну | Завантажити PDF

Рис. А.6. Сторінка результатів для методу Fuzzy MCDM

Результати методу Fuzzy MCDM

Ваги критеріїв

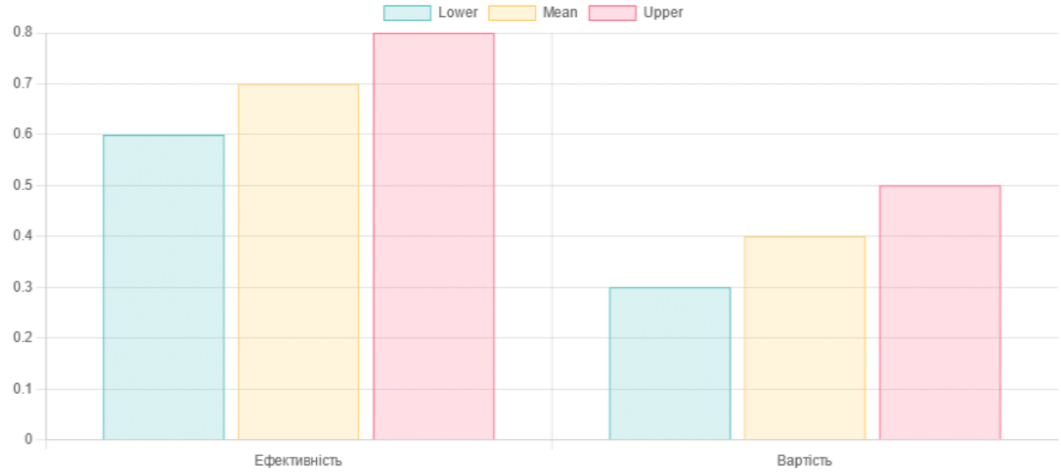
Критерій	Lower	Mean	Upper
Ефективність	0.600	0.700	0.800
Вартість	0.300	0.400	0.500

Оцінки альтернатив

Альтернатива	Lower	Mean	Upper	Defuzzified
Продукт 1	0.409	0.591	0.809	0.597
Продукт 2	0.327	0.491	0.691	0.497
Продукт 3	0.300	0.473	0.682	0.479

Найкраща альтернатива: Продукт 1

Розподіл ваг критеріїв



Оцінки альтернатив

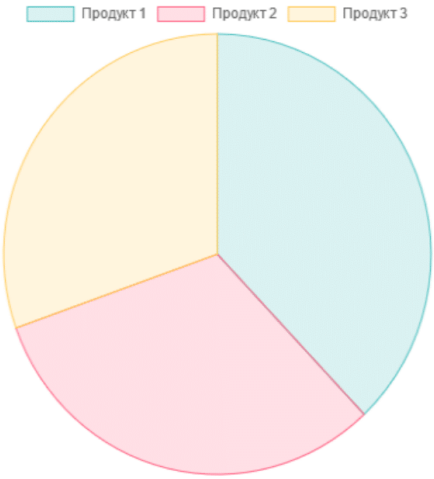


Рис. А.7. Вигляд згенерованого PDF-файлу для результатів методу Fuzzy MCDM

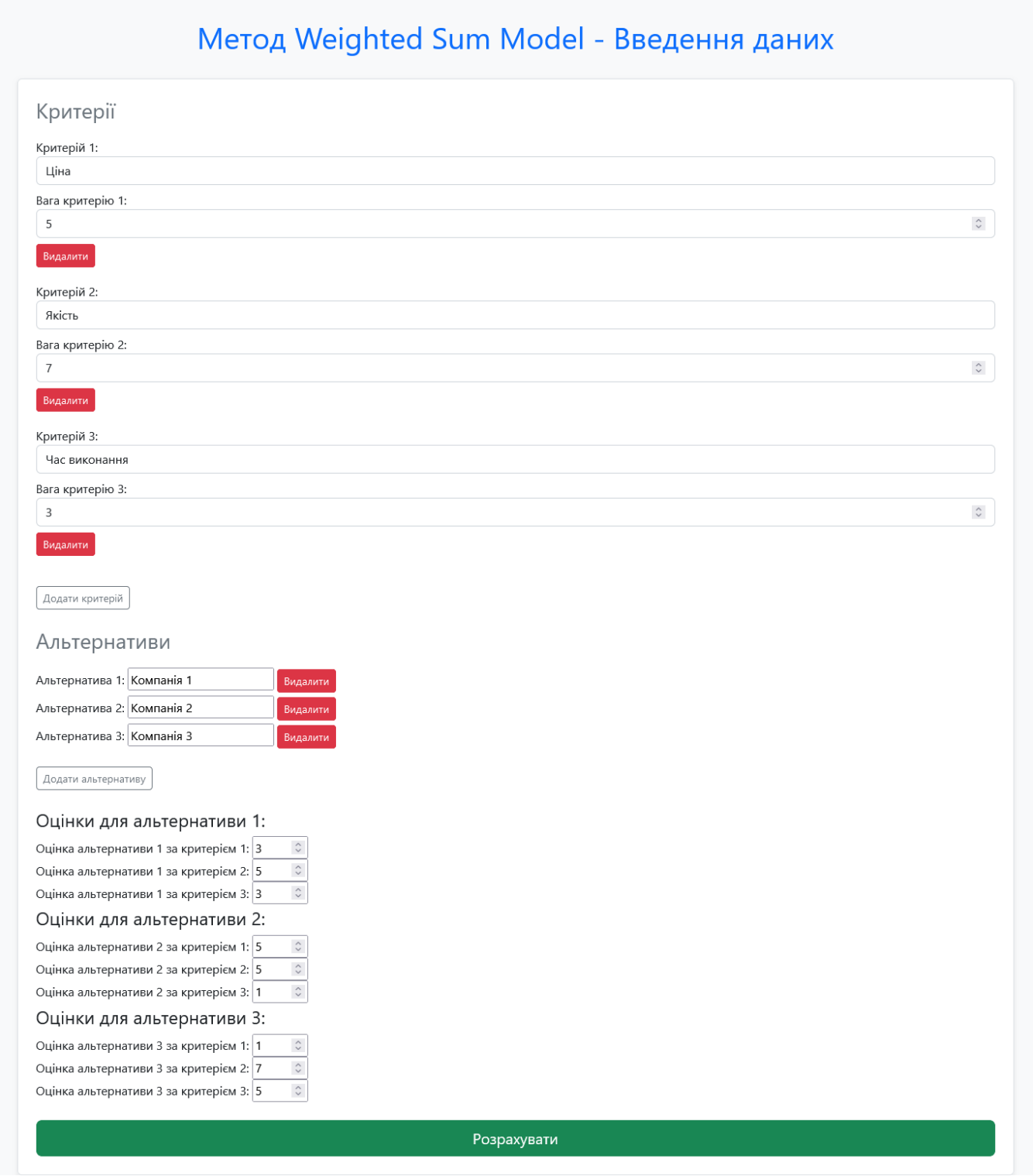


Рис. А.8. Сторінка «Метод Weighted Sum Model - Введення даних»

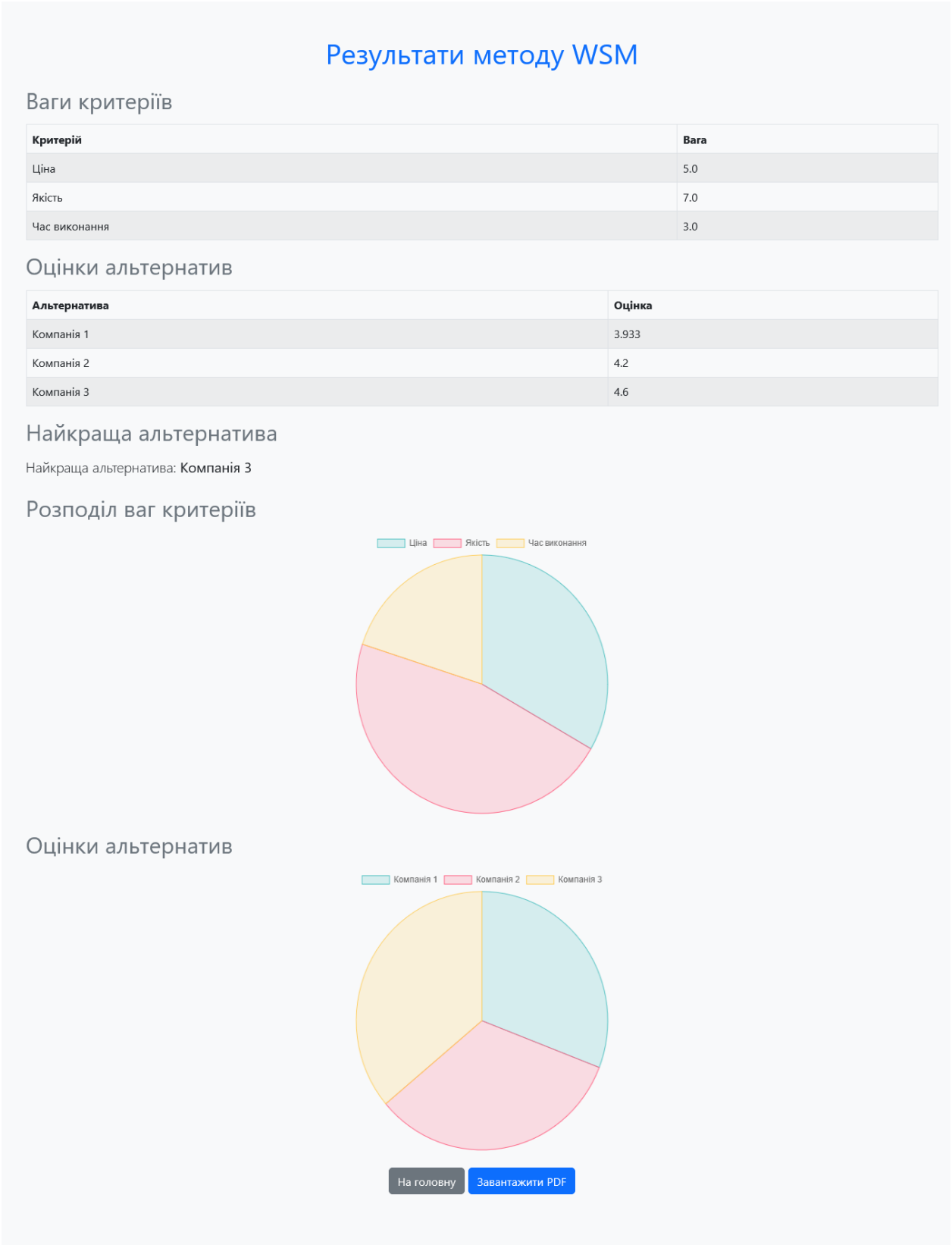


Рис. А.9. Сторінка результатів для методу WSM

Результати методу WSM

Ваги критеріїв

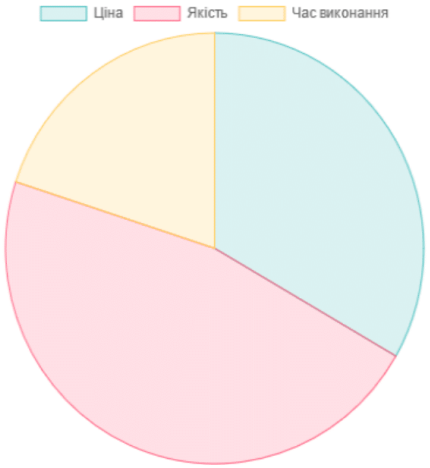
Критерій	Вага
Ціна	5.000
Якість	7.000
Час виконання	3.000

Оцінки альтернатив

Альтернатива	Оцінка
Компанія 1	3.933
Компанія 2	4.200
Компанія 3	4.600

Найкраща альтернатива: Компанія 3

Розподіл ваг критеріїв



Альтернативи

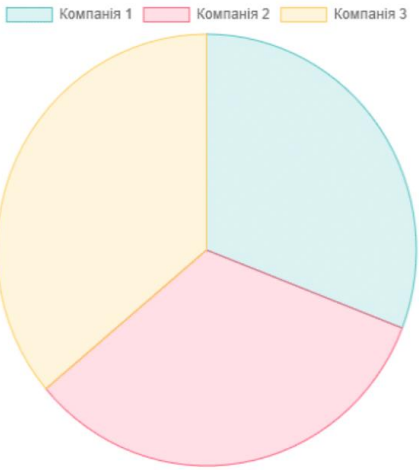


Рис. А.10. Вигляд згенерованого PDF-файлу для результатів методу WSM