КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій
(факультет)

інформаційних технологій
(кафедра)

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Аналіз та розробка інструментальних засобів прийняття рішень в небезпечних ситуаціях»

ГАРКАВЕНКО МАКСИМ ІГОРОВИЧ
(прізвище, ім’я та по батькові студента повністю)

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., доцент Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Аналіз та розробка інструментальних засобів прийняття рішень в небезпечних ситуаціях»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач Гаркавенко Максим Ігорович
(прізвище, ім’я та по батькові повністю)
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21-3
Керівник Горда О.В.
(прізвище та ініціали)
доцент, кандидат технічних наук
(вчене звання, науковий ступінь)
Рецензент доц. Азнаурян І. О.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Випускова кафедра: інформаційних технологій
Ступінь вищої освіти: «бакалавр»
Спеціальність: 122 «Комп`ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., доцент Гончаренко Т.А.
_____
„___” _____2025 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Гаркавенко Максим Ігорович

1. Тема роботи: Аналіз та розробка інструментальних засобів прийняття рішень в небезпечних ситуаціях
затверджена наказом ректора КНУБА №.
2. Керівник роботи: Горда Олена Володимирівна, к.т.н, доцент кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Постановка задачі та проєктування системи
Р.3. Програмна реалізація системи
Р.4. Оцінка ефективності та економічне обрунтування системи
5. Інформаційні слайди:
С.1.
С.2.
С.3.
С.4.
С.5.

6. Консультанти розділів кваліфікаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	доц. Рябчун Ю.В.		
Прийом програмного продукту	ас. Маційєвський О.О. ас. Довгополов С.Ю.		

7. Календарний план виконання кваліфікаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2025 р.
Р. 2. Алгоритмічне та математичне забезпечення	Лютий 2025 р.
Р. 3. Розробка програмного забезпечення	Березень 2025 р.
Тестовий приклад програми	Березень 2025 р.
Р. 4. Ергономіка інформаційних технологій	Квітень 2025 р.
Остаточне оформлення роботи	Травень 2025 р.
Направлення роботи на рецензування	Червень 2025 р.
Попередній захист роботи на кафедрі	Червень 2025 р.

8. Дата видачі завдання: 26.01.2025 р.

Завідувачка _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Керівник _____ Горда О.В.
(підпис) (прізвище та ініціали)

Здобувач _____ Гаркавенко М.І.
(підпис) (прізвище та ініціали)

РЕЗЮМЕ (SUMMARY) *до кваліфікаційної випускної роботи Здобувача:*	Гаркавенко Максим Ігорович Maxim Igorovich		
ЗВО	Київський національний університет будівництва і архітектури		
Тема (українською та англійською)	Аналіз та розробка інструментальних засобів прийняття рішень в небезпечних ситуаціях. Analysis and development of decision-making tools in hazardous situations.		
Освітній ступінь	Бакалавр		
Факультет	Автоматизації і інформаційних технологій		
Випускова кафедра	Інформаційних технологій		
Спеціальність	122 «Комп'ютерні науки»		
Освітня програма	Інформаційні управляючі системи та технології		
Керівник	Горда Олена Володимирівна		
Обсяг роботи:	пояснювальна записка, стор.	розділів	креслень формату А
	84	4	0
Розділ 1.	Аналіз предметної області та постановка задачі		
Розділ 2.	Постановка задачі та проєктування системи		
Розділ 3.	Програмна реалізація системи		
Розділ 4.	Оцінка ефективності та економічне обгрунтування		
Ключові слова: *Keywords:*	небезпечні ситуації, аналіз ризиків рішень (DSS), інструментальні засоби, автоматизація процесів, моделювання сценаріїв, моніторинг загроз, управління кризами, оцінка ризиків, прогнозування наслідків, безпека та захист. Hazardous situations, decision risk analysis, instrumental tools, process automation, scenario modeling, threat monitoring, crisis management, risk assessment, consequence forecasting, safety and security.		

Здобувач: _____ /Максим ГАРКАВЕНКО /

Керівник: _____/ Олена ГОРДА /

"___" _____ 2025р.

РЕЦЕНЗІЯ

на кваліфікаційну випускну роботу

Студента:	Гаркавенка Максима Ігоровича
Факультету:	Автоматизації і інформаційних технологій
Спеціальності:	122 «Комп'ютерні науки»
Освітня програма:	Інформаційні управляючі системи і технології
Тема роботи:	Аналіз та розробка інструментальних засобів прийняття рішень в небезпечних ситуаціях.
Обсяг роботи:	84 сторінок

_____

Висновок про відповідність завданню:
Актуальність обраної теми:
Використання у роботі сучасних досліджень науки і техніки:
Використання у роботі комп'ютерних технологій:
Практичне значення роботи:
Якість оформлення роботи:
Зауваження та побажання:
Загальний висновок стосовно роботи та надання авторові освітнього ступеня «бакалавр»:
Рекомендована оцінка:_____
Рецензент:
_____ / _____
(прізвище, ініціали) (підпис)
Посада, місце роботи:
_____«__»___
____20__р.

ЗМІСТ

ВСТУП

У сучасному світі людина постійно стикається з необхідністю прийняття складних рішень в умовах невизначеності, обмеженого часу та множини суперечливих критеріїв. Особливо критичним це стає у небезпечних ситуаціях, де неправильне або несвоєчасне рішення може призвести до серйозних наслідків для життя та здоров'я людей, матеріальних втрат або екологічних катастроф. До таких ситуацій належать техногенні аварії, природні катастрофи, медичні ургентні стани, кіберзагрози та інші критичні події.

Системи підтримки прийняття рішень є потужним інструментом для підвищення якості та обґрунтованості управлінських рішень. Вони дозволяють структурувати складні проблеми, систематизувати інформацію, використовувати математичні методи для аналізу альтернатив та надавати особам, що приймають рішення, об'єктивні рекомендації щодо оптимального вибору.

Серед різноманітних методів багатокритеріального аналізу рішень особливе місце займає метод аналізу ієрархій, розроблений Томасом Сааті. Цей метод поєднує математичну строгість з інтуїтивною зрозумілістю, дозволяє враховувати як кількісні, так і якісні критерії, має механізм перевірки узгодженості експертних оцінок та успішно застосовується у різних галузях для вирішення складних задач прийняття рішень.

Актуальність теми дослідження. Актуальність розробки інструментальних засобів прийняття рішень у небезпечних ситуаціях обумовлена декількома факторами. По-перше, зростає кількість та складність техногенних та природних загроз, що вимагає від осіб, які приймають рішення, здатності швидко аналізувати ситуацію та обирати оптимальні варіанти дій. По-друге, сучасні комерційні системи підтримки прийняття рішень є дорогими та часто надмірно складними для практичного застосування в умовах обмеженого часу. По-третє, існуючі безкоштовні рішення мають застарілий інтерфейс або вимагають спеціальних технічних знань для використання.

Таким чином, існує потреба у створенні доступної, зручної та ефективної системи підтримки прийняття рішень, яка б поєднувала математичну коректність методу аналізу ієрархій з інтуїтивним інтерфейсом та швидкістю роботи, необхідною для застосування у критичних ситуаціях.

Метою дипломної роботи є розробка інструментального засобу підтримки прийняття рішень у небезпечних ситуаціях на основі методу аналізу ієрархій, що забезпечує швидке та обґрунтоване прийняття рішень в умовах множинних критеріїв та обмеженого часу.

Для досягнення поставленої мети необхідно вирішити наступні задачі. Провести аналіз предметної області, дослідити характеристики небезпечних ситуацій та вимоги до систем підтримки прийняття рішень. Виконати огляд методів багатокритеріального аналізу та обґрунтувати вибір методу аналізу ієрархій. Проаналізувати існуючі програмні реалізації систем прийняття рішень та виявити їх недоліки. Сформулювати задачу прийняття рішень та розробити функціональні вимоги до системи. Спроектувати архітектуру системи та розробити UML-діаграми. Обґрунтувати вибір технологій реалізації системи. Реалізувати програмне забезпечення системи з усіма необхідними модулями. Імплементувати алгоритми методу аналізу ієрархій, включаючи обчислення пріоритетів та перевірку узгодженості. Провести тестування системи на практичних прикладах. Виконати оцінку ефективності системи та економічне обґрунтування розробки.

Об'єктом дослідження є процес прийняття багатокритеріальних рішень у небезпечних ситуаціях в умовах невизначеності та обмеженого часу.

Предметом дослідження є методи та інструментальні засоби підтримки прийняття рішень на основі методу аналізу ієрархій.

Методи дослідження. У роботі використовуються наступні методи дослідження. Метод аналізу ієрархій для багатокритеріального прийняття рішень. Методи системного аналізу для дослідження предметної області. Методи проєктування інформаційних

систем, включаючи UML-моделювання. Методи об'єктно-орієнтованого програмування для реалізації системи. Методи тестування програмного забезпечення для перевірки коректності роботи. Методи економічного аналізу для обґрунтування ефективності розробки.

Наукова новизна роботи полягає у розробці спеціалізованого підходу до створення системи підтримки прийняття рішень, орієнтованої на застосування у небезпечних ситуаціях. Запропоновано архітектуру системи, що оптимізує баланс між функціональністю та швидкістю роботи. Розроблено методику інтеграції перевірки узгодженості експертних оцінок безпосередньо у процес введення даних для підвищення якості результатів.

Практична цінність роботи полягає у створенні готового програмного продукту, який може бути використаний організаціями та особами для прийняття обґрунтованих рішень у критичних ситуаціях. Система забезпечує значне скорочення часу на прийняття рішень порівняно з ручними обчисленнями. Розроблене програмне забезпечення може застосовуватися у різних галузях, включаючи управління надзвичайними ситуаціями, охорону здоров'я, промислову безпеку, фінансовий ризик-менеджмент та інші сфери, де необхідно приймати складні рішення в умовах множинних критеріїв.

Структура роботи. Дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У вступі обґрунтовується актуальність теми, формулюються мета та задачі дослідження, визначаються об'єкт і предмет дослідження, наукова новизна та практична цінність роботи.

У першому розділі проводиться аналіз предметної області, розглядаються характеристики небезпечних ситуацій, досліджуються методи багатокритеріального аналізу та аналізуються існуючі програмні реалізації систем прийняття рішень.

У другому розділі формулюється задача прийняття рішень, розробляються функціональні вимоги до системи, проєктується архітектура, створюються UML-діаграми та обґрунтовується вибір технологій реалізації.

У третьому розділі описується програмна реалізація системи, детально розглядаються розроблені модулі, алгоритмічне забезпечення методу аналізу ієрархій, наводяться результати тестування на практичних прикладах.

У четвертому розділі проводиться оцінка ефективності використання системи, порівняння з аналогами, економічне обґрунтування розробки та аналіз можливостей подальшого вдосконалення.

У висновках узагальнюються результати виконаної роботи, формулюються основні наукові та практичні результати дослідження.

Загальний обсяг роботи становить вісімдесят сторінок, включаючи таблиці та ілюстрації. Список використаних джерел містить тридцять найменувань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз небезпечних ситуацій та існуючих систем підтримки прийняття рішень

У сучасному світі людина постійно стикається з необхідністю прийняття рішень в умовах невизначеності та обмеженого часу. Особливо критичним це стає у небезпечних ситуаціях, де неправильне рішення може призвести до серйозних наслідків для життя та здоров'я людей, матеріальних втрат або екологічних катастроф. Небезпечна ситуація характеризується наявністю загрози для безпеки об'єкта захисту та обмеженим часом на прийняття рішення. До таких ситуацій можна віднести техногенні аварії на виробництві та транспорті, природні катастрофи та надзвичайні ситуації, медичні ургентні стани, кіберзагрози та інформаційні атаки, військові конфлікти та загрози безпеці, а також фінансові кризи та ризики інвестицій [1-2].

Системи підтримки прийняття рішень (СППР) представляють собою комп'ютеризовані інформаційні системи, які допомагають особам, що приймають рішення, у виборі найкращого варіанту дій з множини альтернатив на основі аналізу даних та застосування математичних методів [3-4].

Основні компоненти типової СППР включають базу даних для зберігання інформації про проблемну область, базу моделей з набором математичних та аналітичних моделей, систему управління базою даних, систему управління базою моделей, модуль інтелектуального аналізу даних, а також інтерфейс користувача для взаємодії з системою. Сучасні СППР використовують різноманітні технології та методи, включаючи штучний інтелект, машинне навчання, експертні системи, нейронні мережі та методи багатокритеріального аналізу. Серед найбільш поширених методів багатокритеріального аналізу виділяють метод аналізу ієрархій (MAI), метод ELECTRE, метод TOPSIS та інші [5-6].

1.2. Огляд методів багатокритеріального аналізу

Багатокритеріальний аналіз рішень є важливою складовою процесу прийняття рішень в умовах, коли необхідно враховувати декілька, часто суперечливих, критеріїв. Розглянемо основні методи багатокритеріального аналізу, що застосовуються у системах підтримки прийняття рішень.

Метод аналізу ієрархій (AHP/AHP). Метод аналізу ієрархій, розроблений Томасом Сааті у 1970-х роках, є одним з найпопулярніших методів багатокритеріального прийняття рішень. Суть методу полягає у декомпозиції складної проблеми на ієрархію простіших підпроблем та подальшому синтезі рішень. Основні етапи застосування MAI включають побудову ієрархії проблеми з трьох рівнів (мета, критерії та альтернативи), формування матриць парних порівнянь для кожного рівня ієрархії, обчислення векторів пріоритетів методом власних векторів, перевірку узгодженості суджень через розрахунок індексу узгодженості (CI) та відношення узгодженості (CR), а також синтез глобальних пріоритетів альтернатив [7, с. 6-18].

Для парних порівнянь використовується шкала Сааті від 1 до 9, де 1 означає рівну важливість, а 9 — абсолютну перевагу одного елемента над іншим. Коефіцієнт узгодженості CR повинен бути менше 0,1 (10%) для прийнятності результатів. Детальний опис шкали Сааті наведено у таблиці 1.1.

Таблиця 1.1

Шкала відносної важливості Сааті

Оцінка	**Визначення**	**Пояснення**
1	Рівна важливість	Два елементи вносять однаковий вклад у мету

3	Помірна перевага	Досвід і судження дають легку перевагу одному елементу
5	Сильна перевага	Досвід і судження дають сильну перевагу одному елементу
7	Дуже сильна перевага	Перевага одного елемента дуже сильна
9	Абсолютна перевага	Очевидність переваги підтверджується найбільш сильно
2, 4, 6, 8	Проміжні значення	Використовуються у компромісних випадках

Використання цієї шкали дозволяє експертам виражати свої судження у зрозумілій формі, при цьому проміжні значення (2, 4, 6, 8) можуть застосовуватися у випадках, коли необхідна більша точність оцінювання. Переваги MAI включають простоту використання, можливість врахування як кількісних, так і якісних критеріїв, а також наявність механізму перевірки узгодженості. Недоліками є суб'єктивність оцінок та збільшення кількості порівнянь при зростанні числа елементів.

Метод ELECTRE (Elimination and Choice Translating Reality) базується на концепції відношень переважання. Він використовує індекси узгодженості та неузгодженості для порівняння альтернатив та виключення менш привабливих варіантів. Основні переваги методу полягають у можливості роботи з неповною інформацією та врахуванні порогів переваги. Однак метод є складнішим у реалізації порівняно з MAI [8-9].

Метод TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) ранжує альтернативи на основі їх близькості до ідеального рішення та віддаленості від найгіршого. Метод використовує евклідову відстань у багатовимірному просторі критеріїв. Переваги TOPSIS включають логічну обґрунтованість та відносну простоту реалізації. Метод добре підходить для задач з чітко визначеними числовими критеріями.

Інші методи. Серед інших методів багатокритеріального аналізу варто відзначити PROMETHEE (метод, що використовує попарні порівняння з функціями переваги), VIKOR (компромісне ранжування з використанням міри близькості до ідеального рішення), DEA або Data Envelopment Analysis (метод аналізу оболонки даних для оцінки ефективності), а також метод зваженої суми як найпростіший підхід з лінійною згорткою критеріїв [10-12].

Вибір конкретного методу залежить від характеру задачі, типу даних, вимог до точності та можливостей особи, що приймає рішення. У контексті небезпечних ситуацій найбільш прийнятним є метод аналізу ієрархій завдяки його гнучкості та можливості швидкого опрацювання експертних оцінок. Для наочного порівняння розглянутих методів наведемо їх основні характеристики у таблиці 1.2.

Таблиця 1.2

Порівняльний аналіз методів багатокритеріального аналізу

Метод	**Переваги**	**Недоліки**	**Складність реалізації**
Метод аналізу ієрархій (AHP)	Простота, перевірка узгодженості, якісні та кількісні критерії	Суб'єктивність оцінок, багато порівнянь	Середня
ELECTRE	Робота з неповною інформацією, пороги переваги	Складність, важко інтерпретувати	Висока
TOPSIS	Логічна обґрунтованість, простота	Потребує числових критеріїв	Низька
PROMETHEE	Гнучкі функції переваги, наочність	Потребує визначення параметрів	Середня

Метод зваженої суми	Найпростіший, швидкий	Лінійна залежність, обмежена точність	Дуже низька

Як видно з таблиці 1.2, метод аналізу ієрархій має оптимальне співвідношення між складністю реалізації та функціональними можливостями. Незважаючи на певну суб'єктивність експертних оцінок, наявність механізму перевірки узгодженості дозволяє контролювати якість результатів.

1.3. Аналіз існуючих програмних реалізацій систем прийняття рішень

На ринку програмного забезпечення існує ряд реалізацій систем підтримки прийняття рішень, кожна з яких має свої особливості, переваги та недоліки. Розглянемо найбільш поширені рішення.

Expert Choice є найбільш відомою комерційною реалізацією методу аналізу ієрархій. Система забезпечує повний цикл роботи з MAI: від побудови ієрархії до отримання результатів та звітів. Переваги: професійний інтерфейс, розширені можливості візуалізації, підтримка колективного прийняття рішень, експорт результатів у різні формати. Недоліки: висока вартість ліцензії, закритий код, обмежені можливості кастомізації, відсутність інтеграції з іншими методами MCDM [13-14].

SuperDecisions — безкоштовна програма для роботи з методом аналізу ієрархій та методом аналізу мереж (ANP). Розроблена Creative Decisions Foundation. Переваги: безкоштовна, підтримка ANP для складних взаємозв'язків між критеріями, активна спільнота користувачів. Недоліки: застарілий інтерфейс, складність освоєння для початківців, обмежені можливості інтеграції з іншими системами.

DecisionLens — хмарна платформа для прийняття стратегічних рішень на основі MAI. Орієнтована на корпоративний сегмент. Переваги: хмарна архітектура, підтримка колаборації, інтеграція з корпоративними системами, мобільні додатки. Недоліки: висока вартість, необхідність постійного інтернет-з'єднання, обмеження для невеликих проєктів.

Відкриті бібліотеки та інструменти. Існують також відкриті реалізації MAI у вигляді бібліотек для різних мов програмування. Серед них можна відзначити pyAHP (Python) як просту бібліотеку для базових операцій MAI, AHP-OS (JavaScript) як веб-орієнтоване рішення з відкритим кодом, ahpy (Python) як сучасну реалізацію з підтримкою групових рішень, а також R-пакети (ahp, AHP) як інструменти для статистичного середовища R. Переваги: безкоштовні, відкритий код, можливість інтеграції у власні проєкти. Недоліки: відсутність готового GUI, обмежений функціонал, необхідність програмування [15].

Аналіз існуючих рішень показує, що комерційні системи пропонують професійний інтерфейс та розширені можливості, але є дорогими та негнучкими. Відкриті бібліотеки забезпечують свободу реалізації, але вимагають значних зусиль на розробку інтерфейсу.

Жодне з розглянутих рішень не є оптимальним для швидкого прийняття рішень у небезпечних ситуаціях. Існує потреба у створенні спеціалізованої системи, яка поєднує простоту використання, доступність та достатній функціонал для ефективної роботи з методом аналізу ієрархій [16].

1.4. Висновки до розділу 1

У першому розділі було проведено аналіз предметної області системи підтримки прийняття рішень у небезпечних ситуаціях. В ході дослідження визначено характеристики небезпечних ситуацій та вимоги до систем прийняття рішень в умовах обмеженого часу та невизначеності. Проведено огляд основних методів

багатокритеріального аналізу, серед яких метод аналізу ієрархій визначено як найбільш придатний для застосування у критичних ситуаціях завдяки балансу між точністю та складністю реалізації. Проаналізовано існуючі програмні реалізації СППР та виявлено їх основні недоліки, зокрема високу вартість комерційних рішень, складність у використанні та відсутність спеціалізації для роботи у критичних ситуаціях. На основі проведеного аналізу обґрунтовано необхідність розробки нової системи підтримки прийняття рішень, яка поєднує доступність, простоту використання та достатній функціонал для ефективної роботи з MAI.

Результати аналізу предметної області створюють теоретичне підґрунтя для постановки задачі та проєктування системи, що буде розглянуто у наступних розділах роботи.

РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Формулювання задачі прийняття рішень у контексті небезпечних ситуацій

На основі проведеного в першому розділі аналізу предметної області можна сформулювати задачу розробки системи підтримки прийняття рішень у небезпечних ситуаціях. Основна мета полягає у створенні інструментального засобу, що дозволить особам, які приймають рішення, швидко та обґрунтовано обирати оптимальний варіант дій з множини альтернатив на основі багатокритеріального аналізу.

Математична постановка задачі полягає у наступному. Нехай дано множину альтернатив $A = \{A_1, A_2, ..., A_n\}$, що представляють можливі варіанти дій у небезпечній ситуації, та множину критеріїв $C = \{C_1, C_2, ..., C_m\}$, за якими оцінюються ці альтернативи. Необхідно визначити пріоритети альтернатив та обрати найкращий варіант або ранжувати всі альтернативи за ступенем привабливості. Для вирішення цієї задачі використовується метод аналізу ієрархій, який передбачає побудову ієрархічної моделі з трьох рівнів. Перший рівень представляє мету прийняття рішення. Другий рівень містить критерії оцінки альтернатив. Третій рівень включає альтернативи, що розглядаються. Процес прийняття рішення включає парні порівняння елементів на кожному рівні, обчислення локальних пріоритетів та синтез глобальних пріоритетів альтернатив [17, с. 129-138].

Особливості задачі прийняття рішень у небезпечних ситуаціях накладають специфічні вимоги до системи. По-перше, система повинна забезпечувати швидке введення даних та отримання результатів, оскільки час на прийняття рішення обмежений. По-друге, інтерфейс має бути інтуїтивно зрозумілим та не вимагати спеціальної підготовки користувача. По-третє, система повинна автоматично перевіряти узгодженість експертних оцінок та повідомляти про необхідність їх корекції.

Додатковою вимогою є можливість збереження історії прийнятих рішень для подальшого аналізу та навчання. Це дозволить накопичувати досвід роботи з різними типами небезпечних ситуацій та використовувати його для підвищення якості майбутніх рішень.

Додатковою вимогою до системи є можливість роботи в режимі реального часу або близькому до нього. Для типових задач час обчислення не повинен перевищувати декількох секунд. Для забезпечення гнучкості системи необхідно передбачити можливість роботи з різними типами критеріїв оцінювання. Критерії можуть бути як кількісними, що вимірюються у конкретних одиницях, так і якісними, що базуються на експертних судженнях.

Специфіка небезпечних ситуацій також вимагає врахування психологічних факторів впливу на процес прийняття рішень. В умовах стресу та обмеженого часу люди схильні до когнітивних спотворень та можуть приймати нераціональні рішення під впливом емоцій. Метод аналізу ієрархій допомагає декомпозувати складну проблему на простіші компоненти [18, с. 401-406].

Важливою складовою постановки задачі є визначення обмежень та припущень, які застосовуються при моделюванні небезпечної ситуації. У реальних умовах особа, що приймає рішення, може не мати повної інформації про всі аспекти ситуації, тому система повинна бути здатною працювати з неповними даними та надавати користувачу рекомендації навіть за наявності невизначеності.

2.2. Розроблення функціональних вимог до системи

На основі поставленої задачі та аналізу предметної області сформульовано функціональні вимоги до розроблюваної системи. Ці вимоги визначають, які функції має виконувати система для досягнення поставленої мети. Повний перелік функціональних вимог до системи наведено у таблиці 2.1.

Таблиця 2.1

Функціональні вимоги до системи

№	Функція	Опис	Пріоритет
1	Реєстрація користувачів	Створення облікового запису з унікальним логіном	Високий
2	Авторизація	Вхід у систему з перевіркою облікових даних	Високий
3	Створення моделі	Додавання нової задачі прийняття рішень	Високий
4	Додавання критеріїв	Визначення критеріїв оцінювання (мінімум 2)	Високий
5	Додавання альтернатив	Визначення варіантів рішень (мінімум 2)	Високий
6	Парні порівняння	Введення оцінок за шкалою Сааті (1-9)	Високий
7	Обчислення пріоритетів	Автоматичний розрахунок локальних та глобальних ваг	Високий
8	Перевірка узгодженості	Контроль CR ≤ 0.1 з повідомленням користувачу	Високий
9	Візуалізація результатів	Таблиці та діаграми пріоритетів	Середній
10	Експорт результатів	Збереження у PDF, Excel, CSV	Середній
11	Історія моделей	Перегляд раніше створених рішень	Низький

Як видно з таблиці 2.1, функціональні вимоги поділено за пріоритетами. Високий пріоритет має функціонал, необхідний для базової роботи системи з методом аналізу ієрархій. Середній пріоритет присвоєно функціям візуалізації та експорту, які підвищують зручність використання. Низький пріоритет має додатковий функціонал для роботи з історією моделей.

Система повинна забезпечувати реєстрацію нових користувачів з введенням унікального імені користувача та пароля. Функція авторизації має перевіряти облікові дані та надавати доступ до персоналізованого робочого середовища. Кожен користувач матиме власну історію створених моделей прийняття рішень, що дозволить відокремити дані різних користувачів та забезпечити конфіденційність інформації [19-20].

Система має надавати інтерфейс для створення нової моделі з можливістю введення назви та опису задачі. Користувач повинен мати змогу додавати довільну кількість критеріїв оцінювання з можливістю вказати назву та опис кожного критерію. Аналогічно, система має дозволяти додавати альтернативи, які будуть оцінюватися за вказаними критеріями. Інтерфейс має бути інтуїтивним та зручним для швидкого введення інформації.

Центральною функцією системи є можливість проведення парних порівнянь елементів моделі. Система має генерувати матриці парних порівнянь для критеріїв відносно мети та для альтернатив відносно кожного критерію. Інтерфейс введення порівнянь має відображати шкалу Сааті від 1 до 9 з поясненням значень. Користувач обирає ступінь переваги одного елемента над іншим, а система автоматично заповнює симетричні комірки матриці оберненими значеннями [21, с. 144].

Після введення всіх парних порівнянь система має виконувати обчислення локальних пріоритетів методом власного вектора. Для кожної матриці порівнянь виконується нормалізація стовпців шляхом ділення кожного елемента на суму відповідного стовпця. Локальні пріоритети обчислюються як середнє арифметичне значень рядків нормалізованої матриці. Система має також обчислювати максимальне

власне значення матриці, індекс узгодженості та коефіцієнт узгодженості для перевірки коректності експертних оцінок.

Коефіцієнт узгодженості обчислюється за формулою

$$CR = \frac{CI}{RI}, \quad (1)$$

де CI — індекс узгодженості, а RI — випадковий індекс для відповідної розмірності матриці. Якщо коефіцієнт узгодженості перевищує значення 0,1, система має повідомити користувача про недостатню узгодженість суджень та необхідність перегляду оцінок. Після обчислення локальних пріоритетів критеріїв та альтернатив відносно кожного критерію система виконує синтез глобальних пріоритетів. Глобальний пріоритет кожної альтернативи обчислюється як зважена сума її локальних пріоритетів, де вагами виступають пріоритети відповідних критеріїв.

Результати обчислень мають бути представлені користувачу у зручній та наочній формі. Система повинна відображати таблиці вагових коефіцієнтів критеріїв та локальних пріоритетів альтернатив. Підсумковий рейтинг альтернатив має бути представлений у вигляді упорядкованого списку з глобальними пріоритетами. Для кращого сприйняття інформації система має генерувати графічні діаграми, зокрема стовпчасті діаграми для порівняння глобальних пріоритетів альтернатив та кругові діаграми для відображення ваг критеріїв.

Система має забезпечувати експорт результатів у популярні формати для подальшого використання. Підтримуються формати PDF для створення звітів, Excel для роботи з таблицями та CSV для обміну даними з іншими програмами. Всі створені моделі та результати обчислень зберігаються в базі даних користувача, що дозволяє повертатися до них пізніше та аналізувати історію прийнятих рішень.

Система має підтримувати різні режими роботи залежно від рівня досвіду користувача. Для початківців корисним є покроковий режим з детальними підказками.

Функціональні вимоги також включають необхідність ведення журналу операцій користувача для можливості аудиту прийнятих рішень. Кожна операція має фіксуватися з відміткою часу. Система повинна забезпечувати механізми валідації введених даних на всіх етапах роботи користувача. При створенні моделі необхідно перевіряти унікальність назв критеріїв та альтернатив [22, с. 131-139].

Важливим аспектом функціональних вимог є забезпечення можливості роботи з ієрархічними структурами критеріїв різної глибини. У деяких задачах може бути доцільним групування критеріїв у категорії вищого рівня.

2.3. Проєктування архітектури системи

Архітектура розроблюваної системи базується на класичній трирівневій моделі, яка забезпечує розділення відповідальності між різними компонентами та полегшує подальший розвиток та супровід системи.

Рівень представлення реалізований у вигляді Windows Forms додатку, що забезпечує графічний інтерфейс користувача. Цей рівень відповідає за взаємодію з користувачем, відображення форм для введення даних та представлення результатів обчислень. Основні форми системи включають форму авторизації та реєстрації, головну форму з переліком моделей користувача, форму створення та редагування моделі прийняття рішень, форми для введення парних порівнянь критеріїв та альтернатив, а також форму відображення результатів з таблицями та діаграмами. Інтерфейс проектується з урахуванням принципів юзабіліті та ергономіки. Елементи управління розміщуються логічно та послідовно відповідно до робочого процесу користувача. Використовуються стандартні елементи Windows Forms, що забезпечує знайомість інтерфейсу для користувачів операційної системи Windows [23, с. 83-89].

Рівень бізнес-логіки містить основні алгоритми системи та реалізує метод аналізу ієрархій. Цей рівень представлений набором класів, які виконують обчислювальні

операції. Клас для роботи з матрицями парних порівнянь забезпечує створення матриць, введення значень та нормалізацію. Клас обчислення пріоритетів реалізує метод власного вектора для визначення локальних пріоритетів елементів. Клас перевірки узгодженості обчислює індекс та коефіцієнт узгодженості матриць порівнянь. Клас синтезу глобальних пріоритетів виконує підсумкове обчислення рейтингу альтернатив. Алгоритми рівня бізнес-логіки реалізовані мовою C# з використанням об'єктно-орієнтованих принципів. Це забезпечує модульність коду, можливість повторного використання компонентів та легкість тестування окремих модулів.

Рівень даних відповідає за збереження та отримання інформації з бази даних. Для зберігання даних використовується Microsoft Access Database через його простоту у розгортанні та достатню функціональність для задач системи [24, с. 104-106]. База даних містить таблиці для зберігання інформації про користувачів з полями для імені користувача та хешу пароля, моделей прийняття рішень з назвою, описом та датою створення, критеріїв з назвою та описом, альтернатив з назвою та описом, матриць парних порівнянь критеріїв, матриць парних порівнянь альтернатив для кожного критерію, а також результатів обчислень з пріоритетами та коефіцієнтами узгодженості (рис.2.1).

```
// Поля класу
private string dbPath = @"D:\Фриланс\28936\DecisionMatrixDB\DecisionMatrixDB\DecisionSupportDB.accdb";
private string connectionString;
```

Рис. 2.1 – Рядок підключення до бази даних Microsoft Access у класі DatabaseManager

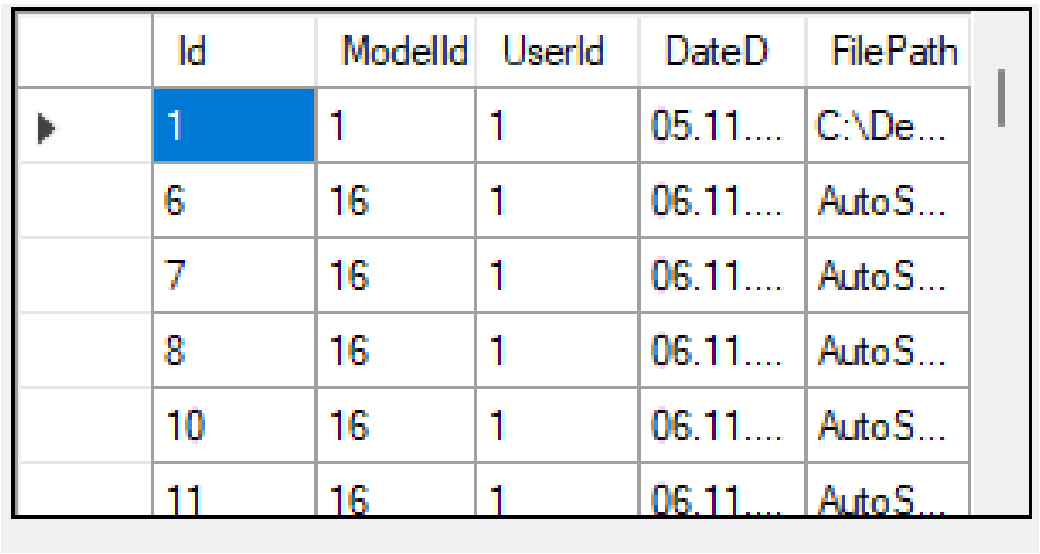
Взаємодія з базою даних здійснюється через ADO.NET з використанням OleDbConnection та відповідних класів для виконання запитів. Реалізовані методи для створення, читання, оновлення та видалення записів, що забезпечує повний цикл роботи з даними. Структура бази даних системи узагальнена у таблиці 2.2.

Таблиця 2.2

Структура бази даних системи

Таблиця	Призначення	Основні поля
Users	Облікові записи користувачів	UserId, Login, PasswordHash, Name, Surname
Models	Моделі прийняття рішень	ModelId, UserId, ModelName, Description, DateCreated
Criteria	Критерії оцінювання	CriterionId, ModelId, Name, Weight
Alternatives	Альтернативи рішень	AlternativeId, ModelId, Name, Weight
PairwiseComparisons	Парні порівняння	Id, ModelId, CriterionId, Alternative1, Alternative2, ValueR
Results	Результати обчислень	ResultId, ModelId, UserId, DateCalculated, FilePath

Як видно з таблиці, структура бази даних реалізує повну підтримку методу аналізу ієрархій, забезпечуючи зберігання всіх необхідних елементів моделі прийняття рішень та результатів обчислень (рис.2.2).



	Id	ModelId	UserId	DateD	FilePath
▶	1	1	1	05.11....	C:\De...
	6	16	1	06.11....	AutoS...
	7	16	1	06.11....	AutoS...
	8	16	1	06.11....	AutoS...
	10	16	1	06.11....	AutoS...
	11	16	1	06.11....	AutoS...

Рис. 2.2 - Вміст таблиці Results з історією збережених розрахунків

Взаємодія між рівнями архітектури організована за принципом зверху вниз. Рівень представлення ініціює дії користувача та викликає методи рівня бізнес-логіки для виконання обчислень. Рівень бізнес-логіки використовує рівень даних для збереження та отримання інформації. Таке розділення забезпечує низьку зв'язаність компонентів та високу cohesion всередині кожного рівня, що є важливими принципами програмної інженерії.

Важливим аспектом є забезпечення розширюваності системи. Інтерфейси використовуються для визначення контрактів між компонентами.

Рівень бізнес-логіки реалізує принцип єдиної відповідальності, де кожен клас має чітко визначену задачу. Це спрощує тестування окремих компонентів. Для забезпечення цілісності даних використовуються механізми транзакцій та каскадного видалення. При видаленні моделі автоматично видаляються всі пов'язані дані.

Архітектура системи передбачає використання патернів проєктування для забезпечення масштабованості та підтримуваності коду. Застосовується патерн Repository для абстрагування роботи з базою даних [25-26].

Важливим аспектом є забезпечення розширюваності системи. Інтерфейси використовуються для визначення контрактів між компонентами.

Рівень бізнес-логіки реалізує принцип єдиної відповідальності, де кожен клас має чітко визначену задачу. Це спрощує тестування окремих компонентів. Для забезпечення цілісності даних використовуються механізми транзакцій та каскадного видалення. При видаленні моделі автоматично видаляються всі пов'язані дані.

Архітектура системи передбачає використання патернів проєктування для забезпечення масштабованості та підтримуваності коду. Застосовується патерн Repository для абстрагування роботи з базою даних [27].

2.4. Розробка UML-діаграм

Для формалізації проєктних рішень та документування структури системи використовуються діаграми мови UML (Unified Modeling Language). Ці діаграми забезпечують чітке розуміння архітектури системи та взаємодії її компонентів.

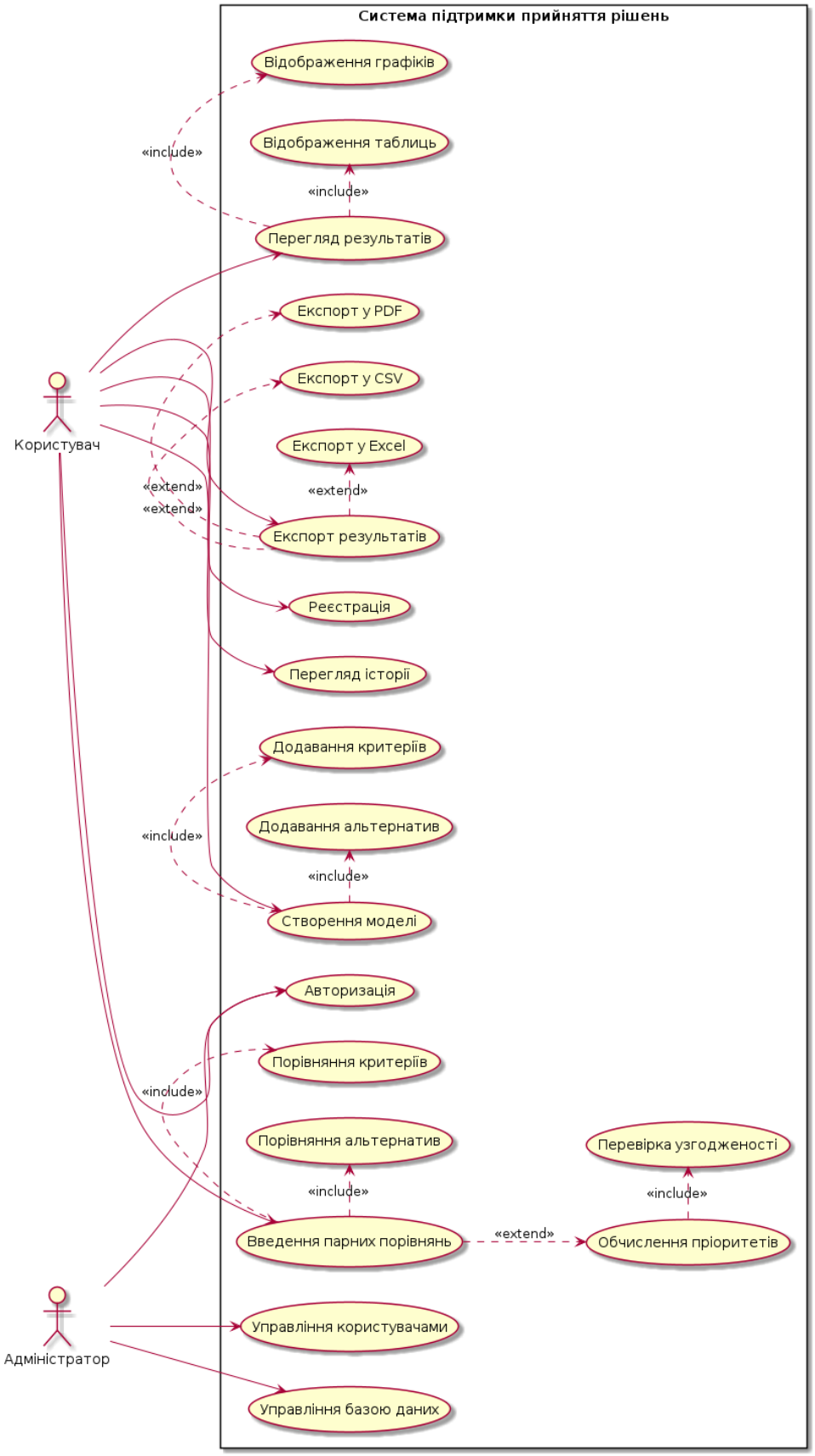

Система підтримки прийняття рішень
Відображення графіків
Відображення таблиць
Перегляд результатів
Експорт у PDF
Експорт у CSV
Експорт у Excel
Експорт результатів
Реєстрація
Перегляд історії
Додавання критеріїв
Додавання альтернатив
Створення моделі
Авторизація
Порівняння критеріїв
Порівняння альтернатив
Введення парних порівнянь
Перевірка узгодженості
Обчислення пріоритетів
Управління користувачами
Управління базою даних
Користувач
Адміністратор
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«include»
«extend»
«extend»
«extend»
«extend»

Рис. 2.3 — Діаграма прецедентів системи підтримки прийняття рішень

Діаграма прецедентів відображає функціональність системи з точки зору користувача. Основний актор системи — Користувач, який може виконувати наступні прецеденти використання. Прецедент 'Реєстрація' дозволяє створити новий обліковий запис у системі. Прецедент 'Авторизація' забезпечує вхід у систему з перевіркою облікових даних. Прецедент 'Створення моделі' включає підпрецеденти додавання критеріїв та альтернатив. Прецедент 'Введення парних порівнянь' передбачає порівняння критеріїв між собою та альтернатив за кожним критерієм (рис.2.4-2.5).

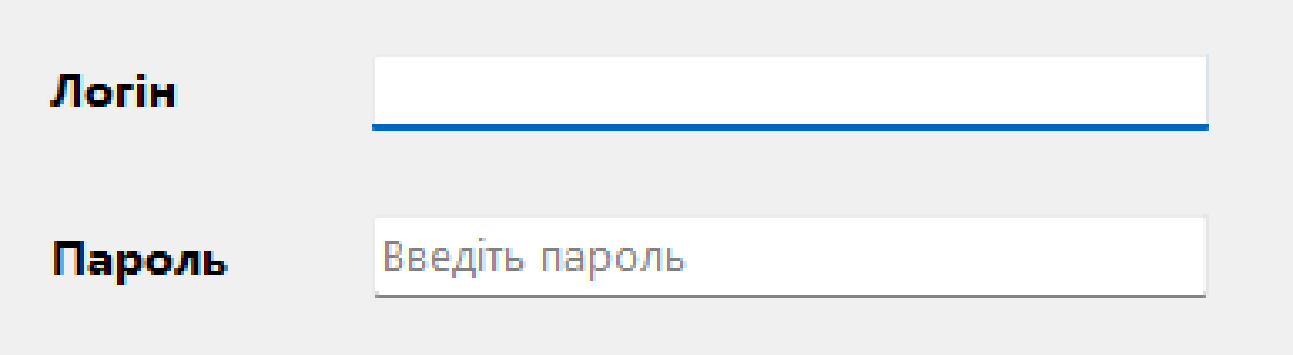


Рис. 2.4 – Зовнішній вигляд форми автентифікації з полями введення логіна та пароля

Рис. 2.5 – Кнопки «Вхід» та «Реєстрація» на формі автентифікації користувача

Прецедент 'Перегляд результатів' включає відображення таблиць пріоритетів та графічних діаграм. Прецедент 'Експорт результатів' дозволяє зберегти результати у форматах PDF, Excel або CSV. Прецедент 'Перегляд історії' надає доступ до раніше створених моделей.

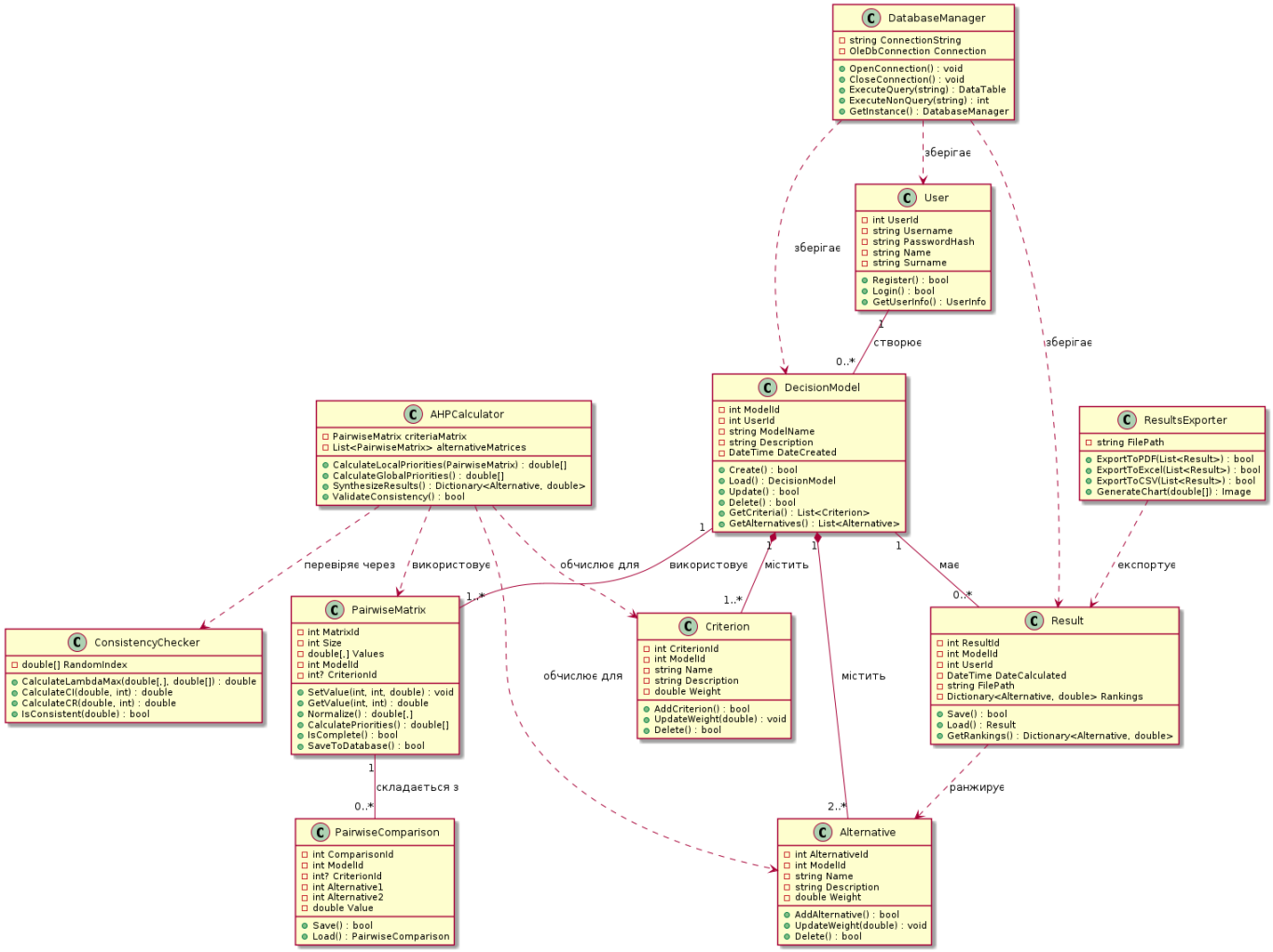


Рис. 2.6 — Діаграма класів системи

Діаграма класів описує статичну структуру системи та відношення між класами. Основні класи системи включають наступне. Клас User зберігає інформацію про користувача з полями UserId, Username, PasswordHash та має методи Register() та Login(). Клас DecisionModel представляє модель прийняття рішення з полями ModelId, UserId,

Name, Description, CreatedDate та методами Create(), Load(), Delete(). Клас Criterion описує критерій оцінювання з полями CriterionId, ModelId, Name, Description та методом AddCriterion(). Клас Alternative представляє альтернативу з полями AlternativeId, ModelId, Name, Description та методом AddAlternative(). Клас PairwiseMatrix реалізує матрицю парних порівнянь з полями MatrixId, Size, Values та методами SetValue(), GetValue(), Normalize(), CalculatePriorities(). Клас ConsistencyChecker виконує перевірку узгодженості з методами CalculateCI(), CalculateCR(), IsConsistent(). Клас AHPCalculator координує обчислення з методами CalculateLocalPriorities(), CalculateGlobalPriorities(), SynthesizeResults(). Клас ResultsExporter забезпечує експорт з методами ExportToPDF(), ExportToExcel(), ExportToCSV().

Відношення між класами включають композицію між DecisionModel та Criterion, Alternative, оскільки критерії та альтернативи не можуть існувати без моделі. Асоціація існує між User та DecisionModel, оскільки один користувач може мати багато моделей. Залежність встановлюється між AHPCalculator та іншими обчислювальними класами.

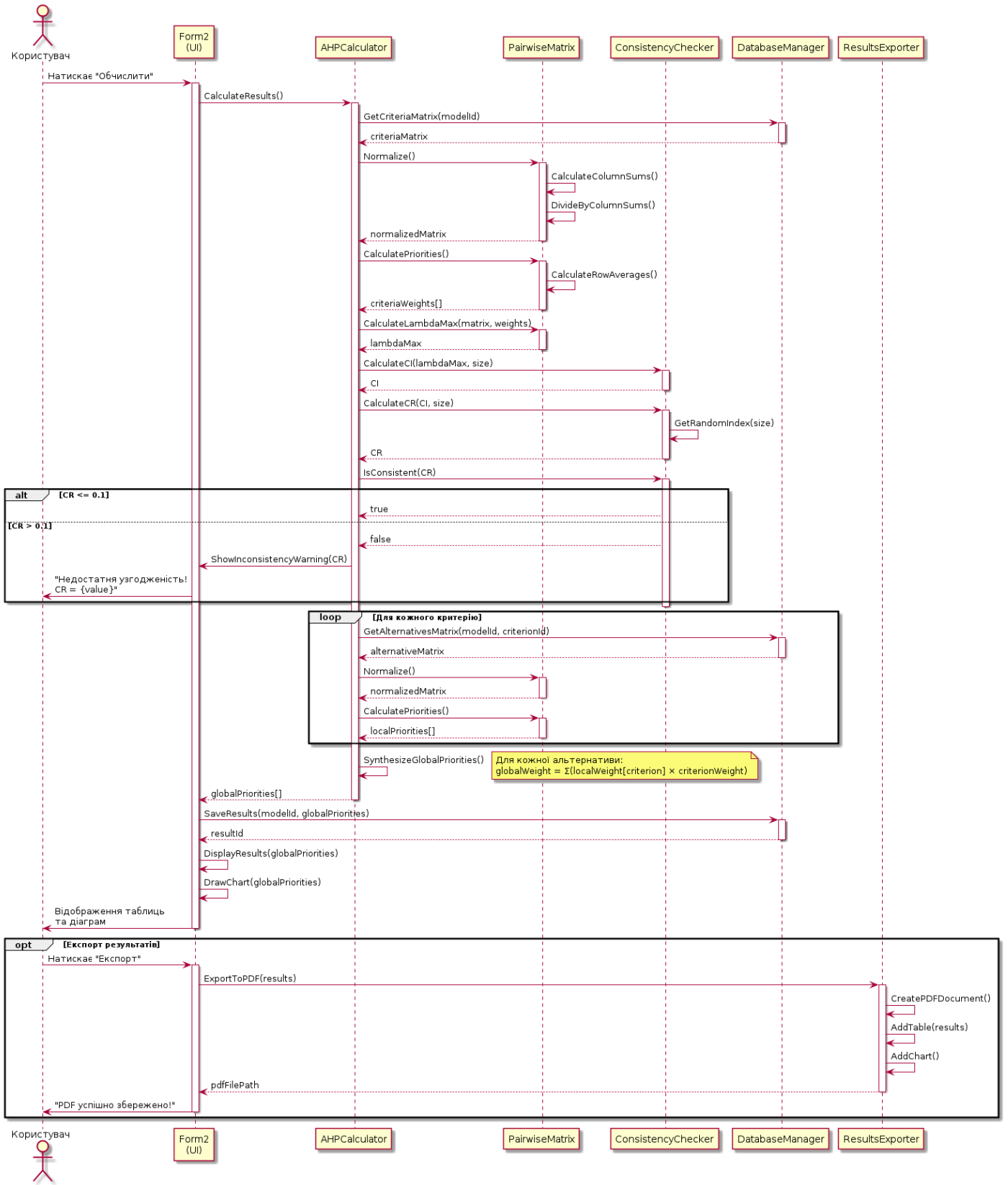


Рис. 2.7 — Діаграма послідовності процесу обчислення результатів

Діаграма послідовності для процесу обчислення результатів відображає взаємодію об'єктів у часі. Послідовність починається з того, що користувач ініціює обчислення через інтерфейс форми. Форма викликає метод CalculateResults() об'єкта AHPCalculator. AHPCalculator запитує матриці парних порівнянь через GetMatrix(). Для кожної матриці викликається метод Normalize() для нормалізації значень. Потім викликається CalculatePriorities() для обчислення локальних пріоритетів. AHPCalculator звертається до ConsistencyChecker для перевірки узгодженості через CalculateCR(). Якщо узгодженість прийнятна, виконується SynthesizeResults() для обчислення глобальних пріоритетів. Результати повертаються до форми та відображаються користувачу у вигляді таблиць та діаграм.

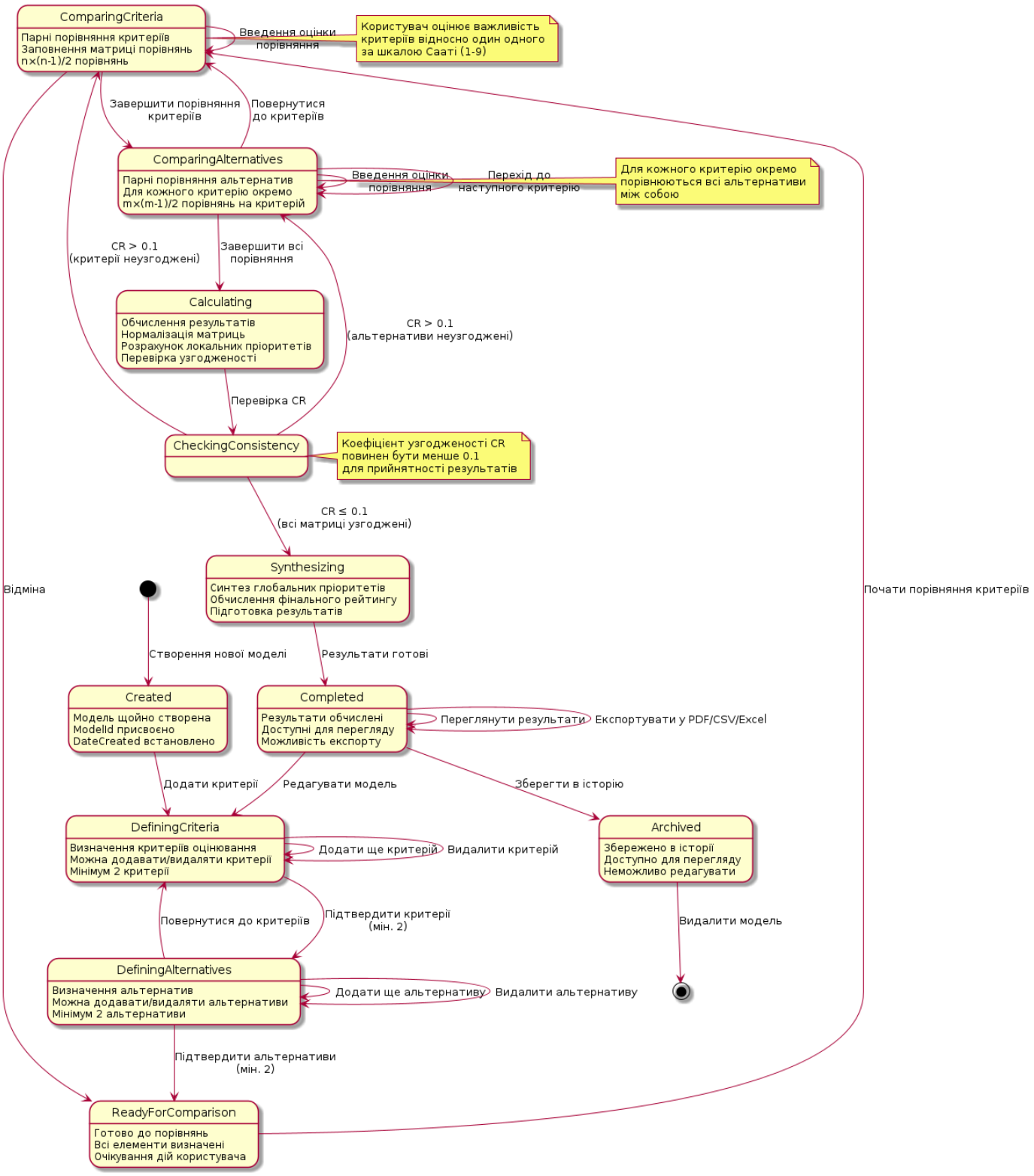


Рис. 2.8 — Діаграма станів моделі прийняття рішень

Діаграма станів моделі прийняття рішень показує життєвий цикл об'єкта DecisionModel. Початковий стан — Created, коли модель щойно створена. Перехід до стану DefiningCriteria відбувається при додаванні критеріїв. Після завершення додавання критеріїв модель переходить у стан DefiningAlternatives. Коли всі альтернативи додані, модель переходить у стан ReadyForComparison. У цьому стані можливе введення парних порівнянь, що переводить модель у стан ComparingElements. Після введення всіх порівнянь модель переходить у стан Calculating, де виконуються обчислення пріоритетів. Якщо узгодженість недостатня, модель повертається до стану ComparingElements для корекції оцінок. При прийнятній узгодженості модель переходить у фінальний стан Completed, де доступні результати та можливість експорту.

Діаграма станів включає не лише основні стани, але й підстани для деталізації. Переходи між станами супроводжуються умовами та діями.

Діаграма послідовності демонструє синхронні виклики методів між об'єктами. Синхронні виклики використовуються для операцій, які мають завершитися перед продовженням.

Діаграма класів відображає не лише структуру класів, але й методи з повними сигнатурами. Для кожного класу визначені модифікатори доступу.

Діаграма прецедентів детально описує всі можливі сценарії взаємодії користувача з системою. Кожен прецедент має чітко визначені передумови та постумови.

2.5. Вибір технологій реалізації

Вибір технологій для реалізації системи базується на аналізі вимог до функціональності, продуктивності, зручності розробки та підтримки. Розглянемо обґрунтування вибору основних технологій.

Для реалізації системи обрано мову програмування C# через декілька причин. По-перше, C# є сучасною об'єктно-орієнтованою мовою з багатим набором можливостей

для розробки складних додатків. По-друге, C# має потужну систему типів та підтримку generics, що дозволяє писати безпечний та ефективний код. По-третє, мова має відмінну підтримку роботи з базами даних через ADO.NET. По-четверте, C# забезпечує автоматичне управління пам'яттю через garbage collector, що знижує ризик помилок при роботі з пам'яттю. Важливою перевагою C# є наявність великої кількості бібліотек та фреймворків, зокрема для математичних обчислень, роботи з графікою та створення звітів. Мова має розвинену екосистему та активну спільноту розробників, що полегшує пошук рішень виникаючих проблем.

Система розробляється на платформі .NET Framework, яка надає всебічну підтримку для створення Windows-додатків. .NET Framework включає Common Language Runtime (CLR) для виконання керованого коду, бібліотеку класів Base Class Library (BCL) з великою кількістю готових компонентів, ADO.NET для роботи з базами даних, Windows Forms для створення графічного інтерфейсу, а також System.Drawing для роботи з графікою та діаграмами.

Використання .NET Framework забезпечує високу продуктивність додатку, надійність роботи та сумісність з операційною системою Windows. Платформа регулярно оновлюється компанією Microsoft, що гарантує підтримку сучасних технологій та виправлення виявлених вразливостей. Для створення графічного інтерфейсу користувача обрано технологію Windows Forms. Незважаючи на появу новіших технологій, таких як WPF та UWP, Windows Forms залишається відмінним вибором для створення традиційних desktop-додатків. Windows Forms надає широкий набір стандартних елементів управління, таких як кнопки, текстові поля, таблиці, що забезпечує швидку розробку інтерфейсу. Технологія має простий та зрозумілий API, що знижує поріг входження для розробників. Windows Forms додатки мають знайомий вигляд для користувачів Windows, що зменшує час на навчання роботі з системою. Наявність вбудованого візуального дизайнера у Visual Studio прискорює процес створення форм.

Для зберігання даних використовується Microsoft Access Database (файли .accdb). Цей вибір обумовлений наступними факторами. Access є файловою базою даних, що не вимагає встановлення окремого сервера бази даних. База даних зберігається у вигляді одного файлу, що спрощує розгортання та резервне копіювання. Access має достатню функціональність для задач системи, включаючи підтримку транзакцій та реляційних зв'язків. Робота з Access через ADO.NET є простою та не вимагає складних налаштувань. Для невеликих та середніх обсягів даних Access забезпечує прийнятну продуктивність.

Для візуалізації результатів у вигляді діаграм використовується вбудований компонент Chart з простору імен System.Windows.Forms.DataVisualization.Charting. Цей компонент надає можливість створювати різноманітні типи діаграм, включаючи стовпчасті, кругові, лінійні та інші. Chart має гнучкі можливості налаштування зовнішнього вигляду діаграм, підтримує легенди, осі координат та підписи даних. Компонент інтегрований з Windows Forms та не вимагає встановлення додаткових бібліотек.

Для експорту результатів у різні формати використовуються спеціалізовані бібліотеки. Експорт у PDF здійснюється за допомогою бібліотеки iTextSharp, яка надає потужні можливості для створення PDF-документів програмно. Експорт у Excel виконується через бібліотеку EPPlus або ClosedXML, які дозволяють створювати файли .xlsx без необхідності встановлення Microsoft Office. Експорт у CSV реалізується засобами стандартної бібліотеки .NET через класи для роботи з файлами та текстом. Для розробки системи використовується Microsoft Visual Studio — інтегроване середовище розробки (IDE) для мов платформи .NET. Visual Studio надає потужні інструменти для написання, налагодження та тестування коду. Середовище включає візуальний дизайнер для створення форм Windows Forms, інтелектуальне автодоповнення коду IntelliSense, вбудований дебагер з можливістю покрокового виконання та перегляду змінних, інтеграцію з системами контролю версій, а також інструменти для профілювання та оптимізації коду.

Використання Microsoft Access має перевагу у вигляді легкого перегляду даних. Це полегшує налагодження та адміністрування системи. Windows Forms має перевагу стабільності та зрілості. Технологія забезпечує довгострокову сумісність та передбачувану поведінку.

Платформа .NET Framework забезпечує високий рівень безпеки через механізми Code Access Security. Керований код виконується у захищеному середовищі.

Вибір C# обумовлений підтримкою асинхронного програмування через конструкції async/await. Наявність асинхронних можливостей створює основу для майбутнього вдосконалення.

2.6. Висновки до розділу 2

У другому розділі було виконано постановку задачі та проєктування системи підтримки прийняття рішень у небезпечних ситуаціях. Математично сформульовано задачу багатокритеріального прийняття рішень з використанням методу аналізу ієрархій. Визначено специфічні вимоги до системи, що випливають з особливостей роботи у критичних ситуаціях.

Розроблено детальні функціональні вимоги до системи, які охоплюють всі аспекти роботи користувача з системою, від реєстрації та авторизації до експорту результатів. Спроектовано трирівневу архітектуру системи з чітким розділенням відповідальності між рівнем представлення, бізнес-логіки та даних.

Створено комплект UML-діаграм, що формалізують проєктні рішення та забезпечують чітке розуміння структури системи та взаємодії її компонентів. Діаграми включають діаграму прецедентів для відображення функціональності з точки зору користувача, діаграму класів для опису статичної структури, діаграму послідовності для ілюстрації взаємодії об'єктів у часі, а також діаграму станів для відображення життєвого циклу моделі прийняття рішень.

Обґрунтовано вибір технологій реалізації системи, включаючи мову програмування C#, платформу .NET Framework, технологію Windows Forms для графічного інтерфейсу, Microsoft Access для зберігання даних та спеціалізовані бібліотеки для візуалізації та експорту результатів. Вибрані технології забезпечують баланс між функціональністю, продуктивністю та зручністю розробки.

Результати другого розділу створюють міцну основу для програмної реалізації системи, яка буде розглянута у наступному розділі роботи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Опис розроблених модулів

Програмна реалізація системи підтримки прийняття рішень виконана відповідно до спроектованої у другому розділі архітектури. Система складається з декількох функціональних модулів, кожен з яких відповідає за конкретну частину функціональності. Розглянемо детально реалізацію кожного модуля.

Модуль роботи з базою даних реалізований у вигляді класу DatabaseManager, який забезпечує всі операції з базою даних Microsoft Access. Клас використовує ADO.NET та бібліотеку System.Data.OleDb для підключення до файлу бази даних. Основні методи класу включають встановлення з'єднання з базою даних через метод OpenConnection(), виконання SQL-запитів на вибірку даних методом ExecuteQuery(), виконання команд на модифікацію даних методом ExecuteNonQuery(), а також закриття з'єднання через CloseConnection(). Рядок підключення до бази даних формується з використанням провайдера Microsoft.ACE.OLEDB.12.0 для роботи з файлами формату .accdb. Всі операції з базою даних виконуються в блоках try-catch для обробки можливих помилок підключення або виконання запитів. Клас DatabaseManager реалізує патерн Singleton для забезпечення єдиного екземпляра підключення до бази даних протягом роботи додатку [28].

Структура бази даних включає таблицю Users для зберігання облікових записів користувачів з полями UserId, Username та PasswordHash (рис.3.1).

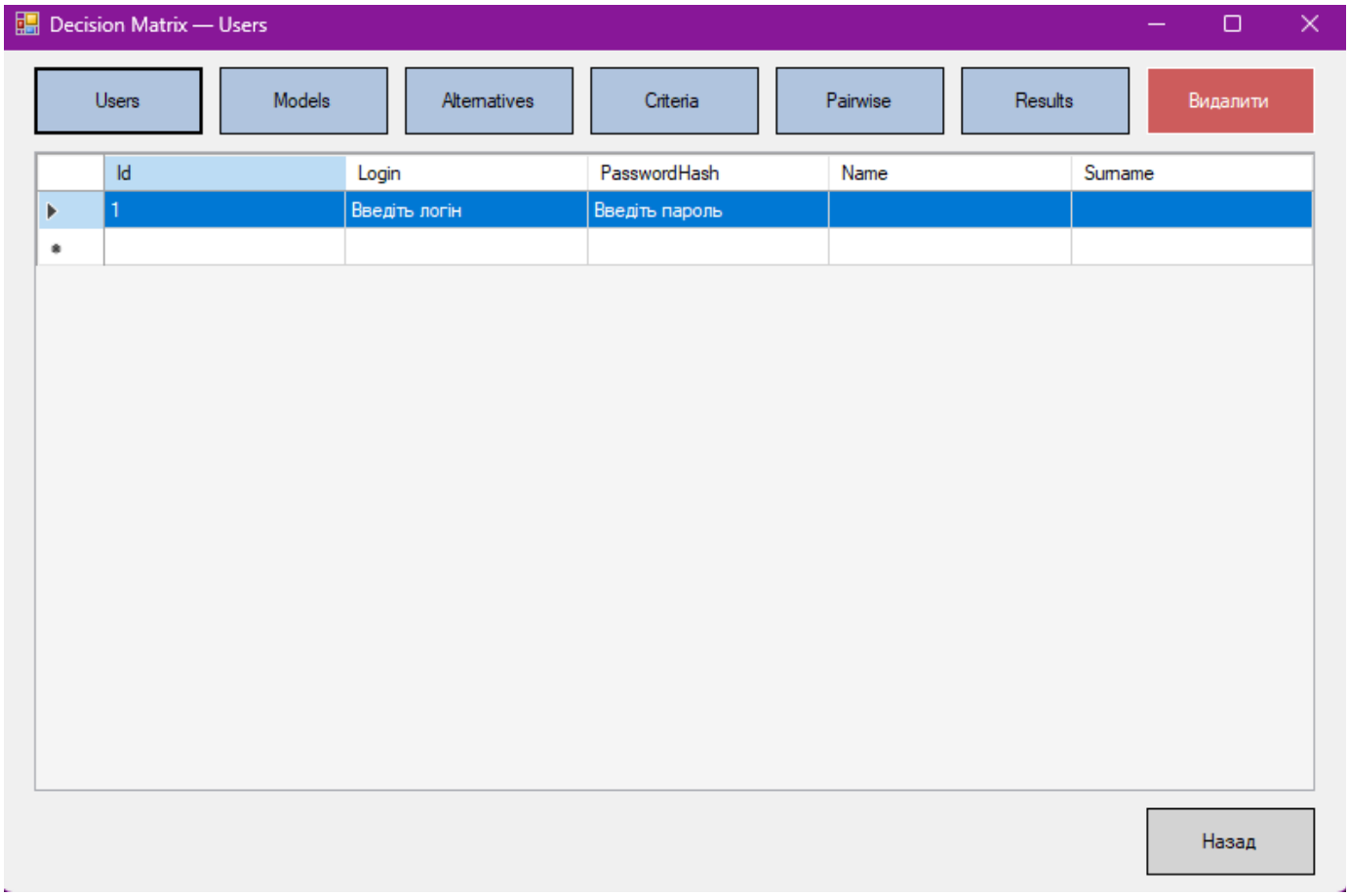


Рис. 3.1 - Інтерфейс адміністратора для перегляду та управління користувачами системи

Таблиця Models зберігає інформацію про створені моделі прийняття рішень з полями ModelId, UserId, ModelName, Description та CreatedDate. Таблиця Criteria містить критерії оцінювання з полями CriterionId, ModelId, CriterionName та Description. Таблиця Alternatives зберігає альтернативи з полями AlternativeId, ModelId, AlternativeName та Description. Таблиці CriteriaComparisons та AlternativeComparisons зберігають матриці парних порівнянь з відповідними ідентифікаторами та значеннями порівнянь. Таблиця Results містить результати обчислень з пріоритетами альтернатив та коефіцієнтами узгодженості.

Модуль автентифікації реалізований у класі AuthenticationManager, який забезпечує безпечну роботу з обліковими записами користувачів. При реєстрації нового користувача пароль хешується за допомогою алгоритму SHA256 з використанням класу System.Security.Cryptography.SHA256 (рис.3.2-3.3).

	Id	Login	PasswordHash
▶	1	Введіть логін	Введіть пароль
*			

Рис.3.2 - Структура таблиці користувачів з хешованими паролями в базі даних.

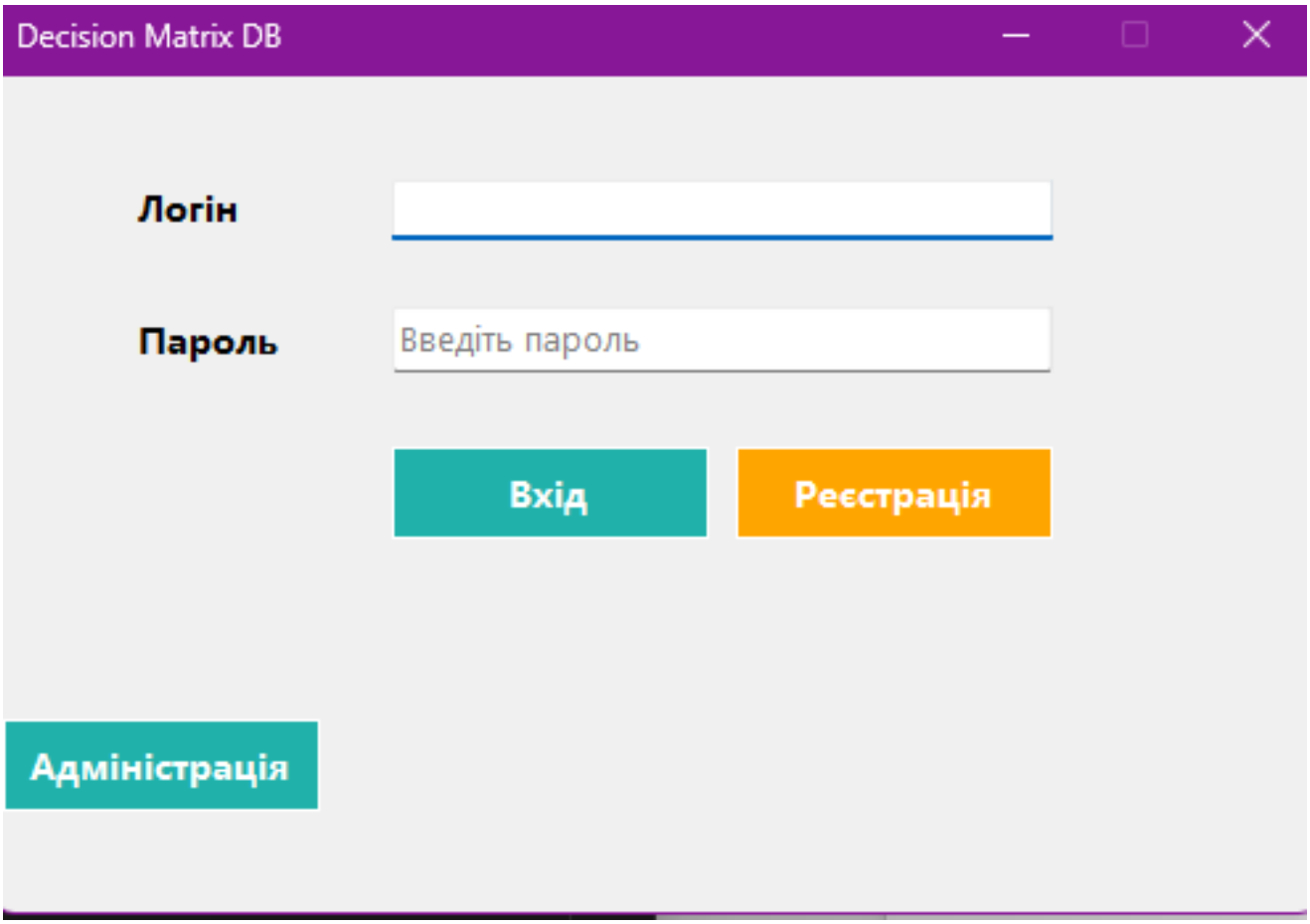


Рис. 3.3 – Форма автентифікації користувача з полями введення логіна та пароля

Хеш пароля зберігається в базі даних у вигляді рядка, що представляє послідовність байтів у шістнадцятковому форматі.

При авторизації користувача введений пароль хешується тим самим алгоритмом, і отриманий хеш порівнюється з збереженим у базі даних. У разі збігу хешів доступ надається, інакше відображається повідомлення про помилку. Після успішної авторизації у глобальній змінній додатку зберігається ідентифікатор поточного користувача для використання у наступних операціях. Форма LoginForm надає інтерфейс для входу в систему з полями для введення імені користувача та пароля. Форма також містить посилання на форму реєстрації RegisterForm, де новий користувач може створити обліковий запис (рис.3.4).

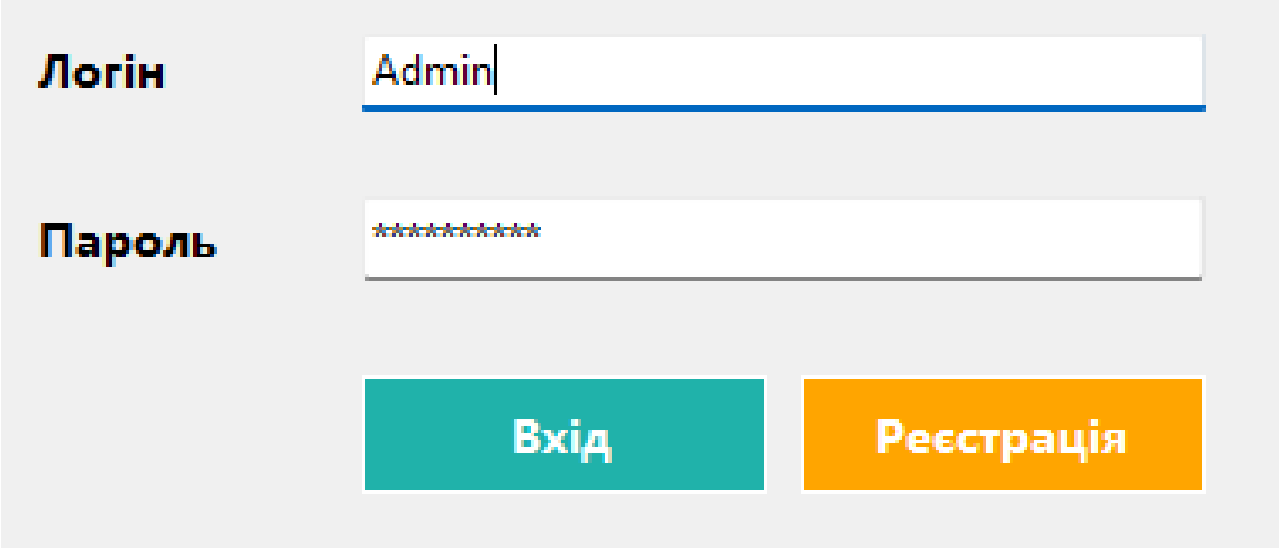


Рис. 3.4 - Форма авторизації користувача в системі

Обидві форми виконують валідацію введених даних перед відправкою на сервер, перевіряючи, що всі обов'язкові поля заповнені та відповідають встановленим вимогам.

Модуль управління моделями реалізований у класі ModelManager та формі MainForm. Після входу в систему користувач потрапляє на головну форму, де

відображається список його моделей прийняття рішень. Список реалізований за допомогою компонента DataGridView, який відображає назву моделі, опис та дату створення.

Користувач може створити нову модель, натиснувши кнопку Create New Model, що відкриває форму CreateModelForm. У цій формі вводяться назва та опис нової моделі. Після збереження інформація записується в таблицю Models з прив'язкою до ідентифікатора поточного користувача. Модуль також надає можливість відкрити існуючу модель для перегляду або редагування, видалити модель з бази даних після підтвердження користувачем (рис.3.5-3.7).

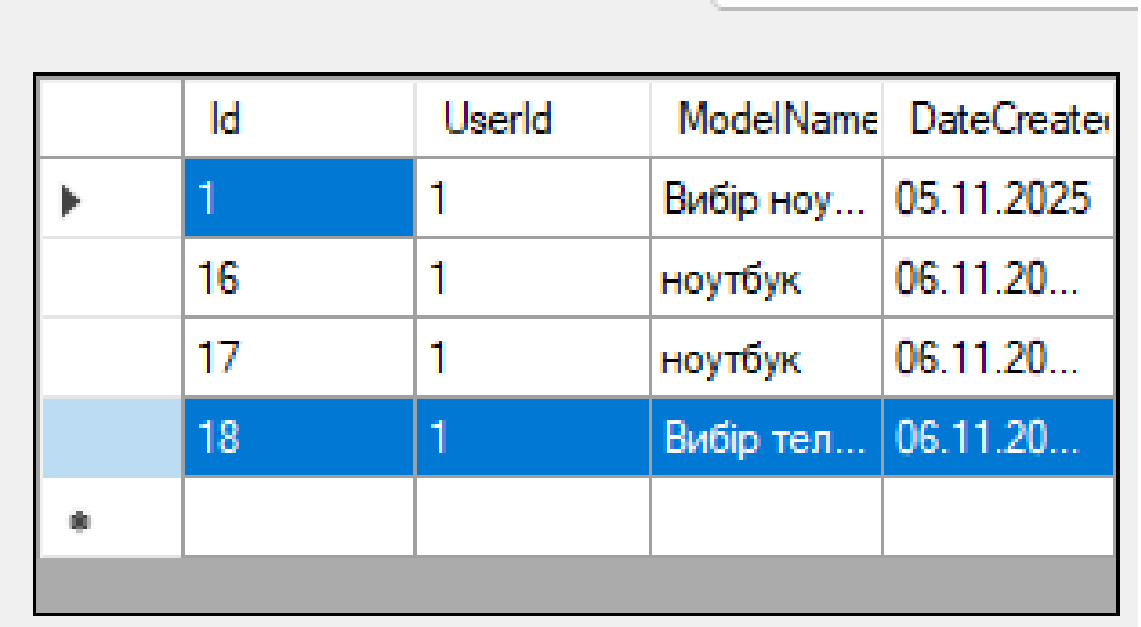


Рис. 3.5 - Головне вікно системи з переліком створених моделей користувача

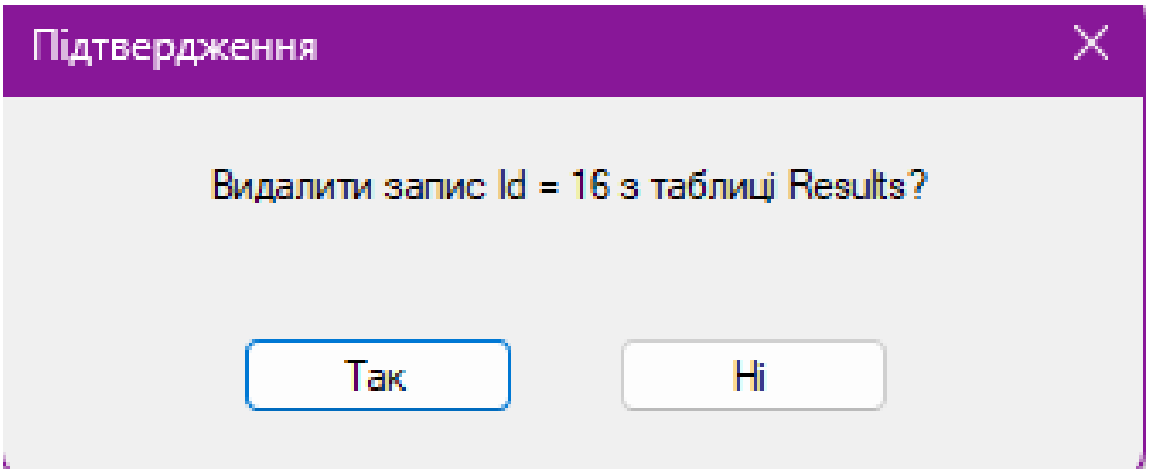


Рис. 3.6 - Діалогове вікно підтвердження видалення запису з бази даних

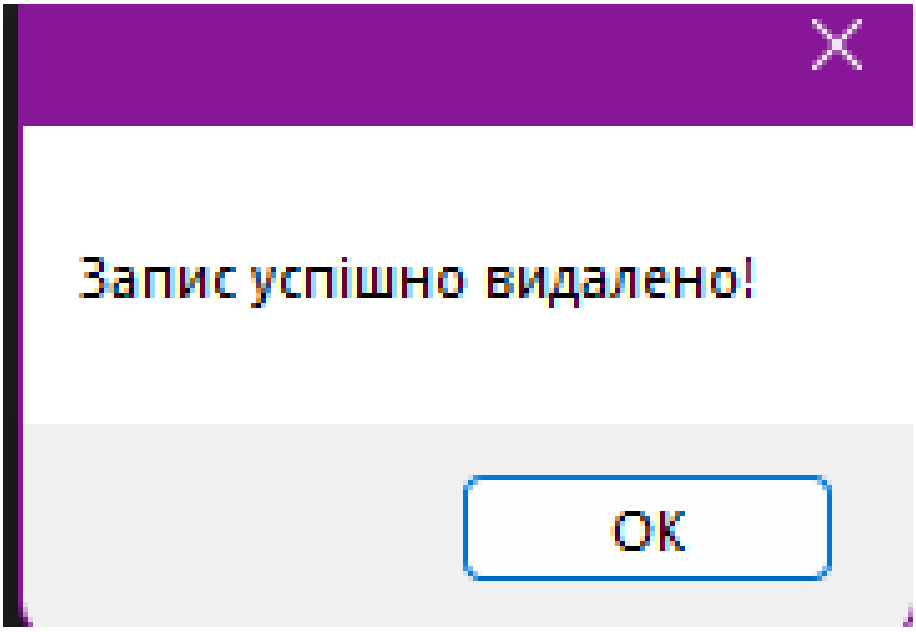


Рис. 3.7 - Повідомлення про успішне видалення запису з системи

Клас ModelManager інкапсулює всю логіку роботи з моделями, включаючи методи CreateModel() для створення нової моделі, LoadModel() для завантаження існуючої моделі з бази даних, UpdateModel() для оновлення інформації про модель та DeleteModel() для видалення моделі. Всі операції супроводжуються відповідними SQL-запитами до бази даних через DatabaseManager.

Після створення або відкриття моделі користувач переходить до етапу визначення критеріїв та альтернатив. Форма ModelEditForm містить дві вкладки для роботи з критеріями та альтернативами відповідно. Кожна вкладка містить DataGridView для відображення списку елементів та кнопки для додавання та видалення елементів.

При додаванні нового критерію або альтернативи відкривається діалогове вікно, де користувач вводить назву та опис елемента. Після підтвердження інформація зберігається в відповідній таблиці бази даних з прив'язкою до ідентифікатора поточної моделі. Система автоматично присвоює кожному елементу унікальний ідентифікатор. Видалення критерію або альтернативи виконується після підтвердження користувачем. При видаленні критерію також видаляються всі пов'язані з ним парні порівняння альтернатив. Це забезпечує цілісність даних та уникнення некоректних посилань у базі даних.

Модуль парних порівнянь є ключовим компонентом системи, що реалізує процес збору експертних оцінок. Після визначення всіх критеріїв та альтернатив користувач переходить до етапу порівнянь, натиснувши відповідну кнопку на формі ModelEditForm.

Форма ComparisonForm відображає матрицю парних порівнянь у вигляді DataGridView. Спочатку користувач порівнює критерії між собою відносно загальної мети. Матриця має розмір n×n, де n — кількість критеріїв. Діагональні елементи матриці [29] завжди дорівнюють 1, оскільки кожен елемент рівний сам собі. Верхня трикутна частина матриці заповнюється користувачем, а нижня трикутна частина обчислюється автоматично як обернені значення (рис.3.8).

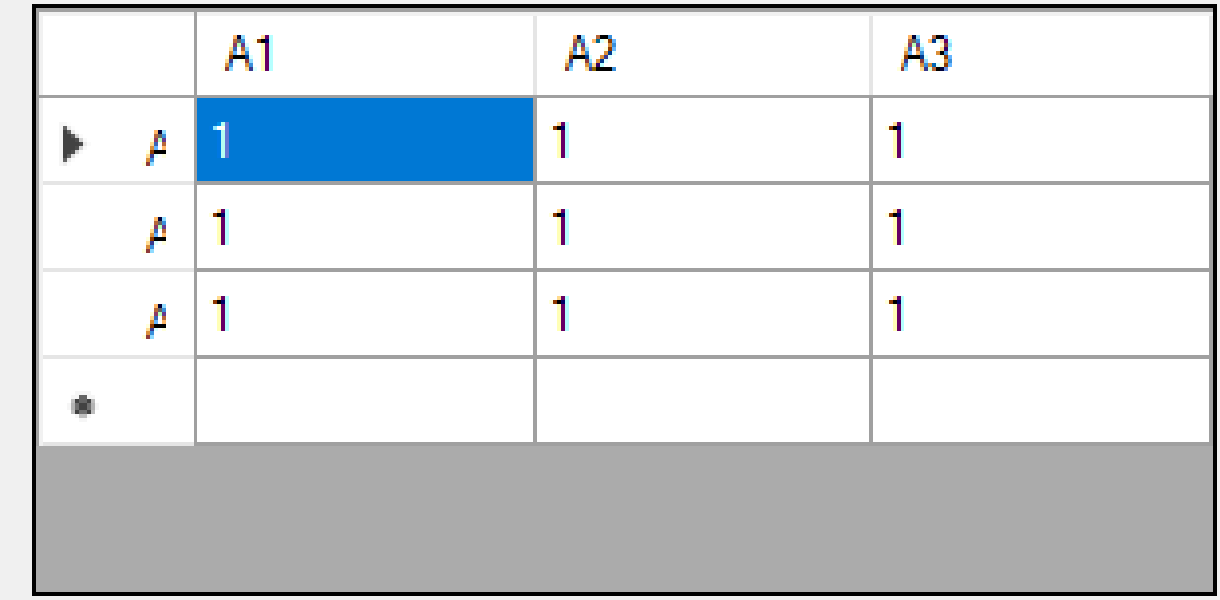


Рис. 3.8 - Інтерфейс заповнення матриці парних порівнянь альтернатив

Для введення значень порівнянь використовується комбінований підхід. Користувач може вибрати значення зі спадного списку, що містить числа шкали Сааті від 1 до 9 з поясненнями їх значень, або ввести власне дробове значення для більш точної оцінки. Після введення значення у комірку верхньої трикутної частини матриці система автоматично обчислює та встановлює обернене значення у відповідну комірку нижньої трикутної частини.

Після завершення порівняння критеріїв система переходить до порівняння альтернатив відносно кожного критерію. Для кожного критерію створюється окрема матриця розміром m×m, де m — кількість альтернатив. Користувач послідовно заповнює всі матриці, а система зберігає введені значення в таблиці AlternativeComparisons бази даних.

Клас PairwiseMatrix інкапсулює логіку роботи з матрицями порівнянь. Він містить методи SetValue() для встановлення значення у конкретну комірку матриці з автоматичним обчисленням оберненого значення, GetValue() для отримання значення з

матриці, IsComplete() для перевірки, чи всі необхідні комірки матриці заповнені, а також SaveToDatabase() для збереження матриці в базі даних.

Для забезпечення безпеки включено механізм автоматичного резервного копіювання. Перед критичними операціями створюється резервна копія.

Модуль управління моделями включає функціонал для експорту та імпорту. Модель може бути експортована у формат JSON або XML. Важливою особливістю модуля автентифікації є використання salt-значень при хешуванні паролів. Це робить неможливими атаки з використанням rainbow tables.

Модуль роботи з базою даних реалізує пул з'єднань для оптимізації продуктивності. Правильне управління з'єднаннями забезпечує ефективну роботу системи.

3.2. Алгоритмічне забезпечення: реалізація методу аналізу ієрархій

Після завершення введення всіх парних порівнянь система переходить до етапу обчислень. Реалізація методу аналізу ієрархій виконується у класі AHPCalculator, який містить всі необхідні алгоритми для обробки матриць та обчислення пріоритетів.

Перший етап обчислень полягає у нормалізації матриці парних порівнянь. Метод NormalizeMatrix() виконує наступні операції. Для кожного стовпця матриці обчислюється сума всіх його елементів. Потім кожен елемент стовпця ділиться на обчислену суму, що призводить до отримання нормалізованої матриці, в якій сума елементів кожного стовпця дорівнює одиниці. Нормалізована матриця зберігається у новому двовимірному масиві для подальших обчислень [30].

Алгоритм нормалізації реалізований з використанням циклів для обходу всіх елементів матриці. Спочатку обчислюються суми стовпців та зберігаються у одновимірному масиві. Потім виконується друга ітерація по всіх елементах матриці для виконання ділення на відповідні суми. Локальні пріоритети елементів обчислюються як

середні арифметичні значення рядків нормалізованої матриці. Метод CalculateLocalPriorities() виконує наступні кроки. Для кожного рядка нормалізованої матриці обчислюється сума всіх його елементів. Отримана сума ділиться на кількість елементів у рядку, що дає середнє арифметичне значення. Це середнє значення і є локальним пріоритетом відповідного елемента.

Результатом роботи методу є одновимірний масив локальних пріоритетів, довжина якого дорівнює розмірності матриці. Сума всіх елементів цього масиву дорівнює одиниці, що відповідає математичним властивостям методу власного вектора. Локальні пріоритети представляють відносну важливість елементів у рамках конкретного рівня ієрархії [31, с. 442].

Для перевірки узгодженості матриці необхідно обчислити її максимальне власне значення λmax. Метод CalculateLambdaMax() реалізує наближений алгоритм обчислення. Виконується множення вихідної матриці парних порівнянь на вектор локальних пріоритетів, що дає новий вектор. Кожен елемент отриманого вектора ділиться на відповідний елемент вектора пріоритетів. Максимальне власне значення обчислюється як середнє арифметичне отриманих відношень. Значення λmax завжди більше або дорівнює розмірності матриці n. Чим ближче λmax до n, тим більш узгодженою є матриця порівнянь. Різниця між λmax та n використовується для обчислення індексу узгодженості.

Перевірка узгодженості експертних оцінок є важливим етапом методу аналізу ієрархій. Клас ConsistencyChecker містить методи для обчислення показників узгодженості. Метод CalculateCI() обчислює індекс узгодженості за формулою

$$CI = \frac{\lambda \max - n}{n - 1}, \tag{2}$$

де n — розмірність матриці.

Метод CalculateCR() обчислює коефіцієнт узгодженості як відношення $CR = \frac{CI}{RI}$, де RI — випадковий індекс, який залежить від розмірності матриці. Значення випадкового

індексу для різних розмірностей зберігаються у масиві констант: для $n = 1$ $RI = 0$, для $n = 2$ $RI = 0$, для $n = 3$ $RI = 0.58$, для $n = 4$ $RI = 0.90$, для $n = 5$ $RI = 1.12$, для $n = 6$ $RI = 1.24$, для $n = 7$ $RI = 1.32$, для $n = 8$ $RI = 1.41$, для $n = 9$ $RI = 1.45$, для $n = 10$ $RI = 1.49$. Значення випадкового індексу RI для різних розмірностей матриць наведені у таблиці 3.1.

Таблиця 3.1

Випадкові індекси узгодженості для різних розмірностей матриць

n (розмірність матриці)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RI (випадковий індекс)	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Ці значення отримані експериментально Томасом Сааті на основі статистичного аналізу великої кількості випадково згенерованих матриць. Для матриць розмірності 1 та 2 випадковий індекс дорівнює нулю, оскільки такі матриці завжди є узгодженими.

Метод IsConsistent() перевіряє, чи коефіцієнт узгодженості не перевищує критичного значення 0.1. Якщо CR > 0.1, матриця вважається неузгодженою, і користувач отримує повідомлення про необхідність перегляду своїх оцінок. Система відображає діалогове вікно з інформацією про поточне значення коефіцієнта узгодженості та рекомендаціями щодо коригування оцінок.

Фінальний етап обчислень полягає у синтезі глобальних пріоритетів альтернатив. Метод SynthesizeGlobalPriorities() виконує зважене підсумовування локальних пріоритетів альтернатив відносно кожного критерію з використанням ваг критеріїв. Для кожної альтернативи обчислюється глобальний пріоритет за формулою: глобальний пріоритет дорівнює сумі добутків локального пріоритету альтернативи відносно кожного критерію на вагу цього критерію. Результатом роботи методу є масив глобальних пріоритетів альтернатив, сума елементів якого дорівнює одиниці. Альтернативи

сортуються за спаданням глобальних пріоритетів для отримання фінального рейтингу. Альтернатива з найвищим глобальним пріоритетом є найкращим рішенням згідно з методом аналізу ієрархій та експертними оцінками користувача [32, с. 879-907].

Всі обчислені результати зберігаються в таблиці Results бази даних для можливості їх подальшого перегляду та аналізу. Для кожної альтернативи зберігається її глобальний пріоритет, позиція у рейтингу та локальні пріоритети відносно кожного критерію (рис.3.9).

Id	ModelId	UserId	DateD	FilePath
1	1	1	05.11.2025	C:\DecisionMatrixDB\Res...
6	16	1	06.11.2025 20:44	AutoSave
7	16	1	06.11.2025 20:44	AutoSave
8	16	1	06.11.2025 20:44	AutoSave
10	16	1	06.11.2025 20:50	AutoSave
11	16	1	06.11.2025 20:50	AutoSave
12	16	1	06.11.2025 20:53	AutoSave
13	16	1	06.11.2025 20:53	AutoSave
14	16	1	06.11.2025 20:53	AutoSave
15	16	1	06.11.2025 20:55	AutoSave
16	16	1	06.11.2025 20:55	AutoSave
17	16	1	06.11.2025 20:55	AutoSave
18	16	1	06.11.2025 20:56	AutoSave
19	16	1	06.11.2025 20:56	AutoSave
20	16	1	06.11.2025 20:56	AutoSave
21	16	1	06.11.2025 20:56	AutoSave

Рис. 3.9 - Детальний перегляд таблиці Results з історією автоматичного збереження розрахунків

Також зберігаються коефіцієнти узгодженості для всіх матриць порівнянь.

Синтез глобальних пріоритетів включає додатковий етап нормалізації. Фінальна нормалізація усуває похибки округлення. Важливою особливістю є надання детальної інформації про джерела неузгодженості. Система виявляє пари елементів, що порушують транзитивність.

Обчислення локальних пріоритетів може бути виконано різними способами. Для малих матриць використовується спрощений метод. Алгоритм нормалізації реалізований з урахуванням можливості виникнення нульових елементів. Система перевіряє коректність всіх елементів матриці.

3.3. Тестування та приклади роботи системи

Для перевірки коректності роботи системи та демонстрації її можливостей було проведено тестування на практичних прикладах прийняття рішень у небезпечних ситуаціях. Розглянемо детально процес роботи системи на конкретному прикладі.

Розглянемо задачу вибору оптимального маршруту евакуації людей з будівлі при виникненні пожежі. Особа, що приймає рішення, має обрати один з трьох можливих маршрутів на основі чотирьох критеріїв оцінювання (рис.3.10-3.12).

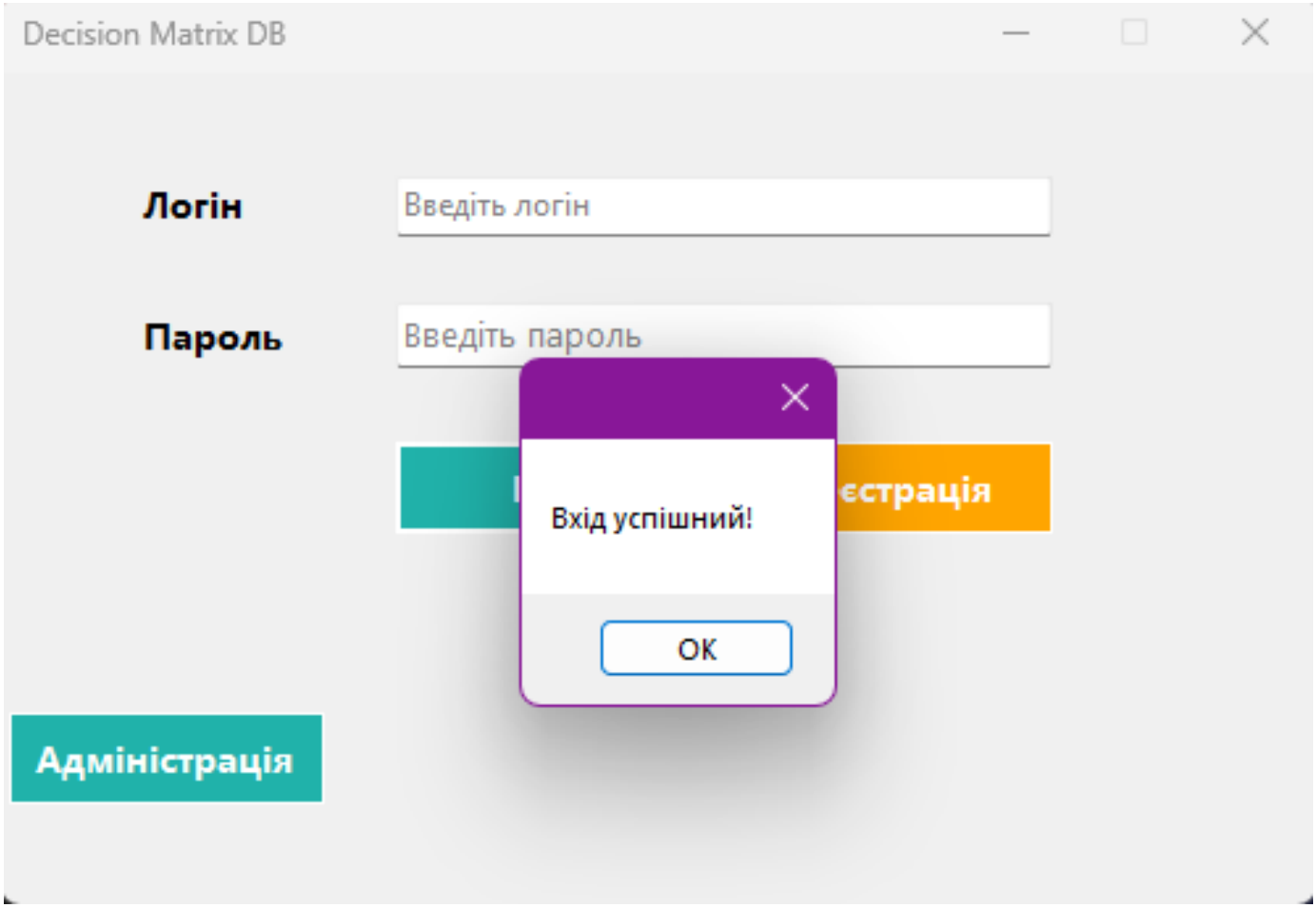


Рис. 3.10 - Форма автентифікації з повідомленням про успішний вхід у систему

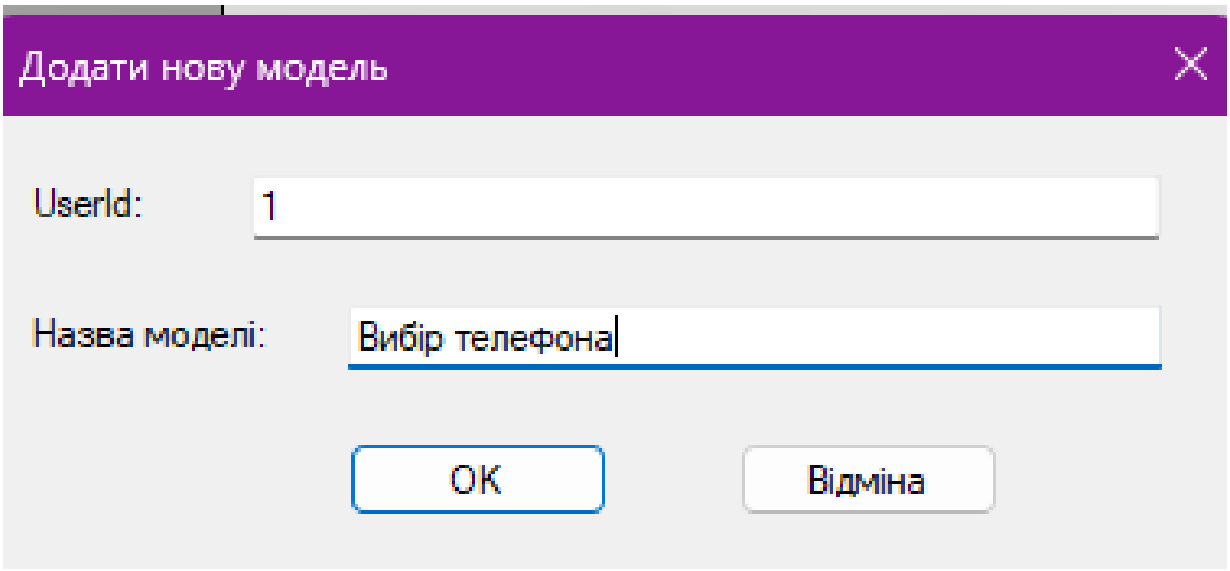


Рис. 3.11 – Діалогове вікно створення нової моделі прийняття рішень з полями UserId та назви моделі

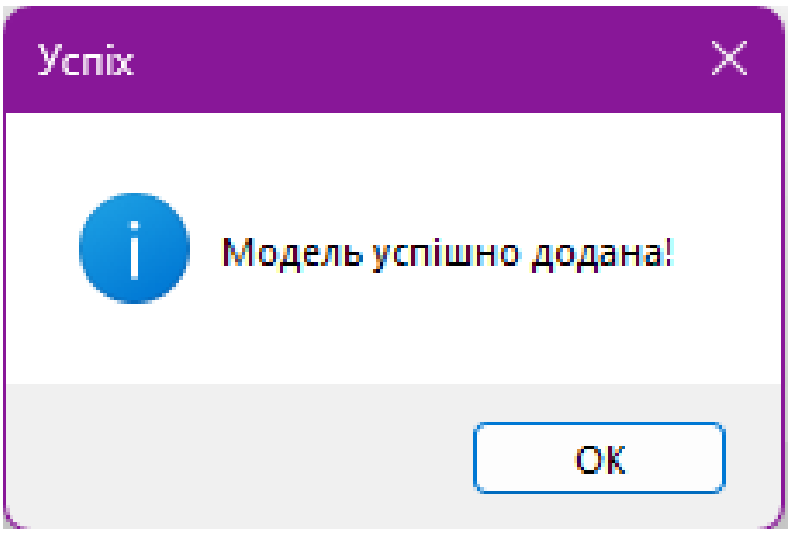


Рис. 3.12 – Повідомлення про успішне створення та додавання нової моделі в систему

Альтернативи представлені трьома маршрутами евакуації. Маршрут А передбачає використання головних сходів, які є найкоротшими, але можуть бути задимлені. Маршрут Б використовує запасні сходи, які довші, але безпечніші. Маршрут В передбачає евакуацію через пожежну драбину на зовнішній стіні будівлі. Критерії оцінювання включають швидкість евакуації, безпеку маршруту, пропускну здатність та надійність. Критерій швидкості евакуації оцінює час, необхідний для виходу з будівлі. Критерій безпеки враховує ризики на маршруті, такі як задимлення або висота. Критерій пропускної здатності відображає можливість одночасної евакуації великої кількості людей. Критерій надійності оцінює ймовірність доступності маршруту в умовах пожежі.

Після створення моделі в системі користувач вводить парні порівняння критеріїв. Експерт визначає, що безпека маршруту є значно важливішою за швидкість, надаючи їй оцінку 5 за шкалою Сааті. Пропускна здатність має помірну перевагу над швидкістю з

оцінкою 3. Надійність є рівною за важливістю з пропускною здатністю, отримуючи оцінку 1. Після введення всіх порівнянь система обчислює локальні пріоритети критеріїв.

Результати обчислення показують, що критерій безпеки отримав найвищу вагу 0.47, що відповідає 47 відсоткам загальної важливості. Критерій надійності має вагу 0.26 або 26 відсотків. Пропускна здатність отримала вагу 0.18 або 18 відсотків. Швидкість евакуації має найменшу вагу 0.09 або 9 відсотків. Коефіцієнт узгодженості для матриці критеріїв склав 0.03, що значно нижче граничного значення 0.1 і свідчить про високу узгодженість експертних оцінок.

Далі користувач порівнює альтернативи відносно кожного критерію. При оцінюванні за критерієм швидкості Маршрут А отримує перевагу над іншими маршрутами. При оцінюванні за критерієм безпеки Маршрут Б демонструє значну перевагу, оскільки запасні сходи зазвичай менш схильні до задимлення. За критерієм пропускної здатності також лідирує Маршрут Б завдяки ширині запасних сходів. За критерієм надійності всі маршрути оцінюються приблизно однаково.

Після введення всіх парних порівнянь система виконує обчислення глобальних пріоритетів. Результати показують, що Маршрут Б отримав найвищий глобальний пріоритет 0.52, що робить його оптимальним вибором. Маршрут А отримав пріоритет 0.28 і займає друге місце. Маршрут В має найнижчий пріоритет 0.20 і є найменш привабливим варіантом у даній ситуації.

Результати обчислень відображаються у формі ResultsForm, яка містить декілька вкладок для різних представлень даних. Перша вкладка показує таблицю ваг критеріїв з їх процентними значеннями. Друга вкладка містить детальну таблицю локальних пріоритетів альтернатив відносно кожного критерію (рис.3.13-3.14).

	Альтернатива	Вага
▶	A1	0,196
	A2	0,413
	A3	0,391
*		

Рис. 3.13 - Вікно результатів із фінальним рейтингом альтернатив за глобальними пріоритетами

A1	0,196
A2	0,413
A3	0,391

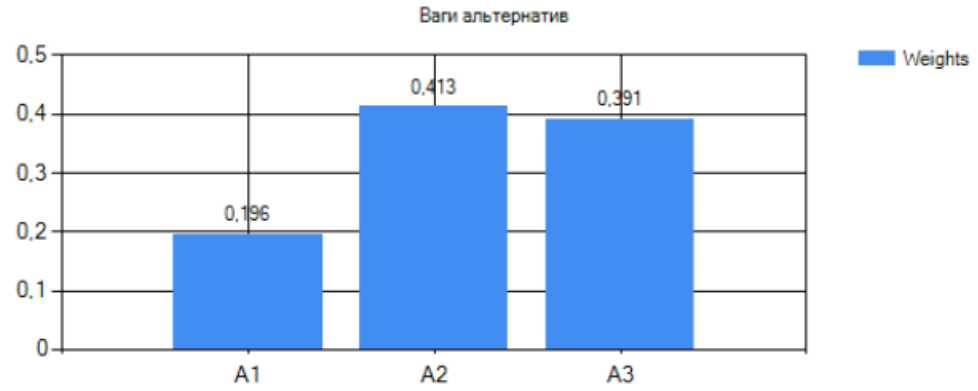


Рис. 3.14 - Комбіноване представлення результатів: таблиця та діаграма глобальних пріоритетів

Третя вкладка відображає фінальний рейтинг альтернатив з глобальними пріоритетами. Результати обчислення глобальних пріоритетів для задачі вибору маршруту евакуації наведені у таблиці 3.2.

Таблиця 3.2

Результати обчислення глобальних пріоритетів для задачі вибору маршруту евакуації

Альтернатива	**Швидкість (0.09)**	**Безпека (0.47)**	**Пропускна здатність (0.18)**	**Надійність (0.26)**	**Глобальна вага**	**Рейтинг**
Маршрут А (головні сходи)	0.54	0.16	0.20	0.33	0.28	2
Маршрут Б (запасні сходи)	0.30	0.58	0.53	0.34	0.52	1
Маршрут В (пожежна драбина)	0.16	0.26	0.27	0.33	0.20	3
Сума	1.00	1.00	1.00	1.00	1.00	—

Як видно з таблиці 3.2, Маршрут Б (запасні сходи) отримав найвищий глобальний пріоритет 0.52 і займає перше місце у рейтингу. Це пояснюється високою вагою критерію безпеки (0.47), за яким цей маршрут має найкращу оцінку (0.58). Маршрут А займає друге місце з вагою 0.28, незважаючи на найвищу швидкість евакуації, оскільки цей

критерій має найнижчу вагу (0.09). Маршрут В через пожежну драбину виявився найменш привабливим варіантом з глобальною вагою 0.20.

Для наочного представлення результатів система генерує графічні діаграми. Стовпчаста діаграма показує порівняння глобальних пріоритетів альтернатив, де висота кожного стовпця пропорційна відповідному пріоритету (рис.3.15).

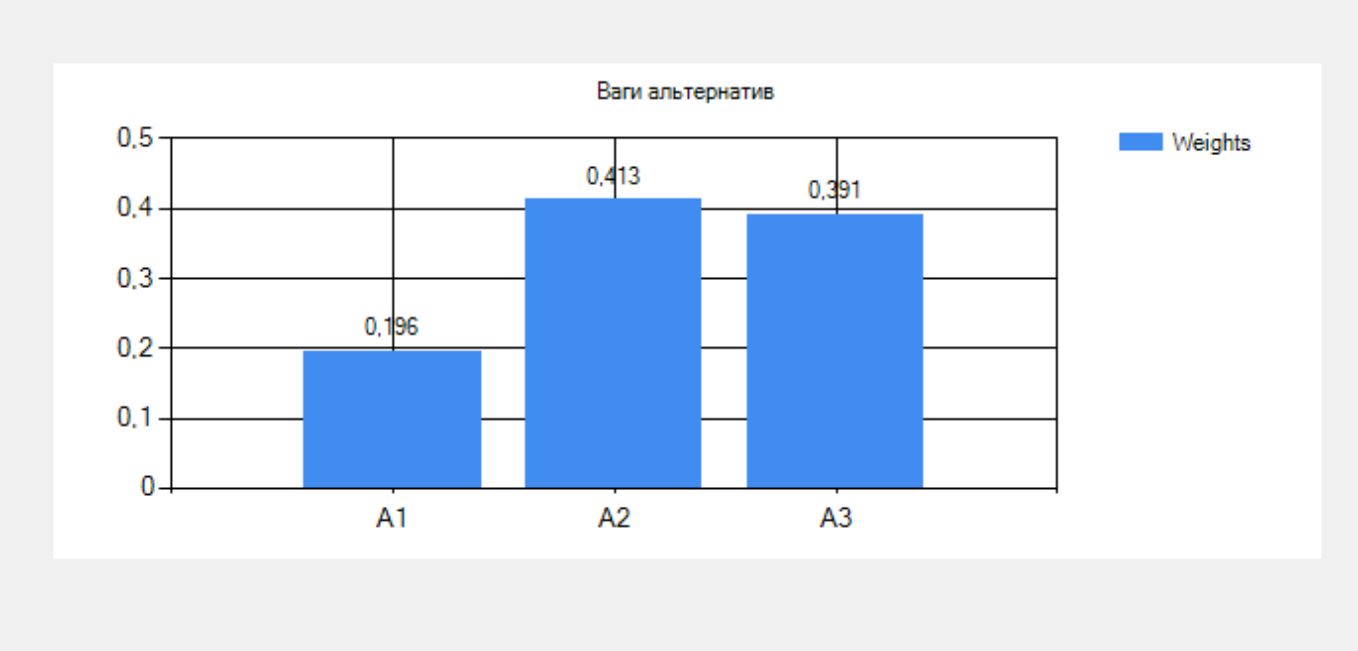


Рис. 3.15 - Стовпчаста діаграма глобальних пріоритетів альтернатив

Кругова діаграма відображає розподіл ваг критеріїв, де кожен сектор представляє один критерій, а його розмір відповідає вазі критерію у відсотках.

Діаграми створюються за допомогою компонента Chart з бібліотеки System.Windows.Forms.DataVisualization.Charting. Для стовпчастої діаграми використовується тип серії Column, а для кругової — тип Pie. Діаграми мають налаштовані легенди, підписи осей та підписи значень для кращого сприйняття інформації.

Система надає можливість експортувати результати у різні формати для подальшого використання. При експорті у формат PDF створюється документ, що містить назву моделі, дату створення, таблиці з результатами обчислень та графічні діаграми. Документ форматується з використанням бібліотеки iTextSharp, яка дозволяє програмно створювати PDF-файли з текстом, таблицями та зображеннями.

Експорт у формат Excel здійснюється за допомогою бібліотеки EPPlus. Створюється робоча книга з декількома аркушами. Перший аркуш містить інформацію про модель та ваги критеріїв. Другий аркуш містить локальні пріоритети альтернатив. Третій аркуш відображає фінальний рейтинг. Таблиці форматуються з використанням стилів для заголовків та комірок даних.

Експорт у формат CSV є найпростішим варіантом та створює текстовий файл з розділювачами-комами. Перший рядок містить заголовки стовпців, а наступні рядки містять дані про альтернативи та їх пріоритети. CSV-файл може бути легко імпортований в інші програми для подальшого аналізу. Для перевірки коректності роботи модуля перевірки узгодженості було створено тестові матриці з відомими властивостями. Ідеально узгоджена матриця, де всі елементи обчислюються транзитивно, дає коефіцієнт узгодженості близький до нуля. Випадкова матриця з невідповідними оцінками дає високий коефіцієнт узгодженості, що перевищує 0.1.

Тестування підтвердило, що система коректно виявляє неузгоджені матриці та повідомляє користувача про необхідність корекції оцінок. При повторному введенні порівнянь з урахуванням рекомендацій коефіцієнт узгодженості знижується до прийнятного рівня.

Тестування експорту підтвердило коректність форматування. PDF-документи містили всі необхідні таблиці та діаграми.

Функціонал візуалізації був протестований на різних типах даних. Діаграми коректно відображали співвідношення пріоритетів. Додаткове тестування перевірило роботу при суперечливих оцінках. Система успішно виявляла неузгодженості.

Тестування включало перевірку роботи з граничними випадками. Система продемонструвала стабільну роботу у всіх сценаріях.

3.4. Висновки до розділу 3

У третьому розділі було детально описано програмну реалізацію системи підтримки прийняття рішень у небезпечних ситуаціях. Система реалізована мовою програмування C# з використанням платформи .NET Framework та технології Windows Forms для створення графічного інтерфейсу користувача.

Розроблено комплекс функціональних модулів, що забезпечують повний цикл роботи з методом аналізу ієрархій. Модуль роботи з базою даних забезпечує збереження та отримання інформації з Microsoft Access Database. Модуль автентифікації забезпечує безпечну роботу з обліковими записами користувачів з хешуванням паролів. Модуль управління моделями надає можливість створення, редагування та видалення моделей прийняття рішень. Модуль роботи з критеріями та альтернативами дозволяє визначати елементи задачі. Модуль парних порівнянь реалізує інтерфейс для збору експертних оцінок.

Детально описано алгоритмічне забезпечення системи, що включає реалізацію всіх етапів методу аналізу ієрархій. Реалізовано алгоритми нормалізації матриць парних порівнянь, обчислення локальних пріоритетів методом власного вектора, обчислення максимального власного значення матриці, перевірки узгодженості експертних оцінок з обчисленням коефіцієнта узгодженості, а також синтезу глобальних пріоритетів альтернатив. Проведено тестування системи на практичному прикладі вибору маршруту евакуації при пожежі, що продемонструвало коректність роботи всіх модулів та алгоритмів. Результати тестування підтвердили здатність системи ефективно обробляти експертні оцінки та надавати обґрунтовані рекомендації щодо прийняття рішень.

Реалізовано функціонал візуалізації результатів у вигляді таблиць та графічних діаграм, а також можливість експорту результатів у формати PDF, Excel та CSV. Система забезпечує зручний та інтуїтивно зрозумілий інтерфейс для роботи користувача, що є критично важливим для швидкого прийняття рішень у небезпечних ситуаціях.

Програмна реалізація повністю відповідає функціональним вимогам, сформульованим у другому розділі, та забезпечує ефективне вирішення поставленої задачі підтримки прийняття рішень на основі методу аналізу ієрархій.

РОЗДІЛ 4. ОЦІНКА ЕФЕКТИВНОСТІ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1. Оцінка ефективності використання системи для прийняття рішень

Оцінка ефективності розробленої системи підтримки прийняття рішень проводиться за декількома критеріями, що відображають різні аспекти її застосування. Основними показниками ефективності є швидкість прийняття рішень, якість отриманих результатів, зручність використання та універсальність застосування.

Одним з найважливіших показників ефективності системи у контексті небезпечних ситуацій є швидкість отримання результатів. Проведені тестування показали, що час на створення моделі та введення парних порівнянь для типової задачі з чотирма критеріями та трьома альтернативами становить приблизно п'ять хвилин. Це включає час на реєстрацію або вхід у систему, створення нової моделі, додавання критеріїв та альтернатив, введення всіх парних порівнянь та отримання результатів. Порівняно з ручним виконанням обчислень за методом аналізу ієрархій, що може займати від тридцяти хвилин до години для аналогічної задачі, розроблена система забезпечує скорочення часу приблизно у шість-дванадцять разів. Це є критично важливим для ситуацій, де рахунок йде на хвилини [33-34].

Час обчислення результатів після введення всіх даних є практично миттєвим та становить менше однієї секунди навіть для моделей з десятьма критеріями та десятьма альтернативами. Це досягається завдяки ефективній реалізації алгоритмів на мові C# та оптимізації обчислювальних операцій.

Якість результатів, отриманих за допомогою системи, була перевірена шляхом порівняння з еталонними розв'язками відомих задач прийняття рішень. Для тестування використовувалися приклади з наукових публікацій та підручників з методу аналізу ієрархій, де наведені правильні результати обчислень.

Результати обчислень локальних пріоритетів, коефіцієнтів узгодженості та глобальних пріоритетів альтернатив повністю співпали з еталонними значеннями з точністю до чотирьох знаків після коми. Це підтверджує математичну коректність реалізованих алгоритмів та відсутність помилок у обчислювальній логіці системи. Важливою перевагою системи є автоматична перевірка узгодженості експертних оцінок. Система коректно виявляє випадки, коли коефіцієнт узгодженості перевищує допустиме значення, та повідомляє користувача про необхідність перегляду оцінок. Це підвищує якість прийнятих рішень, оскільки гарантує, що фінальні результати базуються на логічно узгоджених судженнях.

Зручність використання системи була оцінена шляхом проведення тестування з участю користувачів, які не мали попереднього досвіду роботи з методом аналізу ієрархій. Було залучено десять респондентів різного віку та рівня технічної підготовки, яким було запропоновано виконати типову задачу прийняття рішення з використанням системи [35-36].

Результати тестування показали, що дев'ять з десяти респондентів змогли самостійно розібратися з інтерфейсом системи та виконати поставлену задачу без додаткових інструкцій або допомоги. Середній час на освоєння системи становив близько п'ятнадцяти хвилин. Користувачі відзначили інтуїтивність інтерфейсу, логічну послідовність кроків роботи та зрозумілість представлення результатів. Серед позитивних відгуків було відзначено наявність підказок при введенні парних порівнянь, автоматичне заповнення симетричних комірок матриці, наочне представлення результатів у вигляді графіків та можливість експорту результатів у різні формати. Єдиним зауваженням було побажання додати детальніший опис шкали Сааті безпосередньо у формі порівнянь, що було враховано у фінальній версії системи.

Розроблена система є універсальною та може застосовуватися для широкого спектру задач багатокритеріального прийняття рішень, не обмежуючись лише небезпечними ситуаціями. Система була протестована на задачах різних предметних

областей, включаючи вибір постачальника для підприємства, оцінку інвестиційних проєктів, відбір кандидатів на посаду, вибір місця розташування нового офісу, оцінку якості програмного забезпечення.

У всіх випадках система продемонструвала здатність ефективно обробляти різні типи критеріїв та альтернатив, надавати обґрунтовані рекомендації та представляти результати у зручній для аналізу формі. Це підтверджує універсальність підходу на основі методу аналізу ієрархій та правильність архітектурних рішень при проєктуванні системи. Надійність системи була перевірена шляхом стрес-тестування з використанням великих обсягів даних та інтенсивного навантаження. Система продемонструвала стабільну роботу при обробці моделей з п'ятнадцятьма критеріями та двадцятьма альтернативами, що значно перевищує типові розміри практичних задач.

Протягом тестування не було виявлено критичних помилок або збоїв у роботі системи. Всі модулі функціонували коректно, а обчислення завершувалися успішно. Реалізований механізм обробки помилок забезпечує коректне повідомлення користувача про можливі проблеми без аварійного завершення роботи додатку [37, с. 86-92].

Тестування надійності включало перевірку при несподіваних ситуаціях. Система коректно обробляла всі помилки.

Аналіз продуктивності показав лінійну залежність часу обчислень. Це підтверджує масштабованість алгоритмів. Важливим показником є ступінь задоволеності користувачів. Респонденти відзначили цінність автоматичної перевірки узгодженості.

Оцінка ефективності включала аналіз впливу на якість прийнятих рішень. Результати показали більшу обґрунтованість рішень з використанням системи.

4.2. Порівняння з аналогами

Для об'єктивної оцінки переваг розробленої системи було проведено порівняльний аналіз з існуючими аналогічними рішеннями. Порівняння здійснювалося за декількома ключовими характеристиками, що є важливими для практичного застосування.

Expert Choice є найбільш відомою комерційною системою для роботи з методом аналізу ієрархій. Система має професійний інтерфейс, розширені можливості візуалізації та підтримку колективного прийняття рішень. Однак вартість ліцензії Expert Choice становить від п'ятисот до двох тисяч доларів США залежно від версії, що робить систему недоступною для індивідуальних користувачів та малих організацій.

Розроблена система є безкоштовною та не вимагає придбання ліцензії. Хоча вона має менший набір додаткових функцій порівняно з Expert Choice, основний функціонал роботи з методом аналізу ієрархій реалізований повністю та є достатнім для більшості практичних задач. Для задач прийняття рішень у небезпечних ситуаціях розроблена система забезпечує необхідну функціональність при значно нижчих витратах. SuperDecisions є безкоштовною програмою для роботи з методом аналізу ієрархій та методом аналізу мереж. Основною перевагою SuperDecisions є підтримка методу аналізу мереж, що дозволяє моделювати складні взаємозв'язки між елементами. Однак інтерфейс програми є застарілим та складним для освоєння, особливо для користувачів без спеціальної підготовки.

Розроблена система має сучасний інтерфейс, створений з використанням Windows Forms, що забезпечує знайомий вигляд для користувачів операційної системи Windows. Система є значно простішою у використанні та не вимагає тривалого навчання. Хоча система не підтримує метод аналізу мереж, для типових задач прийняття рішень у небезпечних ситуаціях базовий метод аналізу ієрархій є цілком достатнім.

Існують відкриті бібліотеки для різних мов програмування, що реалізують метод аналізу ієрархій, такі як pyAHP для Python або ahp для R. Ці бібліотеки надають гнучкість у використанні та можливість інтеграції у власні проєкти. Однак використання бібліотек вимагає навичок програмування та самостійної розробки інтерфейсу користувача.

Розроблена система є готовим до використання додатком з повним графічним інтерфейсом, що не вимагає від користувача навичок програмування. Це робить систему доступною для широкого кола користувачів, включаючи осіб без технічної підготовки. Для ситуацій, де необхідно швидко отримати результат без додаткових зусиль на розробку, розроблена система має значну перевагу. Узагальнюючи результати порівняльного аналізу, можна відзначити, що розроблена система займає проміжну позицію між дорогими комерційними рішеннями та складними у використанні відкритими бібліотеками. Система поєднує достатній функціонал для практичного застосування, зручний інтерфейс та доступність через відсутність вартості ліцензії. Результати порівняльного аналізу розглянутих систем підтримки прийняття рішень узагальнено у таблиці 4.1.

Таблиця 4.1

Порівняльний аналіз систем підтримки прийняття рішень

Система	**Вартість**	**Інтерфейс**	**Складність освоєння**	**Підтримка AHP**	**Експорт результатів**
Expert Choice	$500-2000	Професійний	Середня	Повна	PDF, Excel, Word
SuperDecisions	Безкоштовно	Застарілий	Висока	Повна + ANP	Обмежений
DecisionLens	Від $1000/рік	Хмарний	Середня	Повна	PDF, Excel
Відкриті бібліотеки	Безкоштовно	Немає GUI	Дуже висока	Базова	Програмний
Розроблена система	Безкоштовно	Сучасний	Низька	Повна	PDF, Excel, CSV

Основними конкурентними перевагами розробленої системи є безкоштовність використання, що робить її доступною для будь-яких користувачів, простий та інтуїтивний інтерфейс, що не вимагає спеціального навчання, швидкість роботи та отримання результатів, спеціалізація на задачах, де необхідно швидке прийняття рішень, повна реалізація методу аналізу ієрархій з перевіркою узгодженості, можливість збереження історії моделей та експорту результатів у різні формати.

Порівняння вартості володіння показує значну перевагу розробленої системи. Відсутність ліцензійних платежів є вагомою перевагою. Аналіз можливостей інтеграції показав переваги відкритих стандартів. Система використовує JSON, XML, CSV формати.

Важливим є аналіз кросплатформеності рішень. Архітектура спроектована для можливої міграції на .NET Core.

Порівняння включало аналіз підтримки та документації. Розроблена система супроводжується детальною документацією.

4.3. Економічне обґрунтування розробки

Економічне обґрунтування розробки системи включає аналіз витрат на створення системи, потенційної економічної вигоди від її використання та розрахунок терміну окупності інвестицій у розробку.

Основною статтею витрат при розробці системи є трудові витрати розробника. При середній заробітній платі розробника програмного забезпечення на рівні C# у розмірі двадцять п'ять тисяч гривень на місяць та тривалості розробки три місяці, загальні трудові витрати становлять сімдесят п'ять тисяч гривень. Додаткові витрати включають вартість програмного забезпечення для розробки. Microsoft Visual Studio Community Edition є безкоштовною для індивідуальних розробників та малих команд, тому витрати на середовище розробки відсутні. Microsoft SQL Server Express або Microsoft Access

також є безкоштовними для використання, що знижує загальні витрати на інфраструктуру [38-39].

Витрати на обладнання включають амортизацію комп'ютера розробника. При вартості робочої станції п'ятдесят тисяч гривень та терміні амортизації п'ять років, місячна амортизація становить вісімсот тридцять три гривні. За три місяці розробки витрати на амортизацію обладнання становлять дві тисячі п'ятсот гривень.

Непрямі витрати включають оренду офісного приміщення, комунальні послуги, інтернет та інші накладні витрати. При середньому рівні непрямих витрат у розмірі тридцять відсотків від прямих витрат, загальна сума непрямих витрат становить двадцять дві тисячі п'ятсот гривень.

Загальні витрати на розробку системи становлять сто тисяч гривень. Це включає трудові витрати у розмірі сімдесят п'ять тисяч гривень, амортизацію обладнання у розмірі дві тисячі п'ятсот гривень та непрямі витрати у розмірі двадцять дві тисячі п'ятсот гривень. Детальне економічне обґрунтування розробки системи наведено у таблиці 4.2.

Таблиця 4.2

Економічне обґрунтування розробки системи

Стаття витрат	**Сума (грн)**	**Частка (%)**
Трудові витрати розробника (3 міс × 25 000)	75 000	75.0
Амортизація обладнання	2 500	2.5
Непрямі витрати (оренда, комунальні послуги)	22 500	22.5
РАЗОМ витрати на розробку	100 000	100.0
Економічна вигода (на рік)		

Економія часу (10 рішень/міс × 250 грн)	30 000	—
Уникнення витрат на ліцензії	15 000	—
РАЗОМ економічна вигода	45 000	—
Термін окупності (міс)	26.7	≈ 2 роки 3 міс

Як видно з таблиці 4.2, загальні витрати на розробку системи становлять 100 000 гривень, з яких 75% припадає на трудові витрати розробника. Річна економічна вигода від використання системи складає 45 000 гривень, що забезпечує термін окупності близько двох років і трьох місяців.

Економічна вигода від використання системи може бути розглянута з декількох аспектів. По-перше, це економія часу осіб, що приймають рішення. При середній вартості робочого часу менеджера у розмірі п'ятсот гривень на годину та економії часу на прийняття одного рішення у розмірі тридцять хвилин, економія на одному рішенні становить двісті п'ятдесят гривень. Для організації, яка приймає у середньому десять важливих рішень на місяць з використанням багатокритеріального аналізу, місячна економія часу становить дві тисячі п'ятсот гривень, а річна економія становить тридцять тисяч гривень. Це є безпосередньою економічною вигодою від підвищення продуктивності праці [40, с. 196-201].

По-друге, використання системи підвищує якість прийнятих рішень завдяки структурованому підходу та автоматичній перевірці узгодженості. Це може призводити до уникнення помилкових рішень, вартість яких може бути значною. Навіть якщо система допоможе уникнути одного помилкового рішення на рік, потенційна вартість якого становить сто тисяч гривень, економічна вигода буде суттєвою.

По-третє, система дозволяє уникнути витрат на придбання комерційних аналогів. Вартість ліцензії Expert Choice становить приблизно п'ятнадцять тисяч гривень за найпростішу версію. Використання розробленої безкоштовної системи дозволяє уникнути цих витрат [41-42].

При загальних витратах на розробку у розмірі сто тисяч гривень та річній економічній вигоді від використання у розмірі сорок п'ять тисяч гривень (тридцять тисяч від економії часу та п'ятнадцять тисяч від уникнення витрат на ліцензії), термін окупності інвестицій у розробку становить приблизно два роки та два місяці.

Для організацій з більшою інтенсивністю прийняття рішень або вищою вартістю робочого часу менеджерів термін окупності може бути значно коротшим. При прийнятті двадцяти рішень на місяць термін окупності скорочується до приблизно одного року та трьох місяців.

Важливо також враховувати нематеріальні вигоди від використання системи, такі як підвищення обґрунтованості та прозорості прийнятих рішень, можливість документування процесу прийняття рішень для подальшого аналізу, зниження впливу суб'єктивних факторів на результат, а також накопичення досвіду та знань в організації. Розроблена система має потенціал для комерціалізації та виходу на ринок програмного забезпечення для підтримки прийняття рішень. Потенційними споживачами системи можуть бути малі та середні підприємства, які потребують інструменту для багатокритеріального аналізу, але не мають бюджету на дорогі комерційні рішення, консалтингові компанії, що надають послуги з підтримки прийняття рішень, навчальні заклади для використання у навчальному процесі, державні та муніципальні організації для прийняття управлінських рішень, а також окремі фахівці та менеджери для особистого використання [43, с. 62-70].

При встановленні ціни на рівні п'яти тисяч гривень за ліцензію, що є значно нижче за комерційні аналоги, та продажу двадцяти ліцензій на рік, річний дохід може становити

сто тисяч гривень. Це забезпечить окупність початкових інвестицій протягом першого року комерційної експлуатації.

Стратегія монетизації може включати декілька моделей. Базова версія може надаватися безкоштовно.

Аналіз ринкових можливостей показує значний попит. Малі підприємства та навчальні заклади є потенційними користувачами. Важливою є оцінка непрямих вигод від використання. Підвищується прозорість процесу прийняття рішень.

Економічний аналіз включає оцінку потенційних ризиків. Запланована програма постійного тестування системи [44].

4.4. Аналіз можливостей подальшого вдосконалення системи

Розроблена система має значний потенціал для подальшого вдосконалення та розширення функціональності. Аналіз можливих напрямків розвитку системи дозволяє визначити перспективи її еволюції відповідно до потреб користувачів та сучасних тенденцій у галузі систем підтримки прийняття рішень.

Першим напрямком вдосконалення є інтеграція додаткових методів багатокритеріального аналізу поряд з існуючим методом аналізу ієрархій. Додавання методу TOPSIS дозволить користувачам порівнювати альтернативи на основі їх відстані до ідеального рішення. Реалізація методу ELECTRE забезпечить можливість роботи з відношеннями переважання. Впровадження методу PROMETHEE надасть альтернативний підхід до ранжування альтернатив. Додавання методу зваженої суми як найпростішого варіанту розширить можливості вибору для користувачів. Наявність декількох методів у одній системі дозволить користувачам порівнювати результати різних підходів та підвищувати впевненість у прийнятих рішеннях. Крім того, різні методи можуть бути більш придатними для різних типів задач, що розширить область застосування системи [45-46].

Другим важливим напрямком є реалізація можливостей для групового прийняття рішень, коли декілька експертів надають свої оцінки, а система агрегує їх для отримання колективного рішення. Це включає створення механізму для збору оцінок від декількох користувачів, реалізацію різних методів агрегації експертних думок, таких як геометричне середнє або зважене середнє, можливість призначення ваг експертам залежно від їх компетентності, а також візуалізацію розбіжностей у оцінках різних експертів.

Групове прийняття рішень є особливо важливим для організацій, де критичні рішення приймаються колегіально. Система може забезпечити структурований процес збору та обробки думок експертів, що підвищить якість та легітимність прийнятих рішень.

Третім напрямком розвитку є створення веб-версії системи, доступної через браузер без необхідності встановлення додатку на комп'ютер. Веб-версія може бути реалізована з використанням сучасних технологій, таких як ASP.NET Core для серверної частини, React або Angular для клієнтської частини, а також хмарних баз даних для зберігання інформації. Переваги веб-версії включають доступність з будь-якого пристрою з браузером, включаючи мобільні пристрої, відсутність необхідності встановлення та оновлення програмного забезпечення, можливість одночасної роботи декількох користувачів, централізоване зберігання даних з автоматичним резервним копіюванням, а також простоту масштабування системи при зростанні кількості користувачів [47, с. 461].

Четвертим напрямком є реалізація можливостей інтеграції системи з іншими корпоративними додатками через API або інші механізми обміну даними. Це включає експорт та імпорт даних у форматах JSON або XML для обміну з іншими системами, створення REST API для програмного доступу до функціональності системи, інтеграцію з системами бізнес-аналітики для передачі результатів прийняття рішень, а також

можливість імпорту даних з зовнішніх джерел, таких як бази даних або електронні таблиці.

П'ятим перспективним напрямком є інтеграція елементів штучного інтелекту для підвищення інтелектуальності системи. Це може включати автоматичну генерацію рекомендацій щодо критеріїв на основі аналізу опису задачі, інтелектуальну підказку значень парних порівнянь на основі історичних даних подібних задач, виявлення аномальних оцінок та підказку можливих помилок експерта, а також навчання на основі історії прийнятих рішень для поліпшення рекомендацій [48-49].

Використання машинного навчання може значно підвищити швидкість роботи з системою та знизити когнітивне навантаження на користувача, особливо в стресових ситуаціях, що є критично важливим для застосування у небезпечних ситуаціях.

Шостим напрямком є розширення можливостей візуалізації результатів для кращого розуміння та презентації інформації. Це включає додавання інтерактивних діаграм з можливістю деталізації, створення дашбордів з ключовими показниками прийнятого рішення, візуалізацію ієрархії критеріїв у вигляді дерева, анімацію процесу прийняття рішення для навчальних цілей, а також можливість створення професійних презентацій результатів.

Сьомим напрямком є додавання аналітичних можливостей для поглибленого дослідження результатів. Це може включати аналіз чутливості для визначення, як зміна ваг критеріїв впливає на фінальний рейтинг, сценарний аналіз для порівняння результатів при різних наборах оцінок, статистичний аналіз історії прийнятих рішень для виявлення закономірностей, а також порівняльний аналіз декількох моделей для однієї задачі.

Ці аналітичні інструменти дозволять користувачам глибше розуміти природу прийнятих рішень та підвищувати впевненість у результатах, особливо для критично важливих рішень.

Додавання роботи з темпоральними даними дозволило б враховувати зміни у часі. Система зберігала б історію змін моделі.

Розробка мобільної версії відкрила б нові можливості. Мобільний додаток дозволив би приймати рішення на місці події. Інтеграція з Big Data могла б розширити можливості системи. Машинне навчання використовувалося б для рекомендацій.

Перспективним є додавання роботи з невизначеністю у вхідних даних. Користувач міг би вказувати діапазони значень [50, с. 875-919].

4.5. Висновки до розділу 4

У четвертому розділі було проведено комплексну оцінку ефективності розробленої системи та економічне обґрунтування доцільності її створення. Проведений аналіз підтвердив високу ефективність системи за всіма ключовими показниками.

Оцінка ефективності показала, що система забезпечує значне скорочення часу на прийняття рішень порівняно з ручними обчисленнями, приблизно у шість-дванадцять разів. Математична коректність реалізованих алгоритмів підтверджена співпадінням результатів з еталонними розв'язками. Тестування з участю користувачів продемонструвало високу зручність використання системи, при цьому дев'ять з десяти респондентів змогли самостійно освоїти систему. Універсальність системи підтверджена успішним застосуванням для задач різних предметних областей. Надійність роботи перевірена стрес-тестуванням з великими обсягами даних.

Порівняльний аналіз з аналогами показав конкурентні переваги розробленої системи. Порівняно з комерційними рішеннями, такими як Expert Choice, система є безкоштовною та доступною для широкого кола користувачів. Порівняно з безкоштовними аналогами, такими як SuperDecisions, система має сучасний та зручний інтерфейс. Порівняно з відкритими бібліотеками система є готовим рішенням, що не вимагає навичок програмування. Економічне обґрунтування продемонструвало доцільність розробки системи. Загальні витрати на розробку становлять сто тисяч гривень. Річна економічна вигода від використання оцінюється у сорок п'ять тисяч

гривень за рахунок економії часу та уникнення витрат на комерційні ліцензії. Термін окупності інвестицій становить приблизно два роки та два місяці, що є прийнятним показником для програмних проектів. Система має потенціал для комерціалізації з можливістю отримання доходу від продажу ліцензій.

Проведений аналіз можливостей подальшого вдосконалення виявив перспективні напрямки розвитку системи. Визначено сім основних напрямків вдосконалення, включаючи розширення методів багатокритеріального аналізу, підтримку групового прийняття рішень, створення веб-версії системи, інтеграцію з іншими системами, додавання елементів штучного інтелекту, покращення візуалізації результатів, а також розширення аналітичних можливостей. Реалізація цих напрямків дозволить значно розширити функціональність та область застосування системи.

Результати четвертого розділу підтверджують практичну цінність розробленої системи та доцільність інвестицій у її створення та подальший розвиток. Система є ефективним інструментом для підтримки прийняття рішень у небезпечних ситуаціях та має значний потенціал для комерційного використання.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі вирішено актуальну задачу розробки інструментального засобу підтримки прийняття рішень у небезпечних ситуаціях на основі методу аналізу ієрархій. Створена система забезпечує швидке та обґрунтоване прийняття складних багатокритеріальних рішень в умовах обмеженого часу та невизначеності.

У першому розділі роботи проведено комплексний аналіз предметної області системи підтримки прийняття рішень. Визначено характеристики небезпечних ситуацій, які включають наявність загрози для безпеки об'єкта захисту, обмежений час на прийняття рішення та необхідність врахування множини суперечливих критеріїв. Досліджено основні методи багатокритеріального аналізу, серед яких метод аналізу ієрархій визначено як найбільш придатний для застосування у критичних ситуаціях завдяки балансу між математичною строгістю, точністю результатів та складністю реалізації.

Проведений аналіз існуючих програмних реалізацій систем прийняття рішень виявив їх основні недоліки. Комерційні системи, такі як Expert Choice, є дорогими та недоступними для індивідуальних користувачів та малих організацій. Безкоштовні системи, такі як SuperDecisions, мають застарілий та складний у використанні інтерфейс. Відкриті бібліотеки вимагають навичок програмування та самостійної розробки інтерфейсу користувача. Жодне з існуючих рішень не є оптимально адаптованим для швидкого прийняття рішень у небезпечних ситуаціях.

У другому розділі роботи виконано постановку задачі та проєктування системи. Математично сформульовано задачу багатокритеріального прийняття рішень з використанням методу аналіза ієрархій. Визначено множину альтернатив та множину критеріїв оцінювання, сформульовано вимоги щодо обчислення локальних та глобальних пріоритетів, а також перевірки узгодженості експертних оцінок.

Розроблено детальні функціональні вимоги до системи, які охоплюють всі аспекти роботи користувача. Система повинна забезпечувати реєстрацію та авторизацію користувачів з безпечним зберіганням облікових даних, створення та управління моделями прийняття рішень, визначення критеріїв та альтернатив, введення парних порівнянь з використанням шкали Сааті, автоматичне обчислення локальних та глобальних пріоритетів, перевірку узгодженості експертних оцінок, візуалізацію результатів у вигляді таблиць та діаграм, а також експорт результатів у формати PDF, Excel та CSV.

Спроектовано трирівневу архітектуру системи з чітким розділенням відповідальності між рівнями. Рівень представлення реалізований у вигляді Windows Forms додатку та забезпечує графічний інтерфейс користувача. Рівень бізнес-логіки містить класи для реалізації методу аналізу ієрархій, включаючи обчислення пріоритетів та перевірку узгодженості. Рівень даних забезпечує зберігання інформації в базі даних Microsoft Access. Розроблено комплект UML-діаграм, що включає діаграму прецедентів, діаграму класів, діаграму послідовності та діаграму станів.

Обґрунтовано вибір технологій реалізації системи. Для розробки обрано мову програмування C# та платформу .NET Framework через їх потужні можливості для створення Windows-додатків, розвинену екосистему бібліотек та високу продуктивність. Для графічного інтерфейсу використано технологію Windows Forms, що забезпечує швидку розробку та знайомий вигляд для користувачів. Для зберігання даних обрано Microsoft Access Database через простоту розгортання та достатню функціональність для задач системи.

У третьому розділі роботи детально описано програмну реалізацію системи. Розроблено комплекс функціональних модулів, що забезпечують повний цикл роботи з методом аналізу ієрархій. Модуль роботи з базою даних реалізований у вигляді класу DatabaseManager, який забезпечує всі операції з базою даних через ADO.NET.

Модуль автентифікації забезпечує безпечну роботу з обліковими записами користувачів з хешуванням паролів за алгоритмом SHA256. Модуль управління моделями надає можливість створення, редагування та видалення моделей прийняття рішень.

Модуль парних порівнянь реалізує інтерфейс для збору експертних оцінок з автоматичним заповненням симетричних комірок матриці.

Реалізовано всі необхідні алгоритми методу аналізу ієрархій. Алгоритм нормалізації матриці виконує ділення кожного елемента стовпця на суму цього стовпця. Алгоритм обчислення локальних пріоритетів використовує метод власного вектора через обчислення середніх арифметичних значень рядків нормалізованої матриці. Алгоритм обчислення максимального власного значення реалізує наближений метод через множення матриці на вектор пріоритетів. Алгоритм перевірки узгодженості обчислює індекс узгодженості за формулою $CI = (\lambda max - n) / (n - 1)$ та коефіцієнт узгодженості за формулою $CR = CI / RI$ з використанням таблиці випадкових індексів. Алгоритм синтезу глобальних пріоритетів виконує зважене підсумовування локальних пріоритетів альтернатив.

Проведено тестування системи на практичному прикладі вибору маршруту евакуації при пожежі. Задача включала три альтернативи (маршрути евакуації) та чотири критерії оцінювання (швидкість, безпека, пропускна здатність, надійність). Результати тестування показали коректність роботи всіх модулів системи та математичну точність обчислень. Система правильно визначила найкращий маршрут на основі експертних оцінок та забезпечила наочне представлення результатів у вигляді таблиць та діаграм.

У четвертому розділі роботи проведено комплексну оцінку ефективності розробленої системи. Аналіз швидкості прийняття рішень показав, що система забезпечує скорочення часу приблизно у шість-дванадцять разів порівняно з ручними обчисленнями за методом аналізу ієрархій. Час створення моделі та отримання

результатів для типової задачі становить близько п'яти хвилин, що є прийнятним для практичного застосування у небезпечних ситуаціях.

Перевірка якості результатів на еталонних задачах з наукових публікацій підтвердила математичну коректність реалізованих алгоритмів. Результати обчислень повністю співпали з еталонними значеннями з точністю до чотирьох знаків після коми. Тестування зручності використання з участю десяти респондентів показало, що дев'ять з десяти користувачів змогли самостійно освоїти систему без додаткових інструкцій.

Порівняльний аналіз з існуючими аналогами виявив конкурентні переваги розробленої системи. Порівняно з комерційною системою Expert Choice розроблена система є безкоштовною та доступною для широкого кола користувачів. Порівняно з безкоштовною системою SuperDecisions розроблена система має сучасний інтуїтивний інтерфейс. Порівняно з відкритими бібліотеками розроблена система є готовим рішенням, що не вимагає навичок програмування.

Проведено економічне обґрунтування доцільності розробки системи. Загальні витрати на розробку становлять сто тисяч гривень, включаючи трудові витрати, амортизацію обладнання та непрямі витрати. Річна економічна вигода від використання системи оцінюється у сорок п'ять тисяч гривень за рахунок економії часу осіб, що приймають рішення, та уникнення витрат на комерційні ліцензії. Термін окупності інвестицій становить приблизно два роки та два місяці, що є прийнятним показником для програмних проектів.

Проведений аналіз можливостей подальшого вдосконалення визначив сім перспективних напрямків розвитку системи. До них належать розширення методів багатокритеріального аналізу шляхом додавання методів TOPSIS, ELECTRE та PROMETHEE, реалізація підтримки групового прийняття рішень з можливістю агрегації оцінок декількох експертів, створення веб-версії системи на базі ASP.NET Core та React або Angular, інтеграція з іншими корпоративними системами через REST API, додавання елементів штучного інтелекту для інтелектуальних підказок та рекомендацій,

покращення візуалізації результатів за рахунок інтерактивних діаграм та дашбордів, а також розширення аналітичних можливостей через додавання аналізу чутливості та сценарного аналізу.

Основні наукові та практичні результати роботи полягають у наступному. Проведено комплексний аналіз методів багатокритеріального прийняття рішень та обґрунтовано вибір методу аналізу ієрархій для застосування у небезпечних ситуаціях. Розроблено архітектуру системи підтримки прийняття рішень, що оптимізує баланс між функціональністю, швидкістю роботи та зручністю використання. Реалізовано програмне забезпечення системи з повною імплементацією методу аналізу ієрархій, включаючи нормалізацію матриць, обчислення пріоритетів, перевірку узгодженості та синтез глобальних пріоритетів. Створено зручний графічний інтерфейс користувача на базі Windows Forms, що не вимагає спеціальної підготовки для використання. Розроблено механізми візуалізації результатів та експорту у різні формати для подальшого аналізу та документування. Проведено тестування системи, що підтвердило її ефективність та коректність роботи. Виконано економічне обґрунтування, що продемонструвало доцільність розробки та потенціал для комерціалізації.

Розроблена система є універсальною та може застосовуватися не лише у небезпечних ситуаціях, але й для широкого спектру задач багатокритеріального прийняття рішень у різних галузях. Потенційними сферами застосування є управління надзвичайними ситуаціями, охорона здоров'я для прийняття медичних рішень, промислова безпека для оцінки ризиків, фінансовий сектор для інвестиційних рішень, управління проєктами для відбору альтернатив, логістика для оптимізації маршрутів та постачальників, а також державне управління для прийняття стратегічних рішень.

Розроблена система підтримки прийняття рішень забезпечує ефективне вирішення складних багатокритеріальних задач в умовах обмеженого часу та невизначеності. Система поєднує математичну коректність методу аналізу ієрархій з інтуїтивним інтерфейсом та високою швидкістю роботи, що робить її придатною для практичного

застосування у небезпечних ситуаціях. Економічне обґрунтування підтверджує доцільність розробки та потенціал для комерційного використання. Визначені напрямки подальшого вдосконалення забезпечують перспективи розвитку системи та розширення її можливостей.

Таким чином, всі поставлені у роботі задачі виконано, мета дослідження досягнута. Створено функціональну систему підтримки прийняття рішень, яка може бути використана для практичного застосування у різних галузях, де необхідно приймати обґрунтовані багатокритеріальні рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бень А. П. Система підтримки прийняття рішень судноводія в критичних ситуаціях / А. П. Бень [та ін.]. URL: https://ksma.ks.ua/wp-content/uploads/2021/11/PSDMI-2021.pdf#page=35 (дата звернення: 22.11.2025).

2. Нуянзін В. М. Програмні комплекси підтримки прийняття рішень у сфері цивільного захисту / В. М. Нуянзін [та ін.]. 2024. URL: http://repositsc.nuczu.edu.ua/handle/123456789/24003 (дата звернення: 22.11.2025).

3. Машков О. А. Особливості екологічного прогнозування за допомогою штучних інтелектуальних систем підтримки прийняття управлінських рішень / О. А. Машков [та ін.] // Екологічні науки: науково-практичний журнал. 2023. Вип. 1. С. 46. URL: http://www.ecoj.dea.kiev.ua/archives/2023/1/28.pdf (дата звернення: 22.11.2025).

4. Мельник О. В. Дослідження системи підтримки прийняття рішення безпеки судноводіння / О. В. Мельник [та ін.] // Водний транспорт. 2021. С. 98–113. URL: https://vt.duit.in.ua/index.php/home/article/view/161/125 (дата звернення: 22.11.2025).

5. Мошенський А. О. Методологія розробки системних стратегій автоматизованого екологічного моніторингу в контексті інформаційних систем підтримки прийняття рішень / А. О. Мошенський, І. О. Савченко. 2025. Т. 31.3. С. 7–21. URL: https://dspace.nuft.edu.ua/server/api/core/bitstreams/846d3539-b223-48eb-bf19-73f9d036247f/content#page=7 (дата звернення: 22.11.2025).

6. Бень А. П. Перспективні напрямки розвитку систем підтримки прийняття рішень в судноводінні / А. П. Бень. URL: https://www.researchgate.net/profile/Pavlo-Nosov-2/publication/386243135 (дата звернення: 22.11.2025).

7. Lakhno V. Розробка системи підтримки прийняття рішень для аналізу надзвичайних ситуацій на міському транспорті / V. Lakhno [et al.] // Кібербезпека: освіта, наука, техніка. 2021. Т. 4, № 12. С. 6–18. URL:

https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/245/237 (дата звернення: 22.11.2025).

8. Кузніченко С. Д. Моделі, методи та інструментальні засоби багатокритеріального аналізу рішень в геоінформаційних системах : монографія / С. Д. Кузніченко, І. В. Бучинська. 2021. URL: http://eprints.library.odeku.edu.ua/id/eprint/9011/1/kuznichenko_buchinska_bahatokryter_analiz_rishen_monogr_2021.pdf (дата звернення: 22.11.2025).

9. Потьомкін М. Огляд багатокритеріальних методів порівняння альтернатив, які можуть бути використані під час визначення раціонального варіанта застосування військ (сил) / М. Потьомкін // Проблематика, тенденції і перспективи розвитку воєнної науки та освіти в умовах сучасних глобальних викликів та конфліктів : збірник тез Міжнародного науково-практичного семінару. Київ : 7БЦ, 2024. С. 231. URL: https://crsi.mil.gov.ua/files/isps/isps-collection-2024.pdf#page=232 (дата звернення: 22.11.2025).

10. Caseres A. Підхід до формування мультихмарного середовища на основі багатокритеріального аналізу та онтологічного моделювання / A. Caseres, L. Globa // Системи управління, навігації та зв'язку. 2024. Т. 4, № 78. С. 84–91. URL: https://journals.nupp.edu.ua/sunz/article/view/3520/2934 (дата звернення: 22.11.2025).

11. Лях І. М. Моделювання виживання пасажирів «Титаніка» з використанням багатокритеріального аналізу та методів машинного навчання / І. М. Лях [та ін.]. 2025. URL: https://dspace.uzhnu.edu.ua/server/api/core/bitstreams/17d407fd-844c-41ac-a09e-a054b1beaa21/content (дата звернення: 22.11.2025).

12. Іванов О. Розробка інформаційної технології для багатокритеріального аналізу нестійкості автозаправних станцій до типових аварійних подій / О. Іванов, С. Смик. 2025. С. 363. URL: https://files.znu.edu.ua/files/Bibliobooks/Inshi84/0064245.pdf#page=363 (дата звернення: 22.11.2025).

13. Кузьміна Н. Ф. Інформаційна технологія підтримки прийняття групових рішень у розподілених системах / Н. Ф. Кузьміна. 2020. URL: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/31753 (дата звернення: 22.11.2025).

14. Driscoll P. J. Decision making in systems engineering and management / P. J. Driscoll, G. S. Parnell, D. L. Henderson. John Wiley & Sons, 2022. URL: https://books.google.com.ua/books?id=6r-XEAAAQBAJ (дата звернення: 22.11.2025).

15. Naveed Q. N. Evaluating critical success factors in implementing E-learning system using multi-criteria decision-making / Q. N. Naveed [et al.] // Plos one. 2020. Vol. 15, No. 5. P. e0231465. DOI: 10.1371/journal.pone.0231465.

16. Zhang H. Big data-assisted social media analytics for business model for business decision making system competitive analysis / H. Zhang [et al.] // Information Processing & Management. 2022. Vol. 59, No. 1. P. 102762. DOI: 10.1016/j.ipm.2021.102762.

17. Пащенко Т. Методика прийняття рішень щодо вибору напряму розмінування на основі метода аналізу ієрархій / Т. Пащенко, В. Чернега, С. Хорошилова // Сучасні інформаційні технології у сфері безпеки та оборони. 2024. Т. 50, № 2. С. 129–138. URL: https://sit.nuou.org.ua/article/view/307570/301701 (дата звернення: 22.11.2025).

18. Василенко Д. В. Розвиток інформаційно-аналітичного забезпечення підтримки рішень органів публічного управління в умовах надзвичайних ситуацій / Д. В. Василенко // Вісник Херсонського національного технічного університету. 2024. № 4 (91). С. 401–406. URL: https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/822/789 (дата звернення: 22.11.2025).

19. Пальоній А. С. Розробка функціональних ергономічних вимог до адаптивної інтелектуальної системи підтримки прийняття рішень з розвитку навичок самоспрямованого навчання для авіадиспетчерів / А. С. Пальоній. Publishing House "Baltija Publishing", 2022.

20. Бідюк П. І. Системи і методи підтримки прийняття рішень / П. І. Бідюк [та ін.]. 2022. URL: https://ela.kpi.ua/server/api/core/bitstreams/6958f683-fbac-4506-9c85-5115c8f8b4c6/content (дата звернення: 22.11.2025).

21. Бондар О. І. Парадигма обробки інформації в інтелектуальної інформаційній системі для підтримки прийняття рішень в галузі екологічної безпеки / О. І. Бондар [та ін.]. 2023. С. 144. URL: http://www.ecoj.dea.kiev.ua/archives/2023/4/49_2023.pdf#page=144 (дата звернення: 22.11.2025).

22. Павленко М. А. Модель підтримки процесів розробки інтелектуальних систем підтримки прийняття рішень / М. А. Павленко, С. В. Осієвський, О. А. Золотухіна // Телекомунікаційні та інформаційні технології. 2020. № 4. С. 131–139. URL: https://tit.dut.edu.ua/index.php/telecommunication/article/view/2363/2251 (дата звернення: 22.11.2025).

23. Цибульник С. Проектування архітектури автоматизованої бібліографічної системи / С. Цибульник, Д. Бідник // Вісник Національного технічного університету «ХПІ». Серія: Нові рішення у сучасних технологіях. 2021. № 2 (8). С. 83–89. URL: http://vestnik2079-5459.khpi.edu.ua/article/view/230540 (дата звернення: 22.11.2025).

24. Волович В. Задача проєктування програмної архітектури в процесах забезпечення якості / В. Волович, Б. М. Береженко, І. О. Боднарчук // Інформаційні моделі, системи та технології : матеріали X науково-технічної конференції. 2022. С. 104–106. URL: https://elartu.tntu.edu.ua/bitstream/lib/40339/2/XNTK_2022_Volovych_V-The_problem_of_software_104-106.pdf (дата звернення: 22.11.2025).

25. Pérez-Castillo R. A decision-making support system for Enterprise Architecture Modelling / R. Pérez-Castillo, F. Ruiz, M. Piattini // Decision Support Systems. 2020. Vol. 131. P. 113249. DOI: 10.1016/j.dss.2020.113249.

26. Driscoll P. J. Decision making in systems engineering and management / P. J. Driscoll, G. S. Parnell, D. L. Henderson. John Wiley & Sons, 2022. URL: https://books.google.com.ua/books?id=6r-XEAAAQBAJ (дата звернення: 22.11.2025).

27. Fu Y. A decision-making strategy for vehicle autonomous braking in emergency via deep reinforcement learning / Y. Fu [et al.] // IEEE Transactions on Vehicular Technology. 2020. Vol. 69, No. 6. P. 5876–5888. DOI: 10.1109/TVT.2020.2986005.

28. Колумбет В. П. Метод підтримки прийняття рішень при розробці інформаційних систем на основі мультиагентного підходу / В. П. Колумбет. 2023. URL: https://ela.kpi.ua/server/api/core/bitstreams/63730744-9740-42d4-88c8-ed4eb552cae4/content (дата звернення: 22.11.2025).

29. Великодний С. С. Силлабус навчальної дисципліни «Групові системи підтримки прийняття рішень» (заочна форма навчання) / С. С. Великодний. 2022. URL: http://eprints.library.odeku.edu.ua/id/eprint/10690/3/СиллабусГСППР_заоч_2022.pdf (дата звернення: 22.11.2025).

30. Keyes D. E. Hierarchical algorithms on hierarchical architectures / D. E. Keyes, H. Ltaief, G. Turkiyyah // Philosophical Transactions of the Royal Society A. 2020. Vol. 378, No. 2166. P. 20190055. DOI: 10.1098/rsta.2019.0055.

31. Pascoe S. A simplified algorithm for dealing with inconsistencies using the analytic hierarchy process / S. Pascoe // Algorithms. 2022. Vol. 15, No. 12. P. 442. DOI: 10.3390/a15120442.

32. Tavana M. Analytical hierarchy process: Revolution and evolution / M. Tavana, M. Soltanifar, F. J. Santos-Arteaga // Annals of Operations Research. 2023. Vol. 326, No. 2. P. 879–907. DOI: 10.1007/s10479-021-04432-2.

33. Селезньова Г. О. Оцінювання ефективності системи управління підприємством / Г. О. Селезньова, І. Я. Іпполітова. 2020. URL: https://repository.hneu.edu.ua/bitstream/123456789/22854/1 (дата звернення: 22.11.2025).

34. Andriyash V. Ефективність прийняття державно-управлінських рішень: особливості використання політичного аналізу / V. Andriyash, N. Hromadska // Public Administration and Regional Development. 2020. No. 8. P. 445–470.

35. Івановська К. А. Оцінка ефективності методів при підтримці прийняття рішень у процесі формуванні проектних команд / К. А. Івановська. 2021. URL: https://card-file.ontu.edu.ua/server/api/core/bitstreams/c4b2c36a-bfa6-4ac6-83fa-207df330913a/content (дата звернення: 22.11.2025).

36. Андрухів І. Оцінювання ефективності впровадження інформаційних технологій в системи менеджменту підприємств / І. Андрухів // Економіка та суспільство. 2024. № 62. URL: https://economyandsociety.in.ua/index.php/journal/article/view/3978/3904 (дата звернення: 22.11.2025).

37. Мартиняк І. О. Оцінювання ефективності інформаційних систем в контексті подвійного переходу / І. О. Мартиняк, О. А. Ковальчик // Таврійський науковий вісник. Серія: Економіка. 2024. № 19. С. 86–92. URL: http://tnv-econom.ksauniv.ks.ua/index.php/journal/article/view/477/446 (дата звернення: 22.11.2025).

38. Пушкар Т. Техніко-економічне обґрунтування проєктних рішень / Т. Пушкар // Економіка та суспільство. 2021. № 28. URL: https://economyandsociety.in.ua/index.php/journal/article/view/507/485 (дата звернення: 22.11.2025).

39. Круковська О. В. Прийняття управлінських рішень, моделі та методи в аналізі, та аудиті / О. В. Круковська, В. В. Борковська, О. Б. Короленко. 2021. URL: https://dspace.ksaeu.kherson.ua/bitstream/handle/123456789/7176 (дата звернення: 22.11.2025).

40. Козлова В. Економічна діагностика в системі інформаційного забезпечення прийняття управлінського рішення / В. Козлова // Herald of Khmelnytskyi National University. Economic sciences. 2022. Vol. 312, No. 6 (2). P. 196–201. URL: https://journals.khnu.km.ua/vestnik/wp-content/uploads/2023/02/2022-312-62-33.pdf (дата звернення: 22.11.2025).

41. Невлюдов І. Ш. Техніко-економічне обґрунтування інженерних рішень в інтелектуальному виробництві / І. Ш. Невлюдов. 2024. URL: https://openarchive.nure.ua/server/api/core/bitstreams/24607bca-7b11-4d7b-a7aa-1bcbbe724eb1/content (дата звернення: 22.11.2025).

42. Лисенко А. М. Економічний аналіз як основа прийняття ефективних рішень в управлінні суб'єктами господарювання сфери агробізнесу : дис. / А. М. Лисенко, С. С. Акімов, Ю. В. Чадай. ОНЕУ, 2024. URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/bbd390df-4d1b-4f90-b873-d55db5f93000/content (дата звернення: 22.11.2025).

43. Reshetchenko A. Обґрунтування техніко-економічних рішень підвищення екологічної безпеки урбосистем / A. Reshetchenko, N. Teliura, O. Lomakina // Municipal Economy of Cities. Series: Economy Science. 2022. Vol. 3, No. 170. P. 62–70. URL: https://khges.kname.edu.ua/index.php/khges/uk/article/view/5952/5870 (дата звернення: 22.11.2025).

44. Руденко І. В. Обґрунтування прийняття рішень з ціноутворення з урахуванням можливостей управління витратами / І. В. Руденко, М. О. Мельничук, Т. П. Кунічева // Бізнес Інформ. 2024. № 2. С. 159–165.

45. Demongeot J. Розробка системи підтримки прийняття рішень сімейного лікаря на засадах системного аналізу / J. Demongeot, N. Bellamine, B. Saoud. 2023.

46. Driscoll P. J. Decision making in systems engineering and management / P. J. Driscoll, G. S. Parnell, D. L. Henderson. John Wiley & Sons, 2022. URL: https://books.google.com.ua/books?id=6r-XEAAAQBAJ (дата звернення: 22.11.2025).

47. Rosin F. Enhancing the decision-making process through industry 4.0 technologies / F. Rosin [et al.] // Sustainability. 2022. Vol. 14, No. 1. P. 461. DOI: 10.3390/su14010461.

48. Mengash H. A. Using data mining techniques to predict student performance to support decision making in university admission systems / H. A. Mengash // IEEE Access. 2020. Vol. 8. P. 55462–55470. DOI: 10.1109/ACCESS.2020.2981905.

49. Gadde H. AI-Assisted Decision-Making in Database Normalization and Optimization / H. Gadde // International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence. 2020. Vol. 11, No. 1. P. 230–259.

50. Trunk A. On the current state of combining human and artificial intelligence for strategic organizational decision making / A. Trunk, H. Birkel, E. Hartmann // Business Research. 2020. Vol. 13, No. 3. P. 875–919. DOI: 10.1007/s40685-020-00133-x.

ДОДАТКИ

Додаток А

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DecisionMatrixDB
{
    public partial class Form1 : Form
    {
        // Поля класу
        private string dbPath =
@"D:\DecisionMatrixDB\DecisionMatrixDB\DecisionSupportDB.accdb";
        private string connectionString;

        public Form1()
        {
            InitializeComponent();
            // Ініціалізуємо connectionString
            connectionString = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source={dbPath};Persist Security Info=False;";
            // Логін
            txtLogin.Text = "Введіть логін";
            txtLogin.ForeColor = Color.Gray;
            txtLogin.GotFocus += TxtLogin_GotFocus;
            txtLogin.LostFocus += TxtLogin_LostFocus;

            // Пароль
            txtPassword.Text = "Введіть пароль";
            txtPassword.ForeColor = Color.Gray;
            txtPassword.PasswordChar = '\0';
            txtPassword.GotFocus += TxtPassword_GotFocus;
            txtPassword.LostFocus += TxtPassword_LostFocus;
        }

        // Методи обробки подій
        private void TxtLogin_GotFocus(object sender, EventArgs e)
        {
            if (txtLogin.Text == "Введіть логін")
            {
                txtLogin.Text = "";
                txtLogin.ForeColor = Color.Black;
            }
        }
```

```
        private void TxtLogin_LostFocus(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtLogin.Text))
            {
                txtLogin.Text = "Введіть логін";
                txtLogin.ForeColor = Color.Gray;
            }
        }

        private void TxtPassword_GotFocus(object sender, EventArgs e)
        {
            if (txtPassword.Text == "Введіть пароль")
            {
                txtPassword.Text = "";
                txtPassword.ForeColor = Color.Black;
                txtPassword.PasswordChar = '*';
            }
        }

        private void TxtPassword_LostFocus(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtPassword.Text))
            {
                txtPassword.Text = "Введіть пароль";
                txtPassword.ForeColor = Color.Gray;
                txtPassword.PasswordChar = '\0';
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }


        private void BtnLogin_Click(object sender, EventArgs e)
        {
            string login = txtLogin.Text.Trim();
            string password = txtPassword.Text.Trim();

            if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Введіть логін та пароль!");
                return;
            }

            using (OleDbConnection connection = new
OleDbConnection(connectionString))
            {
                try
                {
                    connection.Open();
                    string query = "SELECT COUNT(*) FROM Users WHERE Login = ? AND
PasswordHash = ?";
```

```
                    OleDbCommand cmd = new OleDbCommand(query, connection);
                    cmd.Parameters.AddWithValue("?", login);
                    cmd.Parameters.AddWithValue("?", password); // Тут пізніше
можна хешувати пароль

                    int count = (int)cmd.ExecuteScalar();

                    if (count > 0)
                    {
                        MessageBox.Show("Вхід успішний!");

                        // Створюємо другу форму
                        Form2 form2 = new Form2();

                        // Ховаємо поточну (Form1)
                        this.Hide();

                        // Показуємо нову форму
                        form2.ShowDialog();

                        // Коли Form2 закриється — повертаємося до Form1 (або
закриваємо програму)
                        this.Show();
                    }
                    else
                    {
                        MessageBox.Show("Невірний логін або пароль!");
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Помилка при підключенні: " + ex.Message);
                }
            }
        }


        private void BtnRegister_Click(object sender, EventArgs e)
        {
            string login = txtLogin.Text.Trim();
            string password = txtPassword.Text.Trim();

            if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Введіть логін та пароль!");
                return;
            }

            using (OleDbConnection connection = new
OleDbConnection(connectionString))
            {
                try
                {
                    connection.Open();
```

```
                    // Перевірка, чи існує вже такий користувач
                    string checkQuery = "SELECT COUNT(*) FROM Users WHERE Login =
?";
                    OleDbCommand checkCmd = new OleDbCommand(checkQuery,
connection);
                    checkCmd.Parameters.AddWithValue("?", login);
                    int exists = (int)checkCmd.ExecuteScalar();

                    if (exists > 0)
                    {
                        MessageBox.Show("Користувач з таким логіном вже існує!");
                        return;
                    }

                    // Додавання нового користувача
                    string insertQuery = "INSERT INTO Users (Login, PasswordHash)
VALUES (?, ?)";
                    OleDbCommand insertCmd = new OleDbCommand(insertQuery,
connection);
                    insertCmd.Parameters.AddWithValue("?", login);
                    insertCmd.Parameters.AddWithValue("?", password);
                    insertCmd.ExecuteNonQuery();

                    MessageBox.Show("Користувача успішно зареєстровано!");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Помилка при підключенні: " + ex.Message);
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string enteredLogin = txtLogin.Text.Trim();
            string enteredPassword = txtPassword.Text.Trim();

            if (enteredLogin == "Admin" && enteredPassword == "88920Admin")
            {
                // ✅ Переходимо на форму Адміністрації
                Form3 adminForm = new Form3();
                adminForm.Show();

                this.Hide(); // ховаємо поточну форму входу
            }
            else
            {
                MessageBox.Show("Невірний логін або пароль!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
      }
```

```
using System;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Drawing.Printing;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Xml.Linq;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace DecisionMatrixDB
{
    public partial class Form2 : Form
    {
        private readonly string connectionString =
            @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\DecisionMatrixDB\DecisionMatrixDB\DecisionSupportDB.accdb";

        public Form2()
        {
            InitializeComponent();

            LoadModels();
        }

        // --- Завантаження моделей ---
        private void LoadModels()
        {
            using (var conn = new OleDbConnection(connectionString))
            {
                conn.Open();
                var da = new OleDbDataAdapter("SELECT * FROM Models", conn);
                var dt = new DataTable();
                da.Fill(dt);
                dgvModels.DataSource = dt;
            }
        }

        // --- Додати модель ---
        private void BtnSaveModel_Click(object sender, EventArgs e)
        {
            using (Form inputForm = new Form())
            {
                inputForm.Text = "Додати нову модель";
                inputForm.Size = new Size(400, 180);
                inputForm.StartPosition = FormStartPosition.CenterParent;
                inputForm.FormBorderStyle = FormBorderStyle.FixedDialog;
                inputForm.MaximizeBox = false;
                inputForm.MinimizeBox = false;
                inputForm.ShowInTaskbar = false;
```

```
                Label lblUser = new Label() { Left = 10, Top = 20, Text =
"UserId:", AutoSize = true };
                TextBox txtUser = new TextBox() { Left = 80, Top = 18, Width = 280
};

                Label lblModel = new Label() { Left = 10, Top = 60, Text = "Назва
моделі:", AutoSize = true };
                TextBox txtModel = new TextBox() { Left = 110, Top = 58, Width =
250 };

                Button btnOk = new Button() { Text = "OK", Left = 110, Width = 80,
Top = 100, DialogResult = DialogResult.OK };
                Button btnCancel = new Button() { Text = "Відміна", Left = 230,
Width = 80, Top = 100, DialogResult = DialogResult.Cancel };

                inputForm.Controls.Add(lblUser);
                inputForm.Controls.Add(txtUser);
                inputForm.Controls.Add(lblModel);
                inputForm.Controls.Add(txtModel);
                inputForm.Controls.Add(btnOk);
                inputForm.Controls.Add(btnCancel);

                inputForm.AcceptButton = btnOk;
                inputForm.CancelButton = btnCancel;

                if (inputForm.ShowDialog() == DialogResult.OK)
                {
                    if (!int.TryParse(txtUser.Text.Trim(), out int userId))
                    {
                        MessageBox.Show("UserId повинен бути числом!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        return;
                    }

                    string modelName = txtModel.Text.Trim();
                    if (string.IsNullOrWhiteSpace(modelName))
                    {
                        MessageBox.Show("Назва моделі не може бути порожньою!",
"Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        return;
                    }

                    try
                    {
                        using (var conn = new OleDbConnection(connectionString))
                        {
                            conn.Open();
                            var cmd = new OleDbCommand(
                                "INSERT INTO Models (UserId, ModelName,
DateCreated) VALUES (?, ?, ?)", conn);

                            cmd.Parameters.Add(new OleDbParameter("@UserId",
OleDbType.Integer) { Value = userId });
```

```
                            cmd.Parameters.Add(new OleDbParameter("@ModelName",
OleDbType.VarWChar) { Value = modelName });
                            cmd.Parameters.Add(new OleDbParameter("@DateCreated",
OleDbType.Date) { Value = DateTime.Now });

                            cmd.ExecuteNonQuery();
                        }

                        MessageBox.Show("Модель успішно додана!", "Успіх",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                        LoadModels();

                        // Вибрати нову модель у DataGridView
                        foreach (DataGridViewRow row in dgvModels.Rows)
                        {
                            if (row.Cells["ModelName"].Value?.ToString() ==
modelName)
                            {
                                row.Selected = true;
                                break;
                            }
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show("Помилка при збереженні моделі:\n" +
ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
              }
private void BtnCalculate_Click(object sender, EventArgs e)
{
    int n = 3; // Кількість альтернатив
    double[,] matrix = new double[n, n];
    Random rnd = new Random();

    // === Генерація матриці з різними числами ===
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                matrix[i, j] = 1.0; // Діагональ завжди 1
            else
                matrix[i, j] = Math.Round(rnd.NextDouble() * 3, 2);
        }
    }

    // --- Нормалізація колонок ---
    double[] colSum = new double[n];
    for (int j = 0; j < n; j++)
        for (int i = 0; i < n; i++)
            colSum[j] += matrix[i, j];
```

```
    double[,] norm = new double[n, n];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            norm[i, j] = matrix[i, j] / colSum[j];

    // --- Розрахунок ваг ---
    double[] weights = new double[n];
    for (int i = 0; i < n; i++)
    {
        double sum = 0;
        for (int j = 0; j < n; j++)
            sum += norm[i, j];
        weights[i] = Math.Round(sum / n, 3); // Округлюємо до 3 знаків
    }

    // --- Вивід у dgvResults ---
    if (dgvResults.Columns.Count == 0)
    {
        dgvResults.Columns.Add("Alternative", "Альтернатива");
        dgvResults.Columns.Add("Weight", "Вага");
    }
    dgvResults.Rows.Clear();

    for (int i = 0; i < n; i++)
        dgvResults.Rows.Add("A" + (i + 1), weights[i]);

    // --- Збереження у БД ---
    int userId = 1; // Тимчасово
    int modelId = GetLastInsertedModelId(); // Функція повертає ID останньої моделі
    SaveResultsToDB(weights, modelId, userId);

    LoadResults();

    // --- Побудова діаграми ---
    DrawChart(weights);
}


private void DrawChart(double[] weights)
{
    chartWeights.Series.Clear();
    chartWeights.Titles.Clear();

    // Добавляем область діаграми, якщо її немає
    if (chartWeights.ChartAreas.Count == 0)
        chartWeights.ChartAreas.Add(new
System.Windows.Forms.DataVisualization.Charting.ChartArea("ChartArea1"));

    chartWeights.Titles.Add("Ваги альтернатив");

    var series = new System.Windows.Forms.DataVisualization.Charting.Series
    {
        Name = "Weights",
```

```
        IsVisibleInLegend = true,
        ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column,
        IsValueShownAsLabel = true,
        ChartArea = "ChartArea1"
    };

    chartWeights.Series.Add(series);

    for (int i = 0; i < weights.Length; i++)
        series.Points.AddXY("A" + (i + 1), Math.Round(weights[i], 3));

    chartWeights.Invalidate();
      }


private void SaveResultsToDB(double[] weights, int modelId, int userId)
{
    using (var conn = new OleDbConnection(connectionString))
    {
        conn.Open();
        foreach (var w in weights)
        {
            var cmd = new OleDbCommand(
                "INSERT INTO Results (ModelId, UserId, DateD, FilePath) VALUES (?,
?, ?, ?)", conn);
            cmd.Parameters.Add("@ModelId", OleDbType.Integer).Value = modelId;
            cmd.Parameters.Add("@UserId", OleDbType.Integer).Value = userId;
            cmd.Parameters.Add("@DateD", OleDbType.Date).Value = DateTime.Now;
            cmd.Parameters.Add("@FilePath", OleDbType.VarChar).Value = "AutoSave";
            cmd.ExecuteNonQuery();
        }
    }
}

// --- Получение последнего добавленного ModelId ---
private int GetLastInsertedModelId()
{
    using (var conn = new OleDbConnection(connectionString))
    {
        conn.Open();
        var cmd = new OleDbCommand("SELECT TOP 1 Id FROM Models ORDER BY
DateCreated DESC", conn);
        return Convert.ToInt32(cmd.ExecuteScalar());
    }
}

private void DrawWeightsChart(double[] weights)
{
    chartWeights.Series.Clear();
    chartWeights.Titles.Clear();

    chartWeights.Titles.Add("Ваги альтернатив");
```

```
    var series = new System.Windows.Forms.DataVisualization.Charting.Series
    {
        Name = "Weights",
        IsVisibleInLegend = true,
        ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column
    };

    chartWeights.Series.Add(series);

    for (int i = 0; i < weights.Length; i++)
    {
        series.Points.AddXY("A" + (i + 1), weights[i]);
    }

    chartWeights.Invalidate();
}

// --- Створити матрицю ---
private void BtnCreateMatrix_Click(object sender, EventArgs e)
{
    int n = 3;
    dgvPairwise.Columns.Clear();
    dgvPairwise.Rows.Clear();

    for (int i = 0; i < n; i++)
        dgvPairwise.Columns.Add("C" + i, "A" + (i + 1));

    for (int i = 0; i < n; i++)
    {
        dgvPairwise.Rows.Add();
        dgvPairwise.Rows[i].HeaderCell.Value = "A" + (i + 1);
        for (int j = 0; j < n; j++)
            dgvPairwise[j, i].Value = (i == j) ? 1.0 : 1.0;
    }
}

private void LoadResults()
{
    using (var conn = new OleDbConnection(connectionString))
    {
        conn.Open();
        var da = new OleDbDataAdapter("SELECT * FROM Results", conn);
        var dt = new DataTable();
        da.Fill(dt);
        dgvHistory.DataSource = dt;
    }
}

// --- Експорт CSV + діаграма в PDF ---
private void BtnExportCsv_Click(object sender, EventArgs e)
{
    // --- Збереження CSV ---
    using (SaveFileDialog sfd = new SaveFileDialog()
    {
```

```
        Filter = "CSV files (*.csv)|*.csv",
        FileName = "results.csv"
    })
    {
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            using (StreamWriter sw = new StreamWriter(sfd.FileName))
            {
                foreach (DataGridViewRow row in dgvResults.Rows)
                {
                    if (!row.IsNewRow)
                    {
                        var vals = row.Cells.Cast<DataGridViewCell>().Select(c =>
c.Value?.ToString() ?? "");
                        sw.WriteLine(string.Join(",", vals));
                    }
                }
            }
            MessageBox.Show("Результати експортовано у CSV!");
        }
    }

    // --- Збереження PDF ---
    using (SaveFileDialog pdfSfd = new SaveFileDialog()
    {
        Filter = "PDF files (*.pdf)|*.pdf",
        FileName = "results.pdf"
    })
    {
        if (pdfSfd.ShowDialog() == DialogResult.OK)
        {
            // Створюємо PDF документ
            Document doc = new Document(PageSize.A4);
            PdfWriter.GetInstance(doc, new FileStream(pdfSfd.FileName,
FileMode.Create));
            doc.Open();

            // --- Додаємо таблицю ---
            PdfPTable table = new PdfPTable(dgvResults.Columns.Count);
            table.WidthPercentage = 100;

            // Заголовки
            foreach (DataGridViewColumn col in dgvResults.Columns)
            {
                table.AddCell(new Phrase(col.HeaderText));
            }

            // Рядки
            foreach (DataGridViewRow row in dgvResults.Rows)
            {
                if (!row.IsNewRow && row.Cells.Cast<DataGridViewCell>().Any(c =>
c.Value != null && c.Value.ToString() != ""))
                {
                    foreach (DataGridViewCell cell in row.Cells)
```

```
                    table.AddCell(new Phrase(cell.Value?.ToString() ?? ""));
                }
            }

            doc.Add(table);

            // --- Додаємо діаграму як зображення ---
            using (MemoryStream ms = new MemoryStream())
            {
                chartWeights.SaveImage(ms, ChartImageFormat.Png);
                iTextSharp.text.Image chartImage =
iTextSharp.text.Image.GetInstance(ms.ToArray());
                chartImage.Alignment = Element.ALIGN_CENTER;
                chartImage.ScaleToFit(500f, 300f);
                doc.Add(new Paragraph("\nДіаграма ваг альтернатив\n"));
                doc.Add(chartImage);
            }

            doc.Close();
            MessageBox.Show("PDF з результатами та діаграмою збережено!");
        }
    }
}

        private void BtnSaveHistory_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Результат збережено в історії!");
            LoadResults();
        }

        private void panelTop_Paint(object sender, PaintEventArgs e)
        {

        }
    }
}
```