МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
НА ТЕМУ
Система вибору контрагентів для ремонтних служб енергетичної інфраструктури

Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Аналітика даних»
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД- 41

_____Горелик А.Д.
(прізвище та ініціали)

Керівник ___Самохвалов Ю.Я_____
(прізвище та ініціали)
____доктор технічних наук, професор_
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри інтелектуальних технологій
Протокол № 13 від 05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ-2023

Анотація

Горелик Андрій Дмитрович виконав випускну кваліфікаційну роботу на тему “Систему вибору контрагентів для ремонтних служб енергетичної інфраструктури” за спеціальністю 122 - “Комп’ютерні науки”, освітня програма “Аналітика даних”.

У випускній кваліфікаційній роботі проведено аналіз сучасних методів вибору контрагента для ремонтних служб енергетичної інфраструктури, розглянуто методи оцінювання контрагентів, розроблено програмне забезпечення, що обирає найкращого контрагента, на основі обраних спеціалістами критеріїв та альтернатив.

Ключові слова: контрагент, комплектуючих енергетичного відділу; метод аналізу ієрархій; особа, що приймає рішення;

Summary

The degree project: «Movies recommendation system» has completed by Horelyk Andrii specialty 122 – «Computer Science», educational program «Data Analysis».

Andrii Dmytrovych Horelyk has completed his graduation qualification work on the topic "System for selecting contractors for energy infrastructure repair services" in the specialty 122 - "Computer Science," educational program "Data Analytics."

The graduation qualification work includes an analysis of modern methods for selecting a supplier for repair services in energy infrastructure, an examination of supplier evaluation methods, and the development of software that selects the best supplier based on criteria and alternatives chosen by specialists.

Keywords: energy department supplier, analytic hierarchy process, decision-maker

Зміст

Перелік умовних позначень та скорочень

B2B портал(business-to-business),

Приватне Акціонерне Товариство(ПрАТ),

метод аналізу ієрархій(МАІ);

особа, що приймає рішення(ОПР)

Вступ

Сучасний бізнес вимагає ефективного вибору контрагентів, які можуть забезпечити компанію якісними та своєчасно поставленими комплектуючими. В рамках цього контексту особливо важливим є забезпечення ефективного функціонування енергетичної інфраструктури, яка є ключовим компонентом сучасного господарства. Приватне Акціонерне Товариство "Полтавський Алмазний Інструмент" є одним з небагатьох полтавських заводів, що до сих пір функціонує, і є провідним виробником високоякісного алмазного інструменту. Завдяки своїй науково-виробничій базі та лабораторіям, компанія постійно розширює асортимент продукції і успішно постачає свою продукцію до більш ніж 30 країн світу.

Проте, для забезпечення безперешкодного виробництва та виконання ремонтних робіт, ПрАТ "Полтавський Алмазний Інструмент" має обирати найкращих контрагентів комплектуючих. Цей процес вимагає об'єктивного та науково обґрунтованого підходу, щоб запобігти зайвим втратам та ірраціональному використанню ресурсів. З метою покращення цього процесу та забезпечення оптимального вибору контрагентів, виникає необхідність у розробці програмного модулю, який буде допомагати керівництву вибирати найкращого контрагента засобів та послуг для ремонтних служб енергетичної інфраструктури.

Основною метою даної кваліфікаційної роботи є розроблення програмного модулю, який допоможе керівництву обрати кращого контрагента та запобігти зайвим втратам та ірраціональному використанню ресурсів. Дослідження спрямоване на створення системи вибору контрагентів для ремонтних служб енергетичної інфраструктури, яка дозволить забезпечити ефективність та надійність виробничих процесів ПрАТ "Полтавський Алмазний Інструмент". В рамках дослідження

використовуватиметься метод аналізу ієрархій, що дозволить об'єктивно оцінити та порівняти контрагентів за різними критеріями.

Об'єктом дослідження даної роботи є вибір об'єктивно найкращого контрагента для ремонтних служб енергетичної інфраструктури компанії "Полтавський Алмазний Інструмент". Предметом дослідження є оціночні судження спеціалістів про контрагентів та розробка програмного модулю на основі методу аналізу ієрархій. Результати цього дослідження дозволять ефективно обирати контрагентів, забезпечуючи якість та надійність ремонтних робіт у сфері енергетичної інфраструктури. Розроблений програмний модуль дозволить зробити об'єктивні рішення щодо вибору контрагентів та підвищить конкурентоспроможність компанії. Ключові слова дослідження: кабельно-провідникова продукція, освітлювальні прилади, електричні апарати.

Розділ 1. Аналіз проблеми вибору контрагента для ремонтного відділу енергетичної інфраструктури

1.1 Роль контрагента у роботі ремонтного відділу енергетичної інфраструктури

Історично відносини між покупцями та контрагентами часто були антагоністичними; однак, за останні кілька років спостерігається позитивна зміна в цих відносинах. Тенденції, такі як скорочення термінів життя продукту, збільшення швидкості технологічних змін та закупівля у закордонних джерелах, сприяли поліпшенню комунікації та співпраці між покупцями та контрагентами, що має вплив на практику управління, таку як закупівля від одного джерела.

Вибір контрагента, як правило, є тривалим процесом оцінки. Контрагентів оцінюють за кількома критеріями, такими як структура ціноутворення, своєчасна доставка (час та вартість), якість продукту та обслуговування (наприклад, персонал, приміщення, дослідження та розробка, можливості тощо).

Часто ці критерії оцінки включають компроміси. Наприклад, один контрагент може пропонувати недорогі деталі з якістю нижче середнього, тоді як інший контрагент може пропонувати деталі вищої якості, але з невизначеною доставкою, створюючи таким чином компроміси. Крім того, важливість кожного критерію змінюється від одного закупівлі до іншої і ускладнюється фактом, що деякі критерії є кількісними (ціна, якість тощо), тоді як інші є якісними (обслуговування, гнучкість тощо). Тому потрібна техніка, яка може врахувати ставлення приймача рішень до важливості кожного критерію і поєднати якісні та кількісні фактори.

Хоча більшість покупців все ще вважають вартість основним фактором, все частіше використовуються нові, більш інтерактивні та

взаємозалежні критерії вибору. Обширний огляд літератури з критеріїв та підходів до вибору контрагентів можна знайти в роботі Вебера та Еллрама (1993). Нижче наведено підсумок підходів до вибору контрагентів, які користуються популярністю у літературі.

А саме для ПрАТ “Полтавський Алмазний Інструмент” вибір контрагента є критичним етапом, оскільки від цього вирішального рішення залежить успішність ремонтних робіт та забезпечення необхідних матеріалів для енергетичного відділу.

Завод має конкретні потреби щодо ремонтних товарів, які необхідно задовольнити. Це включає інструменти, запчастини, матеріали та інші комплектуючі, які необхідні для ефективного виконання ремонтних робіт в енергетичному відділі. Оскільки якість ремонтних робіт і безперебійність експлуатації обладнання мають вирішальне значення, важливо вибрати надійних та якісних контрагентів, які можуть задовольнити ці вимоги. Розглянемо поетапно стандартний процес вибору контрагента:

1. Визначення потреб - ремонтний відділ визначає свої потреби щодо ремонтних товарів, враховуючи типи та кількість товарів, які потрібно закупити. Це може включати інструменти, запчастини, матеріали тощо.
2. Пошук контрагентів - відділ закупівлі може проводити пошук потенційних контрагентів на основі рекомендацій, реклами, інтернет-пошуку або спеціалізованих директоріїв контрагентів.
3. Оцінка контрагентів - Після знаходження потенційних контрагентів відділ закупівлі проводить оцінку їхніх можливостей, якості продукції, цін, термінів поставки та інших важливих факторів. Це може включати перегляд їхніх веб-сайтів, порівняльний аналіз, читання відгуків користувачів або навіть організацію зустрічей з представниками компаній.

4. Вибір контрагентів - На основі результатів оцінки відділ закупівлі здійснює вибір контрагента, що найкраще відповідає їхнім потребам. Під час цього процесу можуть використовуватися критерії, такі як якість товарів, ціна, надійність контрагента, репутація компанії тощо.
5. Укладання угоди - Після вибору контрагента керівництво та відділ закупівлі укладають угоду або контракт з обраною компанією. У цьому документі встановлюються взаємні зобов'язання сторін, включаючи умови поставки, ціну, терміни, гарантії, вимоги до якості товарів та інші умови співпраці. Угода може бути укладена на певний період або на основі поставок за потребою.

Після укладення угоди завод може здійснювати контроль якості та виконання умов поставки з боку контрагента. Це може включати перевірку товарів перед прийманням, аудити контрагентів, регулярний моніторинг якості та ефективності поставок.

Під час використання даної практики виникає багато проблем, які потрібно вирішувати, а саме:

- Обмежений вибір - Існує можливість обмеженого вибору контрагентів, особливо якщо є специфічні вимоги до товарів або обмежений ринок. Це може ускладнити процес пошуку і знизити конкурентну перевагу.
- Недостатня якість - Не всі контрагенти можуть гарантувати високу якість товарів. Це може призвести до проблем з якістю ремонтних робіт та використання непридатних або ненадійних матеріалів.
- Нестабільність поставок - Деякі контрагенти можуть мати проблеми з постачанням товарів у відповідний термін або в необхідних обсягах. Це може вплинути на продуктивність ремонтної служби та спричинити затримки в ремонтних роботах.

- Вартість та ефективність - Ціна товарів і послуг контрагента може бути важливим фактором при виборі. Завод повинен забезпечити баланс між якістю товарів і їх вартістю, забезпечуючи оптимальне співвідношення ціни та якості.
- Недостатня комунікація - Недостатня комунікація між відділами, які відповідають за вибір контрагента та використання його товарів, може призвести до непорозумінь та несумісності потреб. Недостатнє спілкування може спричинити вибір контрагента, який не повністю задовольняє вимоги ремонтної служби.
- Ризики поставок - Залежність від одного контрагента може створювати ризики для заводу. В разі проблем з поставками, збільшення цін або недоступності товарів, ремонтна служба може потрапити в складну ситуацію, що призведе до затримок у ремонтних роботах.

Враховуючи ці проблеми, важливо здійснювати ретельний аналіз і оцінку контрагентів, забезпечуючи якість, стабільність поставок та оптимальні умови співпраці. Постійний контроль якості та ефективності поставок також допомагає забезпечити успішне співробітництво з контрагентами.

1.2 Оцінка та вибір можливих контрагентів

Поверхнево розглянувши методи для вирішення задачі вибору контрагента в минулому підрозділі, прийшли до висновку, що метод аналізу ієрархії найкраще підходить для даних умов.

Для застосування даного методу маємо обрати вхідні дані: спеціалістів з різних відділ, в подальшому називатимемо - особа, що приймає рішення(ОПР); критерії, за якими будемо обирати контрагента; альтернативи - контрагенти; оціночні судження ОПР щодо критеріїв та альтернатив.

Критерії, що мають впливати на вибір контрагента:

1) Ціновий сегмент
2) Логістика
3) Якість товару
4) Різноманіття асортименту
5) Обслуговування

Перелік контрагентів:

- ТОВ "Е.МІР"
- Schneider Electric UA
- ПП "Сігма Кабель"
- ТОВ "Меганом"
- ТОВ ДП "Поло електрообладнання"

ТОВ "Е.МІР" - Ця компанія є контрагентом-посередником більше десяти років, здійснює продаж і комплексне постачання високоякісної продукції: кабельно-провідникової, електротехнічної, світлотехнічної.

Schneider Electric - це велика світова компанія, що спеціалізується на розробці та виробництві продуктів та рішень у сфері енергетики та автоматизації. В Україні вони працюють в багатьох галузях, включаючи електротехніку, системи управління енергією та інші.

ПП "Сігма Кабель" - Ця компанія спеціалізується на виробництві та постачанні кабельно-провідникової продукції. Вони можуть надавати різноманітні типи кабелів та проводів для різних галузей, таких як електротехніка, телекомунікації тощо.

ТОВ "Меганом" - один із лідерів енергоринку. Вже понад 10 років Меганом Україна є провідним контрагентом кабельно-провідникової продукції, що забезпечує надійну безперебійну роботу будь-якого об'єкта.

ТОВ ДП "Поло електрообладнання" - Оптова торгівля побутовими електротоварами й електронною апаратурою побутового призначення для приймання, записування, відтворювання звуку й зображення.

1.3 Аналіз методів вибору контрагентів для ремонтного відділу енергетичної інфраструктури

З метою забезпечення об'єктивності та наукового підходу до вибору контрагента, пропонується використовувати універсальний метод, в якому кожен відділ буде приймати участь, а експерти з кожного відділу будуть суб'єктивно оцінювати важливість кожного критерію. Додатково, варто враховувати проблему упередженості, яка може виникати при виборі контрагента. Це пов'язано з можливим перекосом у сприйнятті працівниками відділу постачання, які, через тривалі взаємини з певними фірмами, можуть надавати більше значення самому продавцю, ніж якості його товару, що не завжди вигідно для заводу.

Існує декілька методів що допоможуть нам в вирішенні задачі вибору контрагента. Ось короткий опис кількох з них:

- Метод вагової суми (Weighted Sum Method)
- Метод аналізу ієрархії (MAI)(Analytic Hierarchy Process)
- Метод простору балів (Scorecard Approach)
- Метод обмежень (Constrained Optimization)

Ці методи можуть бути використані як окремо, так і в комбінації, в залежності від конкретних потреб та характеристик вибору контрагента.

Метод вагової суми - цей метод полягає в призначенні вагових коефіцієнтів для кожного критерію і ранжуванні контрагентів на основі загальної суми вагованих оцінок. Кожному критерію присвоюється ваговий коефіцієнт, що відображає його важливість, і застосовується формула для розрахунку оцінки кожного контрагента.

Даний метод також має свої недоліки. Ось деякі з них:

- Суб'єктивність визначення вагових коефіцієнтів - вагові коефіцієнти визначаються на основі оцінки важливості кожного критерію. Однак, визначення цих вагових коефіцієнтів може бути

суб'єктивним і залежати від особистих поглядів або припущень. Різні особи можуть мати різні погляди на важливість критеріїв, що може призвести до неточностей у вагових оцінках і, відповідно, в результатах вибору.

- Відсутність урахування взаємозв'язків між критеріями - метод вагової суми розглядає кожний критерій окремо, не враховуючи можливі взаємозв'язки між ними. Однак, у деяких випадках критерії можуть бути взаємозалежними або впливати один на одного. Недостатня увага до цих взаємозв'язків може призвести до недостовірних або недоцільних результатів вибору контрагента.
- Нездатність враховувати нелінійність та нелінійні зв'язки - метод вагової суми передбачає лінійну комбінацію вагованих оцінок критеріїв для оцінки контрагентів. Проте, в деяких випадках важливість критеріїв може мати нелінійну природу або взаємозв'язки між ними можуть бути нелінійними. Метод вагової суми не здатний адекватно врахувати ці нелінійність та нелінійні зв'язки, що може призвести до спотворених результатів.
- Чутливість до змін вагових коефіцієнтів - результати методу вагової суми можуть значно змінюватися залежно від вагових коефіцієнтів, які використовуються. Навіть невеликі зміни в вагових коефіцієнтах можуть призвести до зміни ранжування контрагентів. Це може створювати нестабільність і недостовірність результатів, особливо якщо вагові коефіцієнти визначаються на основі суб'єктивних оцінок.

Враховуючи, що даний метод не враховує зв’язки між критеріями, ми не можемо сказати, що даний метод найкраще відповідає умовам поставленої нами задачі. Також “Чутливість до змін вагових коефіцієнтів” - може сильно вплинути на результат, і створити несприятливі умови.

Метод аналізу ієрархій (MAI) - метод використовує ієрархічну структуру для розкладу проблеми на кілька рівнів критеріїв та підкритеріїв. Цей метод використовує попарне порівняння критеріїв та визначення їх відносних вагових коефіцієнтів. За допомогою матриць порівнянь на основі експертної оцінки, MAI дозволяє визначити пріоритетність критеріїв і здійснити остаточний вибір контрагента.

Хоча MAI є популярним і широко використовується для прийняття рішень, він також має свої мінуси. Ось кілька недоліків методу:

- Чутливість до складності проблеми - MAI може бути чутливим до змін у структурі проблеми та відносної вагомості критеріїв. Навіть незначні зміни в порівнянні або вагових коефіцієнтах можуть призвести до значних змін у відповідях та ранжуванні альтернатив.
- Суб'єктивність оцінок - MAI базується на експертних оцінках, які можуть бути суб'єктивними та піддаються індивідуальному сприйняттю. Різні експерти можуть мати різний рівень знань, досвіду та особистих уподобань, що може призвести до відмінних результатів при застосуванні MAI.
- Складність ітерацій - При великій кількості критеріїв або альтернатив, процес ітерацій у MAI може бути часо- та ресурсомістким. Розрахунок парних порівнянь для кожної пари критеріїв та альтернатив може вимагати значної кількості часу та зусиль.
- Вимога до структурованості - MAI потребує чіткої ієрархічної структури критеріїв та альтернатив. Врахування додаткових аспектів або зміни структури може бути викликом і вимагати перегляду та переробки моделі.

Враховуючи дані недоліки, можна затвердити, що “Суб'єктивність оцінок” ніяк не може бути вирішена в нашій задачі, бо відділи закупівлі та

ремонту заводу, мають приймати участь у виборі контрагента. І в інтересах заводу сформувати результат спираючись на суб’єктивні оцінки локальних експертів.

“Складність ітерацій” - всі критерії та альтернативи будуть вводитися користувачами самостійно, тож кількість вхідних даних не може значним чином вплинути на складність розрахунків.

“Вимога до структурованості” - в даному випадку, критерії та альтернативи, будуть оголошені завчасно, тож структура моделі буде всім відома до початку введення даних.

Метод простору балів - цей метод використовує побалову систему для оцінки контрагентів на основі певних критеріїв. Кожному критерію присвоюється ваговий бал, а контрагенти оцінюються за кожним критерієм. Загальний бал кожного контрагента розраховується шляхом зведення балів по критеріях. Контрагент з найвищим загальним балом вважається найкращим вибором.

Метод простору балів, як і будь-який інший метод, має свої переваги та недоліки. Ось кілька мінусів методу простору балів:

- Суб'єктивність - оцінка контрагентів за допомогою методу простору балів вимагає від експертів або приймача рішень визначення вагових балів для кожного критерію. Оцінка може бути суб'єктивною і залежати від особистих поглядів та переконань експерта, що може спотворити оцінку та вплинути на результати вибору.
- Важкість порівняння - задача порівняння контрагентів за кожним критерієм може бути складною і вимагати значних зусиль та експертного знання. Особливо у випадках, коли критерії мають різну природу або складність вимірювання, визначення відносної вагомості може бути викликом.

- Вразливість до перекосів - метод простору балів може бути вразливим до перекосів або впливу окремих критеріїв на загальну оцінку. Якщо один або кілька критеріїв мають надмірну вагу або неправильно оцінюються, це може призвести до необ'єктивної оцінки контрагентів та вплинути на кінцевий вибір.
- Відсутність гнучкості - метод простору балів може бути менш гнучким у порівнянні з деякими іншими методами. Його результати можуть бути менш чутливими до змін в оцінках або вагах критеріїв, оскільки прямолінійно залежать від призначених балів. В разі зміни пріоритетів або зміни у вимогах до контрагентів, цей метод може вимагати повторної оцінки та перерахунку балів.

Недоліки даного методу, саме для даної задачі, приведемо у вигляді порівняння з методом аналізу ієрархії:

- Структура методу - у методі простору балів оцінюються контрагенти на основі вагових балів для кожного критерію. Критерії оцінюються окремо, а потім обчислюється загальний бал для кожного контрагента. Загальний бал використовується для порівняння контрагентів. З іншого боку, метод аналізу ієрархії використовує ієрархічну структуру, де критерії і підкритерії розглядаються на різних рівнях. Відносні ваги критеріїв визначаються за допомогою матриці парних порівнянь, і використовується розрахунок локальних та глобальних пріоритетів для вибору контрагентів.
- Оцінювання - у методі простору балів оцінка контрагентів здійснюється шляхом призначення балів на основі кожного критерію. Експерт або приймач рішень визначає бали вручну або за допомогою певних оцінювальних правил. У методі аналізу ієрархії, оцінка виконується за допомогою матриці парних

порівнянь, де кожен критерій порівнюється з іншими за шкалою швидкості важливості.

- Обробка результатів - в методі простору балів загальний бал кожного контрагента розраховується шляхом вагового сумування оцінок по кожному критерію. Контрагент з найвищим загальним балом вважається найкращим. У методі аналізу ієрархії розраховуються локальні пріоритети для кожного рівня ієрархії, а потім обчислюються глобальні пріоритети для порівняння контрагентів.
- Універсальність - метод простору балів є простим у застосуванні та розумінні, і може бути широко використаний у простих ситуаціях вибору контрагента. З іншого боку, метод аналізу ієрархії надає більш структурований підхід до прийняття рішень, зокрема при врахуванні багатьох критеріїв і складних взаємозв'язків.

Метод простору балів має свої переваги, але спираючись на обмеження і порівняння з MAI, ми не станемо обирати цей метод для даної задачі.

Метод обмежень - цей метод використовує математичне моделювання для знаходження найкращого контрагента з урахуванням обмежень, таких як бюджетні обмеження або обмеження на час поставки. Задача вибору контрагента формулюється як задача оптимізації, де метою є максимізація чи мінімізація певної функції враховуючи обмеження.

Метод обмежень для вибору контрагента також має свої недоліки. Ось деякі з них:

- Слабкість врахування некерованого ризику - метод обмежень може не враховувати некерований ризик, такий як несподівані зміни на ринку або надзвичайні події. Він зазвичай базується на попередніх даних та передбаченнях, тому може бути обмежений в

урахуванні непередбачуваних обставин, які можуть вплинути на контрагента.

- Обмеженість врахування якості та інновацій - метод обмежень зазвичай зосереджується на параметрах, таких як ціна, доставка та обсяги постачання. Він може недостатньо враховувати якість продукту, рівень обслуговування та інноваційні можливості контрагента, що можуть бути важливими аспектами вибору.
- Залежність від доступних даних - метод обмежень потребує наявності достатньої кількості точних та надійних даних про контрагентів. В разі обмеженого доступу до інформації або неоднорідності даних може бути складно застосувати метод ефективно.
- Недостатня гнучкість - метод обмежень може бути менш гнучким у порівнянні з іншими методами, оскільки він базується на фіксованих обмеженнях та параметрах. Він може не враховувати можливість зміни умов або пріоритетів під час процесу вибору контрагента.
- Важкість урахування багатокритеріальності - метод обмежень може бути менш ефективним у вирішенні багатокритеріальної задачі вибору контрагента, де кілька критеріїв мають різну важливість. У таких випадках використання інших методів, які дозволяють більш гнучкий та комплексний аналіз критеріїв, може бути більш доцільним.

В даному випадку приділяємо більше уваги до пункту важкість урахування багатокритеріальності. Так як під час вибору контрагента ми плануємо обрати якомога більше критеріїв, щоб результат вийшов наближеним до реальності.

З іншого боку, MAI базується на ієрархічній структурі та попередньо визначених вагових коефіцієнтах. Він залучає експертні оцінки та порівняння,

щоб визначити відносну важливість різних критеріїв. AI дозволяє ієрархічно ранжувати критерії та варіанти вибору, шляхом порівняння їх за парами і визначення вагових коефіцієнтів. Цей метод зазвичай використовує більш суб'єктивні оцінки та експертні знання для прийняття рішень.

1.4 Постановка задачі

Результати аналізу вибору контрагентів для ремонтних служб енергетичної інфраструктури показують актуальність створення автоматизованої системи для цього процесу. Основною метою дипломного проекту є розробка системи, яка допоможе автоматизувати етап оцінювання контрагентів. Під час аналізу існуючих методів було виявлено, що математичною основою цієї системи буде метод аналізу ієрархії. Такий підхід спростить процес оцінки контрагентів та дозволить зробити обґрунтоване рішення щодо їх вибору.

Для забезпечення більш глибокого розуміння майбутньої системи необхідно визначити та деталізувати як функціональні, так і нефункціональні вимоги. Цей процес допоможе уточнити завдання та чітко визначити обов'язкові функції, які повинні бути реалізовані. Задоволення функціональних вимог є важливим для належної роботи системи, оскільки вони спрямовані на користувача і визначають основний функціонал, що повинен бути доступний в системі.

Функціональні вимоги:

- Система повинна мати змогу приймати вхідні дані json-файлом, де всі дані будуть типізовані.
- Система повинна мати змогу приймати вхідні дані через інтерфейс користувача, та формувати з них json-файл
- Система повинна бути здатною опрацьовувати дані багатьох експертів

- Система повинна мати можливість приймати як вхідні дані назви критеріїв та альтернатив, та значення матриць попарних порівнянь
- Програма повинна надавати користувачу можливість записати результати розрахунків у файл різних форматів
- Програма повинна виводити назви критеріїв та альтернатив на головний екран
- Програма повинна надавати можливість користувачу завантажити результати.

Нефункціональні вимоги:

- Система повинна бути надійною, швидкою та ефективною.
- Має бути гарантована точність та об’єктивність результатів
- Програма повинна обробляти помилки пов’язані з некоректною формою введених даних.
- Інтерфейс програми повинен бути адаптований під різний розмір екрану чи вікна в якому запущена програма.

Основним результатом функціонування програмного модулю є вибір найкращого контрагента на основі методу аналізу ієрархій. Вихідна інформація включатиме рейтинги контрагентів, розраховані на основі їх оцінок за критеріями вибору та їх вагомостями.

1.5 Висновки до першого розділу

У цьому розділі було проведено аналітичний огляд сучасного стану досліджень у галузі вибору контрагентів. Зазначений огляд виявив проблемні моменти та необхідність вдосконалення методологій та практик вибору контрагентів. Це підтверджує актуальність розробки програмного модулю для об'єктивного вибору найкращого контрагента та оптимального використання ресурсів.

Мета даної випускної кваліфікаційної роботи полягає у розробці програмного модулю, який базується на методі аналізу ієрархій, для вибору найкращого контрагента засобів та послуг. Вимоги до об'єкта дослідження були чітко сформульовані, включаючи системні, функціональні та нефункціональні вимоги, а також вимоги до вхідної та вихідної інформації. Ці вимоги є основою для подальшого розвитку та реалізації програмного модулю.

Враховуючи аналіз результатів досліджень та практик, встановлено актуальність проведення даної випускної кваліфікаційної роботи, спрямованої на розробку програмного модулю, який допоможе керівництву обирати найкращого контрагента та оптимізувати використання ресурсів. Варто відзначити, що розроблений програмний модуль буде ґрунтуватися на методі аналізу ієрархій (MAI), який вже успішно використовується у прийнятті рішень.

Завершуючи перший розділ, можна зробити висновок, що розробка програмного модулю для вибору контрагент є актуальним та важливим завданням. Аналітичний огляд досліджень підтвердив необхідність вдосконалення існуючих підходів та розробки оригінальних рішень. Постановка задачі визначила цілі та вимоги до розроблюваного програмного модулю, що є основою для подальшого розвитку.

Розділ 2. Математична модель вибору контрагентів для ремонтних служб енергетичної інфраструктури

2.1 Особливості методу аналізу ієрархії

Метод аналізу ієрархії(МАІ) - математичний інструмент системного підходу до вирішення складних проблем прийняття рішень. Основне застосування методу полягає в підтримці прийняття рішень за допомогою ієрархічної композиції завдання та рейтингування альтернативних рішень.

Цей метод розроблений американським математиком Томасом Сааті, який написав про нього книги, розроблював програмні продукти і протягом 20 років проводим симпозіуми ISAHP (Interantional Symposium on Analytic Hierarchy Process).

МАІ широко використовується на практиці і активно розвивається вченими всього світу. У його основі поряд з математикою закладені і психологічні аспекти. МАІ дозволяє зрозумілим і раціональним чином структурувати складну проблему прийняття рішень у вигляді ієрархії, порівняти і виконати кількісну оцінку альтернативних варіантів рішення. Метод аналізу ієрархій використовується у всьому світі для прийняття рішень в різноманітних ситуаціях: від управління на міждержавному рівні до розв’язання галузевих і приватних проблем в бізнесі, промисловості, охороні здоров’я і освіті.

Метод надає змогу:

- виокремити структурні елементи задачі прийняття рішень і формалізувати зв’язки між ними
- визначити системи переваги особи(ОПР), що приймає рішення, і критеріїв, за якими оцінюють альтернативи
- синтезувати правило прийняття рішень, яке ґрунтується на перевагах одних альтернатив у порівнянні з іншими

Ієрархія - система, в якій рівні розташовані та пронумеровані так, що:

1. нижній рівень містить рейтингові альтернативи
2. вузли з вищих рівнів можуть домінувати тільки над вузлами нижчих рівнів

В ієрархії зв'язки визначають шляхи однієї спрямованості - від вершини до альтернатив через проміжні рівні, які складаються з вузлів-критеріїв (Рисунок 2.1)

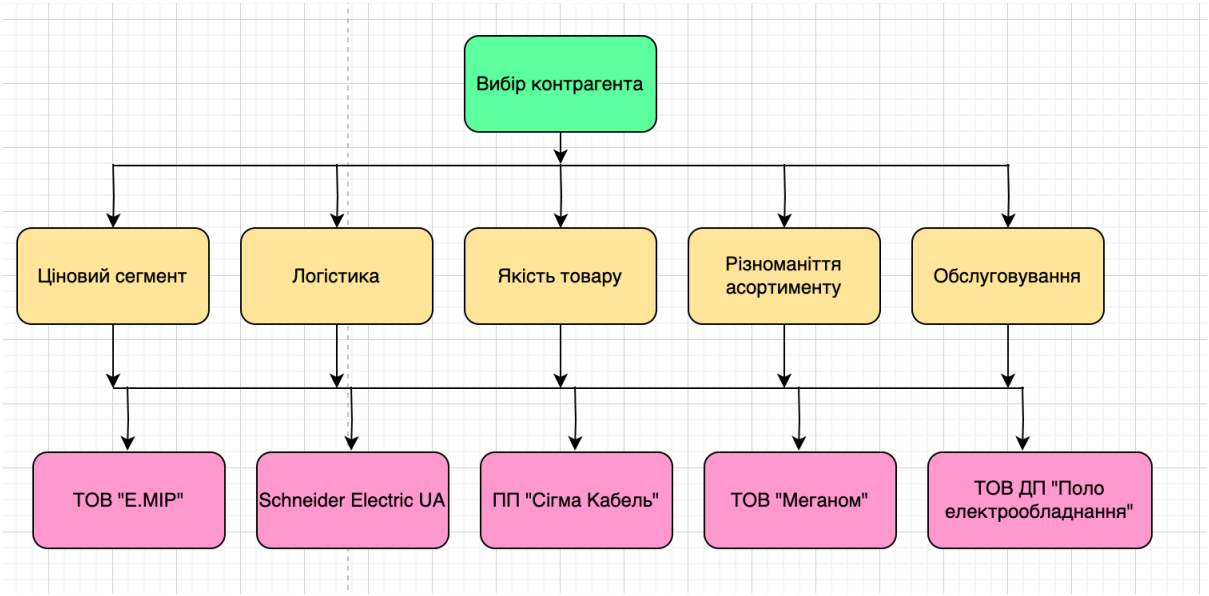

Рисунок 2.1 - приклад застосування MAI, вибір контрагента за критеріями

Ситуації прийняття рішень, до яких може застосовуватися MAI, включають:

- Вибір — вибір однієї альтернативи із заданого набору альтернатив, зазвичай там, де задіяно кілька критеріїв прийняття рішення.
- Ранжування — Введення набору альтернатив у порядку від найбільш до найменш бажаного.
- Пріоритетність — Визначення відносних достоїнств членів набору альтернатив, на відміну від вибору одного або просто ранжування їх

- Розподіл ресурсів — Розподіл ресурсів між набором альтернатив
- Бенчмаркинг — Порівняння процесів у власній організації з процесами інших організацій з найкращими практиками
- Управління якістю — Розгляд багатовимірних аспектів якості та вдосконалення якості
- Вирішення конфліктів — Вирішення суперечок між сторонами з, імовірно, несумісними цілями або позиціями

MAI має кілька переваг, які варто враховувати:

- Структурований підхід - надає структурований підхід до прийняття рішень. Він дозволяє розкласти складну проблему на менші частини, такі як критерії та альтернативи, що спрощує аналіз та розуміння проблеми.
- Вагомість критеріїв - дозволяє враховувати вагомість різних критеріїв при виборі контрагента. Шляхом парного порівняння критеріїв експертами, можна встановити їхню відносну важливість, що допомагає зробити більш обґрунтоване ранжування.
- Логічна модель - надає логічну модель для прийняття рішень. Він використовує математичні принципи та логічні відношення для порівняння критеріїв та альтернатив. Це дозволяє структурувати процес прийняття рішень та зробити його більш об'єктивним.
- Використання експертного знання - дає можливість використовувати експертне знання та досвід для оцінки критеріїв та альтернатив. Це дозволяє враховувати експертні думки та переваги при виборі контрагента.
- Зручний для порівняння - надає фреймворк для порівняння альтернатив. Шляхом створення матриць парних порівнянь, можна систематично порівняти кожну альтернативу з кожним

критерієм. Це допомагає зробити більш об'єктивне та прозоре ранжування.

Розглянемо основні етапи побудови методу аналізу ієрархій.

2.2 Структуризація задачі у вигляді ієрархічної структури з декількома рівнями: цілі, критерії, альтернативи

Аналіз проблеми починається з побудови ієрархічної структури (Рисунок 1.1), яка включає мету(ціль), критерії та альтернативи. Кожен елемент ієрархії може представляти різні аспекти задачі, що розв’язується, причому до уваги можуть бути прийняті як матеріальні так і нематеріальні чинники, вимірювані кількісні параметри та якісні характеристики, об’єктивні дані і суб’єктивні експертні оцінки.

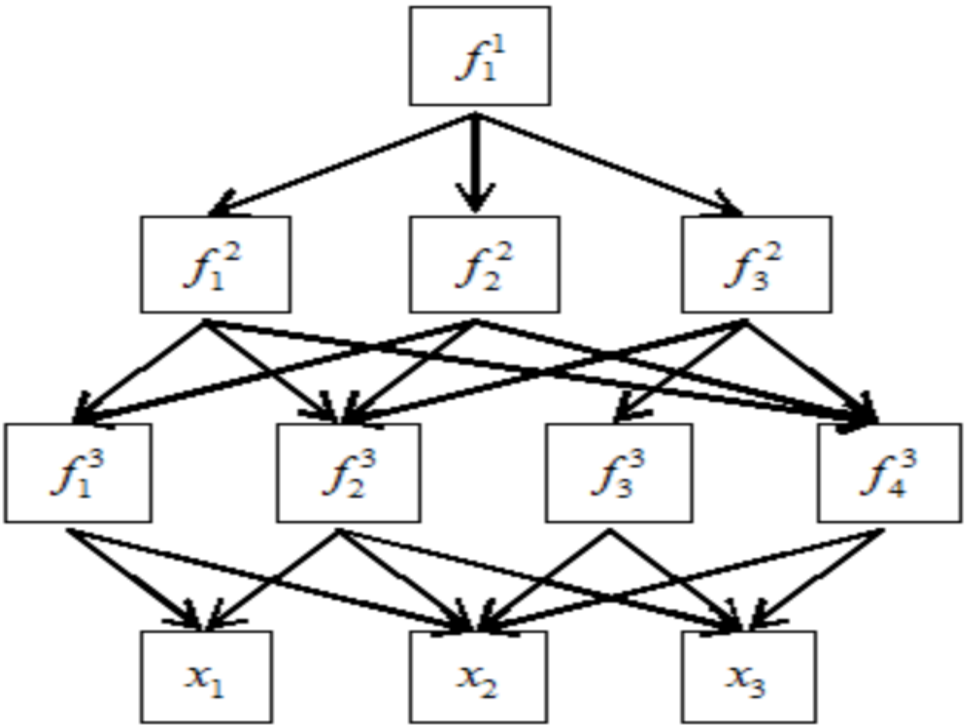

Рисунок 2.2 - приклад моделі ієрархічної задачі

На схемі (Рисунок 2.2) елементи $f_{\square}^{\square}$ - критерії ієрархії, де “i” вказує рівень ієрархії, а “j“ порядковий номер. $\square_{\square}$ - альтернативи

2.3 Визначення пріоритетів елементів ієрархічної структури

Наступним етапом аналізу є визначення пріоритетів, що представляють відносну важливість або перевагу елементів побудованої ієрархічної структури , за допомоги процедури парних порівнянь.

Для встановлення відносної важливості елементів ієрархії використовується шкала відношень Сааті.

Таблиця 2.1 - Шкала відношень Сааті(ступеня значимості дій)

Ступінь значущості	Визначення	Пояснення
1	Однакова значимість	Дві події мають однаковий внесок у досягнення мети
3	Слабка значимість	Існують недостатньо переконливі міркування на користь переваги однієї з дій
5	Істотна значимість	Маються надійні дані дл того, щоб показати перевагу однієї з дій
7	Очевидна значимість	Переконливе свідчення на користь однієї дії перед іншою
9	Абсолютна значимість	Незаперечні переконлив свідчення на користь переваги однієї дії перед іншою

2,4,6,8	Проміжні значення між сусідніми судженнями	Ситуація, коли необхідн компромісне судження

Дана шкала(таблиця 1) надає експерту можливість ставити у відповідність ступеням переваги одного фактору перед іншим - деяке число.

Матриця попарних порівнянь знаходиться за аксіомою пов’язаності.

Якщо m(a,b) - пріоритет, що визначає у скільки разів деякий елементі єрархії “a” має переваги порівняно з іншим елементом “b”, то виконується умова

$$□(□,□) = 1/(□,□) \quad (2.1)$$

Наприклад, якщо “a” в 2 рази має перевагу над елементом “b”, то з цього випливає, що “b” в 0,5 разів має перевагу над “a”. Модель побудови можна розглянути в таблиці 2.2. Де k - може бути як критерієм так і альтернативою, n - загальна кількість розглянутих критеріїв(або альтернатив) в даній матриці попарних порівнянь, $□_{□,□}$ - відношення за шкалою Сааті.

Таблиця 2.2 - правило побудови матриці попарних порівнянь

	$□_1$	$□_2$	…	$□_□$
$□_1$	1	$□_{1,2}$	…	$□_{1□}$
$□_2$	$1/□_{1,2}$	1	…	$□_{2□}$
…	…	…	…	…
$□_□$	$1/□_{1□}$	$1/□_{2□}$	…	1

2.4 Знаходження вектору пріоритетів

Розрахунок вектору пріоритетів включає детальний процес нормалізації матриці зіставлення для отримання відносної важливості елементів у порівнянні один з одним. Давайте розглянемо цей процес крок за кроком з використанням формул.

Припустимо, у нас є матриця зіставлення розміром n x n, де n - кількість елементів у порівнянні. Матриця зіставлення позначається як A і має наступний вигляд:

A = [□$_{□□}$] (розмірність n x n), де □$_{□□}$ - значення, яке відображає відносну важливість елемента i в порівнянні з елементом j.

1) Нормалізація матриці зіставлення:

Нормалізація полягає у діленні кожного елемента □$_{□□}$ на суму елементів відповідного стовпця. Це виконується за формулою:

□$_{□□}$ = □$_{□□}$ / $\sum_{□}^{□}$ □$_{□}$ (для k = 1 до n)

(2.2)

В формулі (2.2) □$_{□□}$ - нормалізоване значення елемента □$_{□□}$. Значення □$_{□□}$ відображають відносну важливість елемента i в порівнянні з елементом j, ураховуючи суму важливостей всіх елементів у стовпці j.

2) Обчислення суми кожного рядка:

Після нормалізації матриці, для кожного рядка матриці b обчислюється сума його елементів:

□$_{□}$ = $\sum_{1}^{□}$ □$_{□}$ (для j = 1 до n) (2.3)

В формулі (2.3), □$_{□}$ - сума елементів у рядку i. Значення s_i відображають вагу або пріоритет елемента i в порівнянні з усіма іншими елементами.

3) Обчислення вектору пріоритетів:

Для отримання вектору пріоритетів, кожне значення s_i ділиться на суму всіх s_i. Це виконується за формулою:

$$□_□ = \frac{\sum_{I}^{□} □_□}{\sum_{I}^{□} □_□} \text{(для k = 1 до n)} \quad (2.4)$$

В формулі (2.4), $□_□$ - значення вектору пріоритетів для елемента i.

Вектор пріоритетів w відображає відносну важливість кожного елемента у порівнянні з усіма іншими елементами і має розмірність n x 1.

Цей вектор надає важливу інформацію про пріоритетність елементів у відповідному порівнянні, що допомагає зробити обґрунтовані висновки та прийняти рішення.

2.5 Оцінювання узгодженості суджень експертів

При порушенні однорідності суджень, ранг матриці буде відмінний від одиниці і вона буде мати кілька власних значень. Однак, при невеликих відхиленнях суджень від однорідності, одне з власних чисел буде істотно більшим за інші і приблизно дорівнюватиме порядковій матриці.

Однорідність суджень експертів оцінюється індексом однорідності(ІО), що дорівнює:

$$□□ = (□_{□□□} − □)/(□ − I) \quad (2.5)$$

В формулі (2.5) $\lambda_{□□□}$ - максимальне власне значення, n - порядок матриці.

Відношення однорідності (ВО) визначається як відношення індексу однорідності до середнього значення (математичного сподівання) індексу однорідності випадково складеної матриці попарних порівнянь по шкалі від 1 до 9 обернено-симетричної матриці з відповідними оберненими величинами елементів.

Відношення однорідності дорівнює:

$$BO = □□/□(□□) \quad (2.5)$$

В формулі (2.5), □(□□) - середнє значення(математичне сподівання) індексу однорідності випадковим чином складеної матриці попарних порівнянь по шкалі від 1 до 9 обернено-симетричної матриці з відповідними оберненими величинами елементів, що базується на експериментальних даних (таблиця 2)

Таблиця 2.2 - Математичне сподівання для певного порядку матриці

Порядок матриці	1	2	3	4	5	6	7	8	9	10	11	12
□(□□)	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49	1,51	1,48

Відношення однорідності в контексті аналізу ієрархії вказує на ступінь узгодженості або консистентності оцінок, наданих експертами. Це поняття використовується для перевірки стабільності та надійності оцінок експертів і визначення їх взаємної узгодженості.

Розрахунок відношення однорідності включає наступні кроки: експерти надають оцінки відносної важливості пар елементів у ієрархії, оцінки використовуються для побудови матриці зіставлення, ваги елементів обчислюються на основі оцінок і нормалізації ваг, розраховується індекс узгодженості Шварца (CI), індекс узгодженості порівнюється з допустимим діапазоном значень для перевірки узгодженості оцінок.

Оцінка узгодженості оцінок експертів є важливим кроком у методі аналізу ієрархії для забезпечення достовірності результатів та прийняття обґрунтованих рішень. Вона допомагає зменшити суб'єктивність і підвищує рівень довіри до отриманих ваг та пріоритетів у ієрархічній структурі.

2.6 Агрегація глобальних ваг

У ієрархії з одним рівнем критеріїв агрегація глобальних ваг є простим обчисленням середнього значення локальних ваг. Давайте розглянемо цей процес більш детально.

Припустимо, ми маємо n критеріїв на одному рівні. Кожному критерію буде відповідати локальна вага, яку ми позначимо як $□_□$, де i = 1, 2, ..., n.

Процес агрегації глобальних ваг складається з наступних кроків:

1. Обчислення суми локальних ваг:

Обчислюємо суму всіх локальних ваг, яка позначається як $□_□$, і вона обчислюється за формулою (2.6).

$$□_□ = □_1 + □_2 + ... + □_□ \quad (2.6)$$

2. Обчислення глобальних ваг:

Для кожного критерію обчислимо його глобальну вагу, яку позначимо як W_i. Глобальна вага обчислюється як відношення локальної ваги до суми локальних ваг за формулою (2.7).

$$□_□ = □_□ / □_□, \text{ для } i = 1, 2, ..., n \quad (2.7)$$

Таким чином, ми отримали глобальні ваги кожного критерію на основі їх локальних ваг. Глобальні ваги відображають важливість кожного критерію в контексті аналізу на глобальному рівні. Вони можуть бути використані для подальшого прийняття рішень або ранжування альтернатив, залежно від конкретного використання ієрархії.

2.7 Опис даних

Спеціально для розробки та тестування програмного застосунку з використанням MAI, було проведено декілька опитувань з працівниками енергетичної сфери, деякі з них безпосередньо працювали на ПрАТ “Полтавський Алмазний Інструмент”.

Визначення осіб, які прийматимуть рішення:

- Представник з відділу закупівель.

- Керівник однієї з ремонтних бригад.
- Власник однієї компанії-контрагента.
- Інженер заводу.

Етапи збору інформації серед учасників обговорення:

1. Узгодження критеріїв та альтернатив:
    i. Учасники узгоджують список критеріїв, які будуть використовуватись для оцінки контрагентів. Список критеріїв можна переглянути в розділі 1.2
    ii. Учасники також визначають список альтернатив (контрагентів), які будуть порівнюватись. Список альтернатив можна переглянути в розділі 1.2
2. Підготовка матриць попарних порівнянь:
    i. Кожен учасник заповнює матрицю попарних порівнянь, оцінюючи кожну альтернативу відносно кожного критерію. Наприклад, представник з відділу закупівель порівнює контрагентів з точки зору важливості критеріїв для відділу закупівель.
    ii. Кожен учасник заповнює свою власну матрицю попарних порівнянь, використовуючи свою експертну оцінку.
3. Збір інформації:
    i. Після того як всі учасники заповнили свої матриці попарних порівнянь, дані збираються та підготовлюються для подальшої обробки.

Збір даних на цьому завершено. Маємо 4 ОПР, 5 критеріїв та 5 альтернатив.

2.8 Висновок до другого розділу

У цьому розділі було повністю описано алгоритм роботи MAI, починаючи з введення даних, розрахунків матриць попарних порівнянь, розрахунків ваг та узгодження оціночних суджень експертів.

Для реалізації описаного алгоритму була побудована та детально розписана архітектура програмного забезпечення. Архітектура включала компоненти, такі як користувацький інтерфейс, модуль обробки даних, модуль обчислення та модуль виводу результатів. Кожен компонент мав свою відповідну функціональність, яка була описана у розділі.

Також було розглянуто процес підготовки даних для методу аналізу ієрархії. Цей процес включав зібрання думок експертів та аналіз їхніх оціночних суджень. Кожен експерт заповнив свою матрицю попарних порівнянь, відображаючи їхні вподобання та ваги критеріїв. Ці дані були зібрані та підготовлені для подальшого використання в методі аналізу ієрархії.

На завершення, розглянули бібліотеки, що будуть застосовані при розробці застосунку, його графічної складової та ядра розрахунків.

В цілому, описаний алгоритм роботи MAI, архітектура програмного забезпечення та процес підготовки даних надають комплексний погляд на метод аналізу ієрархії та його реалізацію. Ця інформація може бути використана для розробки програмного застосунку, який впроваджує метод аналізу ієрархії для прийняття рішень.

Розділ 3. Проектування та розробка системи вибору контрагента для ремонтного відділу енергетичної інфраструктури

3.1 Проектування системи вибору контрагента

Зробимо загальний огляд можливих компонентів та підрозділів архітектури програмного застосунку для методу аналізу ієрархії:

1. Графічний інтерфейс користувача:
    a. Вікна та елементи управління - Реалізація основного вікна програми з кнопками, текстовими полями та іншими елементами управління.
    b. Форми та діалогові вікна - Створення спеціальних форм або діалогових вікон для введення конкретних даних або відображення повідомлень.
    c. Валідація введення - Перевірка правильності введених даних користувачем та надання відповідних повідомлень про помилки.
2. Модуль обробки даних:
    a. Парсинг вхідних даних - Аналіз та розбір вхідних даних, які надає користувач.
    b. Валідація даних - Перевірка правильності та припустимості введених даних, забезпечення коректності формату та типу даних.
    c. Перетворення даних - Конвертація вхідних даних у придатну для обчислень форму, наприклад, у матрицю парних порівнянь.
3. Модуль обчислення:
    a. Алгоритми аналізу ієрархії - Реалізація методу аналізу ієрархії, таких як метод парних порівнянь, метод вагових коефіцієнтів, оцінювання узгодженості суджень експертів
    b. Обчислення ваг - Розрахунок ваг критеріїв та альтернатив на основі вхідних даних.

c. Визначення пріоритетів - Встановлення пріоритетів для критеріїв та альтернатив згідно з результатами аналізу.

4. Модуль виводу результатів:
   a. Текстові відповіді - Формулювання текстових відповідей, що пояснюють прийняті рішення та результати аналізу.
   b. Візуалізація даних - Створення графіків, діаграм або інших візуальних елементів для ілюстрації ієрархії, ваг та результатів.
   c. Збереження результатів - Можливість зберігати результати аналізу для подальшого використання або перегляду.

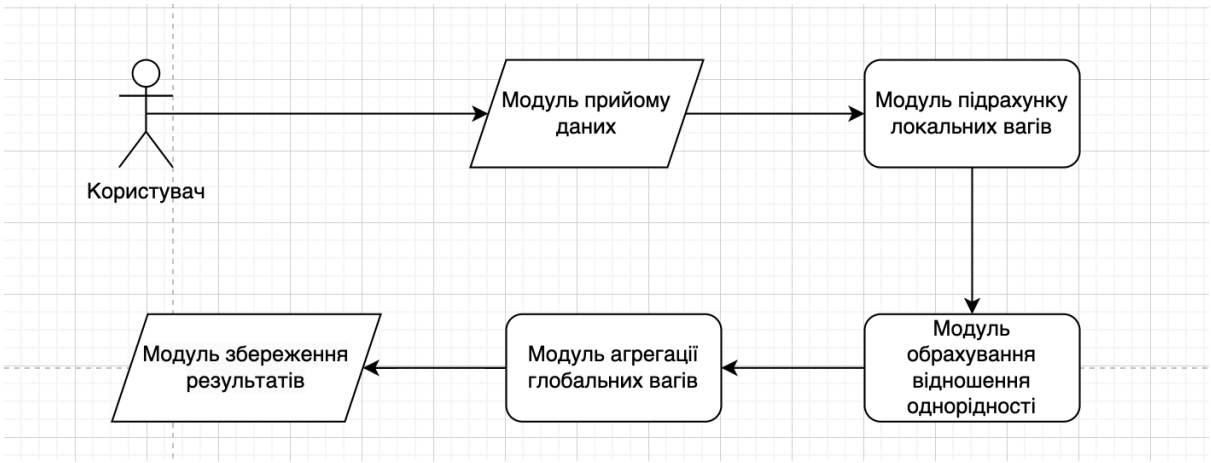

Рисунок 2.2.1 - загальна схема взаємодії модулів системи

На Рисунку 2.2.1 зображено загальну схему взаємодії модулів системи для вибору за методом аналізу ієрархій, що складатиметься з таких модулів, як:

- Модуль прийому даних - дані можуть будуть прийматися як з ручним введенням через користувацький інтерфейс, так і за певним шаблоном json-файлу
- Модуль підрахунку локальних вагів - приймаючи на вхід матрицю попарних порівнянь, повертатиме значення локальних вагів даної матриці, змодельованої певним експертом.

- Модуль обрахування відношення однорідності - підраховує відношення однорідності, та корегує значення попарних порівнянь, у випадку аномалій
- Модуль агрегації глобальних вагів - приймаючи на вхід локальні ваги від усіх експертів, модуль прораховує кінцеві глобальні ваги для кожної альтернативи
- Модуль збереження результатів - зберігає отримані результати у обраний файл користувачем, також демонструє поверхневий результат в інтерфейсі програмного забезпечення

На Рисунку 2.2.2 детально зображено user-flow по інтерфейсу програмного забезпечення

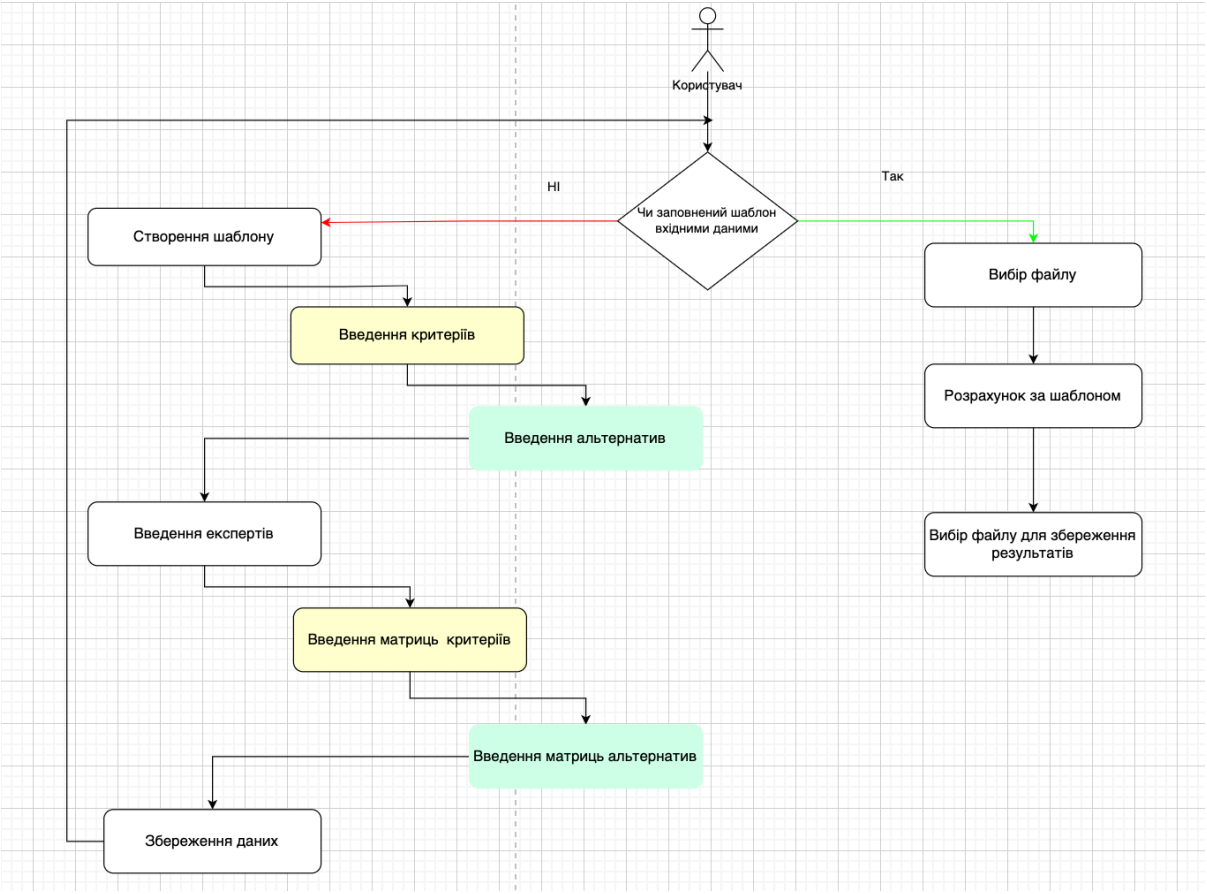

Рисунок 2.2.2 - можливі шляхи користувача в програмі

Вхідні дані включають:

- текстові файли, в яких записані дані суджень експертів
- ручне введення даних суджень експертів

Вихідні дані:

- Найкраща альтернатива, серед запропонованих, з відповідним коефіцієнтом.
- Ранжування альтернатив за кожним критерієм з виведенням значень узгоджених локальних вагів

Чинники, що впливають на роботу застосунку:

- Обрана кількість експертів, критеріїв та альтернатив

Обмеження, що існують, можуть включати:

- Обчислювальна потужність для ефективної роботи алгоритму

Програмний модуль описаний у вигляді IDEF0 (Рисунок 2.2.3)

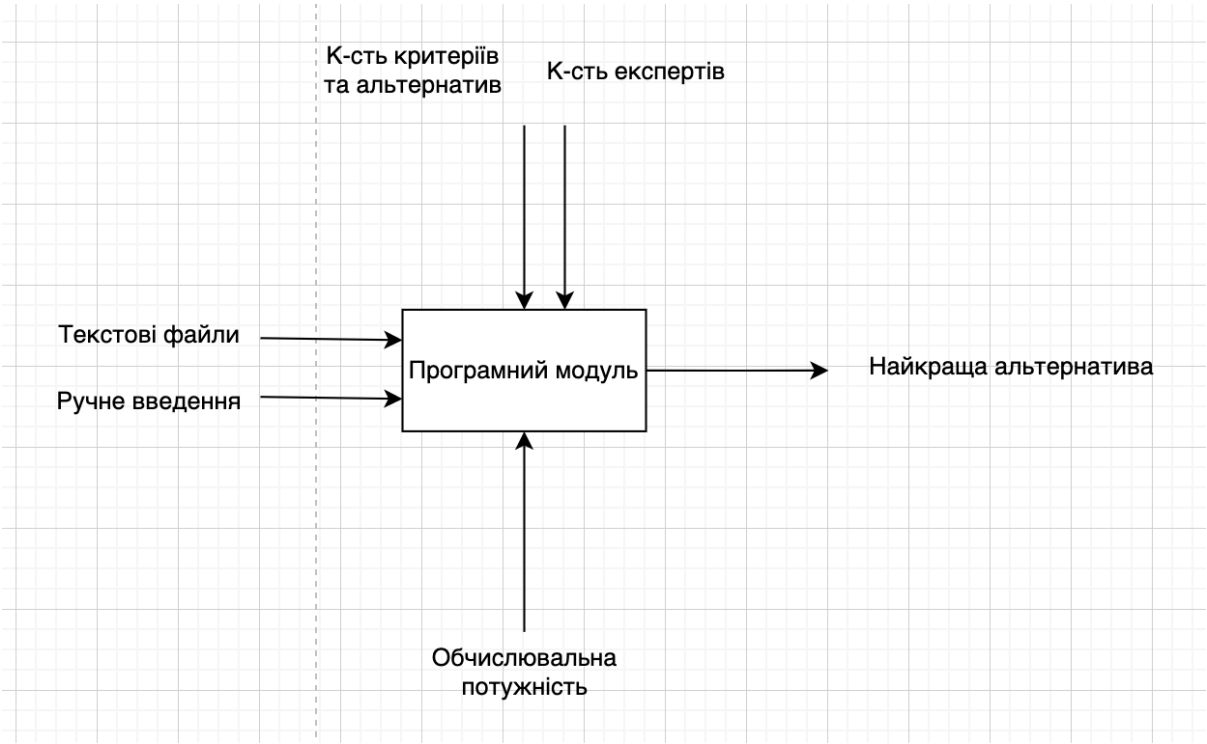

Рисунок 2.2.3 - Опис системи в нотації IDEF0

3.2 Опис інструментів для реалізації програмної частини

Бібліотеки, що будуть використовуватись при розробці системи:

- `json` використовується для роботи з форматом JSON. В цьому коді вона використовується для завантаження та збереження даних у форматі JSON.

- `jsonschema` використовується для валідації JSON-даних за допомогою JSON схеми. У даному коді вона використовується для перевірки, чи відповідають завантажені дані формату, визначеному у JSON схемі.
- `ahpy` (Analytic Hierarchy Process in Python) використовується для реалізації аналітичного ієрархічного процесу (Analytic Hierarchy Process - AHP) у Python. AHP є методом прийняття рішень, який дозволяє моделювати та оцінювати важливість різних критеріїв та альтернатив у процесі прийняття рішень. У цьому коді бібліотека `ahpy` використовується для обчислення оцінок критеріїв та альтернатив, визначення загальних ваг альтернатив та визначення найкращої альтернативи.
- `tkinter` використовується для створення графічного інтерфейсу користувача (GUI) в програмах Python. У цьому коді вона використовується для створення простого GUI, яке містить кнопки, мітки та деревову структуру для відображення результатів. Використовуються класи та функції з `tkinter` для створення та розміщення елементів інтерфейсу на вікні, обробки подій та відображення даних користувачу.
- `filedialog` з модулю `tkinter` використовується для створення діалогового вікна для вибору та збереження файлів. У цьому коді вона використовується для відкриття діалогового вікна для вибору вхідного та вихідного файлів.
- `ttk` з модулю `tkinter` використовується для створення розширених елементів інтерфейсу користувача. У цьому коді вона використовується для створення деревової структури (TreeView), яка використовується для відображення результатів в графічному інтерфейсі.

3.3 Форматування даних для обробки

Вхідні дані включають параметри, критерії, альтернативи та експертні матриці. Параметри вказують на кількість експертів, критеріїв і альтернатив. Критерії представлені списком, включаючи "Ціновий сегмент", "Логістику", "Якість товару", "Різноманіття асортименту" та "Обслуговування". Альтернативи також представлені списком, включаючи "ТОВ 'Меганом'", "Schneider Electric UA", "ПП 'Сігма Кабель'", "ТОВ 'Е.МІР'" та "ТОВ ДП 'Поло електрообладнання'", загалом всі критерії та альтернативи взяті з підрозділу 2.4.

Експертні матриці складаються з матриці критеріїв та матриць альтернатив для кожного експерта. Матриця критеріїв містить порівняння критеріїв за допомогою числових значень. Кожне значення вказує на ступінь переваги одного критерію відносно іншого відповідно до Шкали відношень Сааті (Таблиця 2.1). Схожим чином, матриці альтернатив містять порівняння альтернатив за допомогою числових значень, вказуючи на ступінь переваги однієї альтернативи відносно іншої за кожним критерієм.

Ці дані використовуються для проведення аналізу ієрархій, який допомагає визначити вагові коефіцієнти критеріїв і альтернатив. Шляхом обчислень і узгодження відповідних матриць, визначається найкраща альтернатива відповідно до заданих критеріїв.

Так як процес введення даних в методі аналізу ієрархій займає довготривалий час, був розроблений окремий модуль для введення даних користувачем через інтерфейс, з ціллю збереження даних у файл, для подальшого використання. Детально розглянемо принцип роботи даного модулю:

1) На головному екрані, переходимо у розділ “Створити шаблон”, де обираємо кількість експертів для нашого дослідження, кількість критеріїв, що буде розглянуто при підрахунках, та кількість альтернатив.

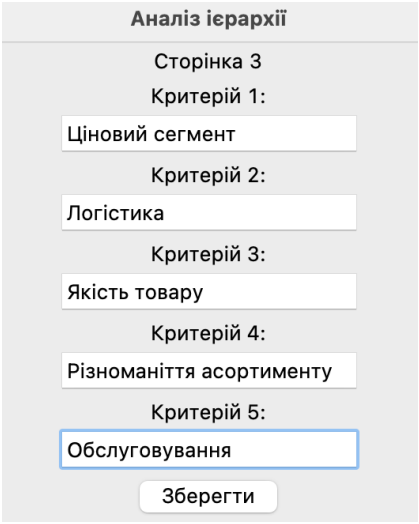

Рисунок 3.2.1 - Інтерфейс введення назв критеріїв, за вже визначеною їх кількістю

2) Після визначення кількості основних параметрів, вводимо назви критерії та альтернатив, для більш зручного виведення

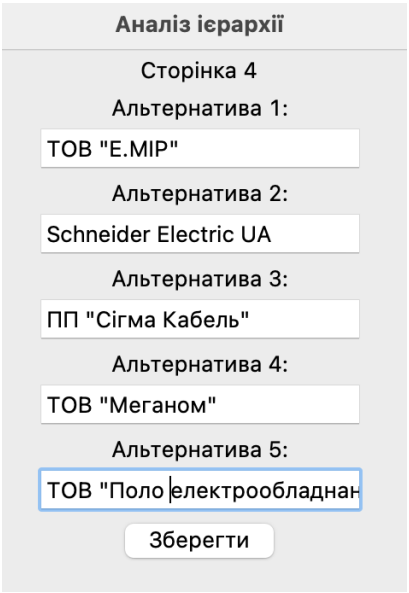

Рисунок 3.2.2 - Інтерфейс введення назв альтернатив, за вже визначеною їх кількістю

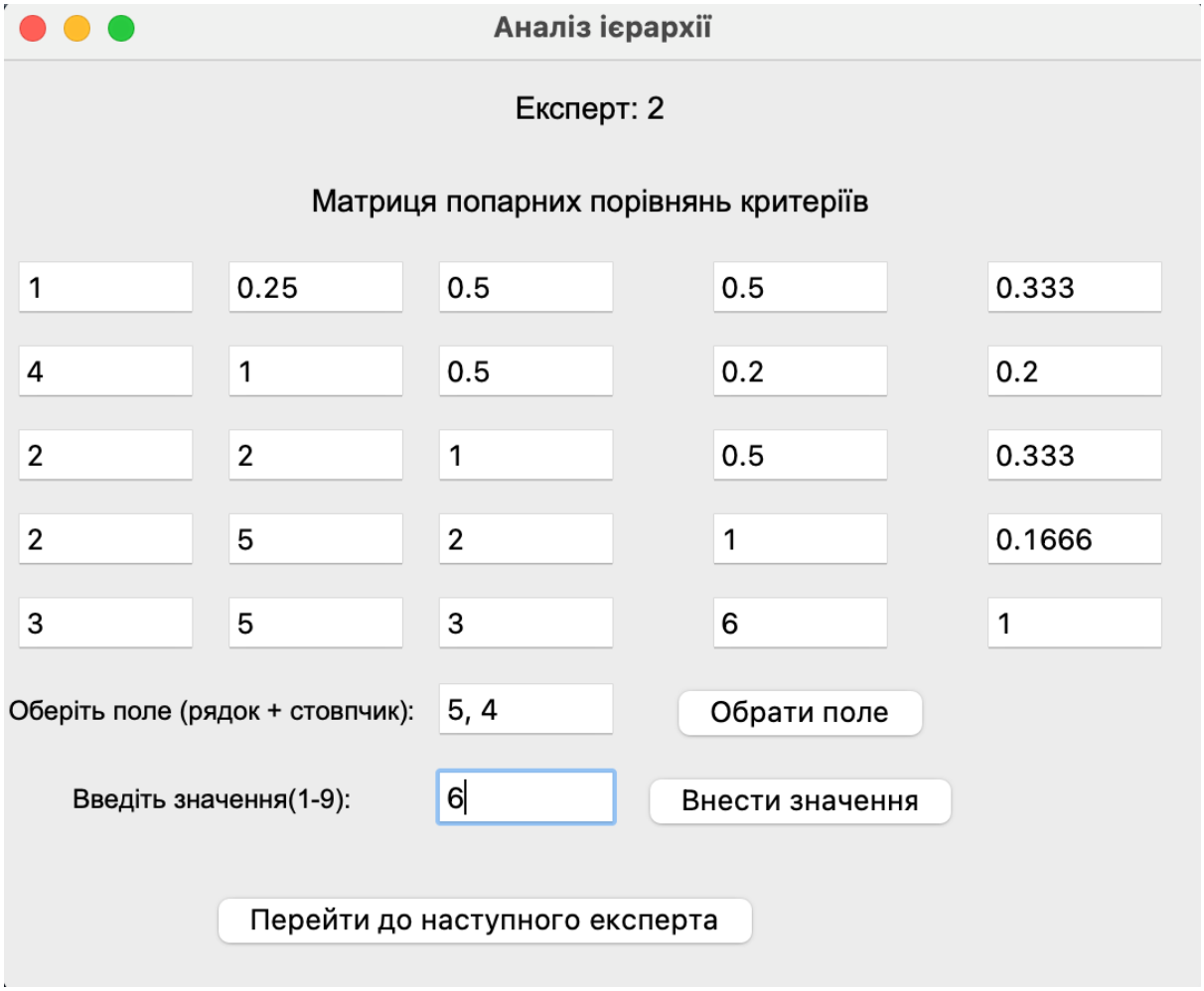

Рисунок 3.2.3 - інтерфейс введення матриці попарних порівнянь критеріїв

3) Надалі, користувачем заповнюються матриці, за правилами побудови матриці попарних порівнянь критеріїв, елементи $\square_{\square,\square}$ *та* $\square_{\square,\square}$ автоматично будуть взаємообернені(користувач не зможе, вводити дані, одночасно в два поля)

4) В результаті користувач отримає json-файл з вже визначеною структурою, що дасть змогу йому відразу скористуватися модулем реалізації методу аналізу ієрархії

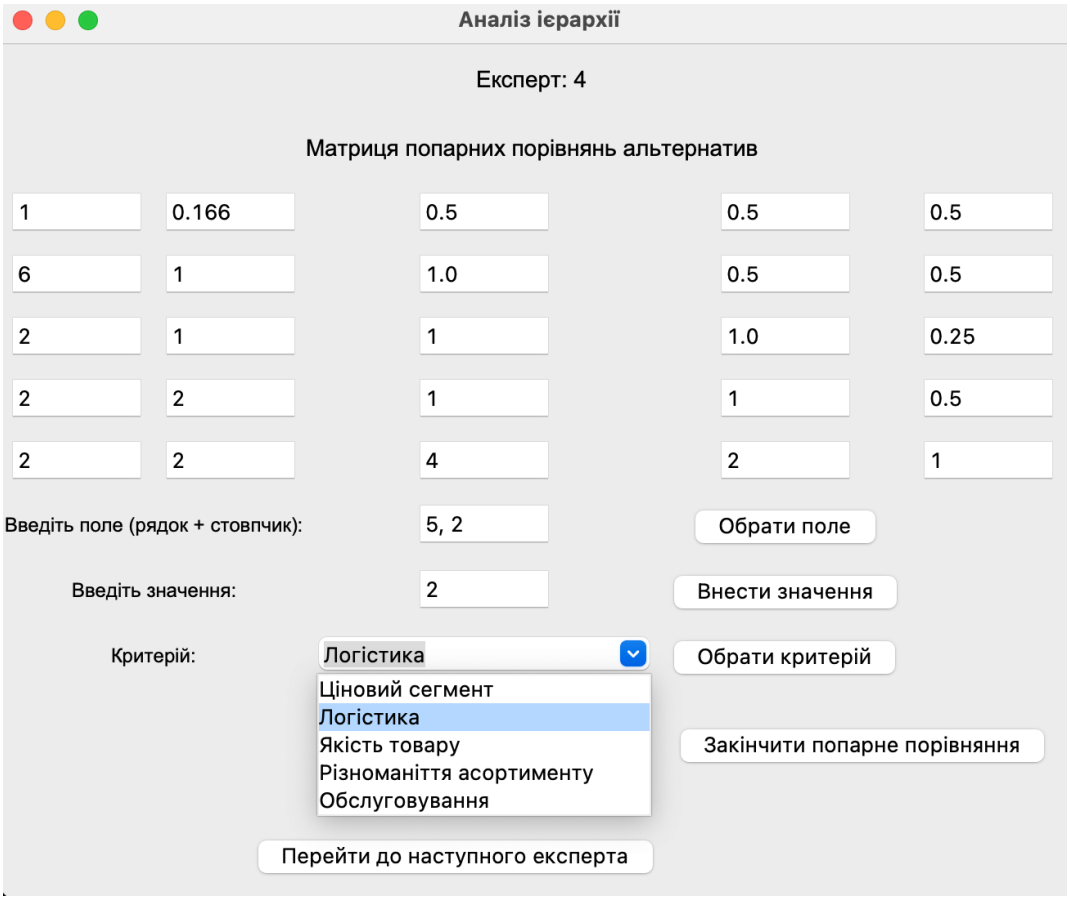

Рисунок 3.2.4 - інтерфейс введення матриці попарних порівнянь альтернатив, з відповідним вибором критеріїв

3.4 Програмна реалізація основних функцій

Код містить програму для обробки даних JSON та розрахунку результатів на основі отриманих матриць і критеріїв. Ми визначаємо JSON схему для валідації даних JSON. Вона описує необхідні поля та їх типи для правильного формату вхідних даних.

Функція open_file відповідальна за відкриття файлу через діалогове вікно вибору, завантаження даних з JSON-файлу та виконання валідації відповідно до заданої схеми. У разі помилки відображається повідомлення про помилку, а в протилежному випадку виводиться перелік критеріїв.

Ядро програми (функція calculate) виконує обчислення на основі отриманих даних JSON. Отримує загальні параметри, матриці критеріїв та альтернатив від кожного експерта, обчислює оцінки критеріїв та альтернатив,

визначає загальні ваги альтернатив та найкращу альтернативу. Результати виводяться на екран та відображаються у деревовій структурі за допомогою ttk.Treeview. Крім того, результати також зберігаються у файлі за допомогою функції save_results.

Функція `calculate` виконує обчислення на основі вхідних даних JSON. Нижче наведено детальну розшифровку роботи функції:

1. Отримання загальних параметрів.

Зчитування значень "experts" (кількість експертів), "criteria" (список критеріїв) та "alternatives" (список альтернатив) з об'єкта "parameters" в JSON-даних.

2. Отримання матриці критеріїв і матриць альтернатив від кожного експерта.

Створення списків "criteria_matrices" та "alternatives_matrices" для збереження матриць порівнянь критеріїв та альтернатив від кожного експерта. Зчитування матриць порівнянь критеріїв і альтернатив з відповідних полів в JSON-даних і додавання їх до відповідних списків.

3. Отримання оцінок критеріїв від кожного експерта і їх узгодження:

Створюється список "expert_criteria" для збереження об'єктів класу `Compare` для оцінок критеріїв від кожного експерта. Далі відбуважться ітерація по матрицях порівнянь критеріїв і створення об'єктів `Compare` для кожного експерта, використовуючи функцію `comparison_from_matrix` (Рисунок 3.4).

Далі йде обчислення середніх оцінок критеріїв шляхом узгодження оцінок від усіх експертів. Результати зберігаються в об'єкті `average_criteria` класу `Compare`.

4. Отримання оцінок альтернатив від кожного експерта і їх узгодження:

- Створення вкладеного списку "expert_alternatives" для збереження списків об'єктів `Compare` для оцінок альтернатив від кожного експерта.

- Ітерація по матрицях порівнянь альтернатив для кожного критерію і створення об'єктів `Compare` для кожного експерта і критерію, використовуючи функцію `comparison_from_matrix`.

- Обчислення середніх оцінок альтернатив для кожного критерію, шляхом узгодження оцінок від усіх експертів. Результати зберігаються в словнику "average_alternatives", де ключами є назви критеріїв, а значеннями - об'єкти `Compare`.

У бібліотеці ahpy (Analytic Hierarchy Process for Python) реалізоване оцінення узгодженості оцінок експертів за допомогою методу контролю узгодженості, відомого як індекс узгодженості Шварца (Consistency Index, CI).

Узгодженість оцінок експертів перевіряється шляхом порівняння парних пріоритетів, які надають експерти елементам у ієрархічній структурі. Кожна пара елементів оцінюється за допомогою числових значень відносної важливості або переваги, які можуть бути задані користувачем. Для оцінення узгодженості розраховується співвідношення між сумарною вагою кожного елемента та його власною вагою у порівнянні з іншими елементами.

Якщо узгодженість є задовільною, оцінка експертів вважається достовірною. Однак, якщо індекс узгодженості перевищує задану межу, це може свідчити про неоднозначність або суперечливість в оцінках, і вимагається подальша обробка оцінок.

ahpy надає зручний інтерфейс для побудови ієрархічних моделей, використання парних порівнянь і розрахунку індексу узгодженості. Основні кроки для оцінки узгодженості оцінок експертів включають:

- Створення об'єкта AHPModel для визначення ієрархії моделі.
- Додавання елементів до моделі за допомогою методу add_node.
- Визначення парних порівнянь між елементами за допомогою методу add_edge та встановлення їх відносних важливостей.
- Виклик методу calculate для розрахунку індексу узгодженості та отримання результату.

Перевірка індексу узгодженості, якщо він перевищує припустиму межу, метод ahpy пропонує у певному відношенні приблизити всі значення даного експерта ближче до середнього.

5. Визначення загальних ваг альтернатив:

- Створення словника "global_weights" для збереження загальних ваг альтернатив.

- Обчислення ваг кожної альтернативи, використовуючи оцінки критеріїв з об'єкта `average_criteria` та оцінки альтернатив зі словника "average_alternatives".

- Результати зберігаються у словнику "global_weights", де ключами є назви альтернатив, а значеннями - їх ваги.

6. Визначення найкращої альтернативи:

Знаходимо альтернативу з найбільшою вагою зі словника "global_weights". Зберігаємо назву найкращої альтернативи у змінну "best_alternative".

Далі формується текстовий результат для виведення та виведення всіх результатів користувачу через графічний інтерфейс. Також для подальшо користування даними, є функція збереження результатів до файлу формату txt. На Рисунку 3.8 можна побачити програмну реалізацію цих етапів мовою Python.

Таким чином, функція calculate виконує всі необхідні обчислення на основі вхідних даних JSON, обробляє матриці порівнянь, обчислює ваги альтернатив та визначає найкращу альтернативу на основі методу аналізу ієрархій.

3.5 Структура інтерфейсу та інструкція для користувача

Вхідних даних від декількох користувачів занадто багато, щоб вводити їх вручну, тож дані краще структурувати у json-файл, за певним шаблоном(Рисунок 3.4.1)(Додаток В).

```
{
  "parameters": {
    "experts": КІЛЬКІСТЬ_ЕКСПЕРТІВ,
    "criteria": КРИТЕРІЇ,
    "alternatives": АЛЬТЕРНАТИВИ
  },
  "criteria": [СПИСОК_КРИТЕРІЇВ],
  "alternatives": [СПИСОК_АЛЬТЕРНАТИВ],
  "expert_matrices": [
    {
      "expert": НОМЕР_ЕКСПЕРТА,
      "criteria_matrix": [[МАТРИЦЯ_ПОРІВНЯННЯ_КРИТЕРІЇВ]],
      "alternatives_matrices": [
        [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],   # ДЛЯ ПЕРШОГО КРИТЕРІЮ
        [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],   # ДЛЯ ДРУГОГО КРИТЕРІЮ
        ...
      ]
    },
    {
      "expert": НОМЕР_ЕКСПЕРТА,
      "criteria_matrix": [[МАТРИЦЯ_ПОРІВНЯННЯ_КРИТЕРІЇВ]],
      "alternatives_matrices": [
        [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],   # ДЛЯ ПЕРШОГО КРИТЕРІЮ
        [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],   # ДЛЯ ДРУГОГО КРИТЕРІЮ
        ...
      ]
    },
    ...
  ]
}
```

Рисунок 3.4.1 - шаблон для введення даних у json-файл

Інструкція по заповненню json-файлу:

- Замініть КІЛЬКІСТЬ_ЕКСПЕРТІВ на ціле число, яке відображає кількість експертів, що беруть участь у процесі прийняття рішень.
- Замініть КРИТЕРІЇ на список критеріїв, які будуть використовуватися для оцінки альтернатив.

- Замініть АЛЬТЕРНАТИВИ на список альтернатив, які будуть порівнюватися за критеріями.
- Для кожного експерта заповніть матрицю порівняння критеріїв (МАТРИЦЯ_ПОРІВНЯННЯ_КРИТЕРІЇВ) та матриці порівняння альтернатив (МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ) для кожного критерію.
- Номер експерта (НОМЕР_ЕКСПЕРТА) повинен бути унікальним для кожного експерта.

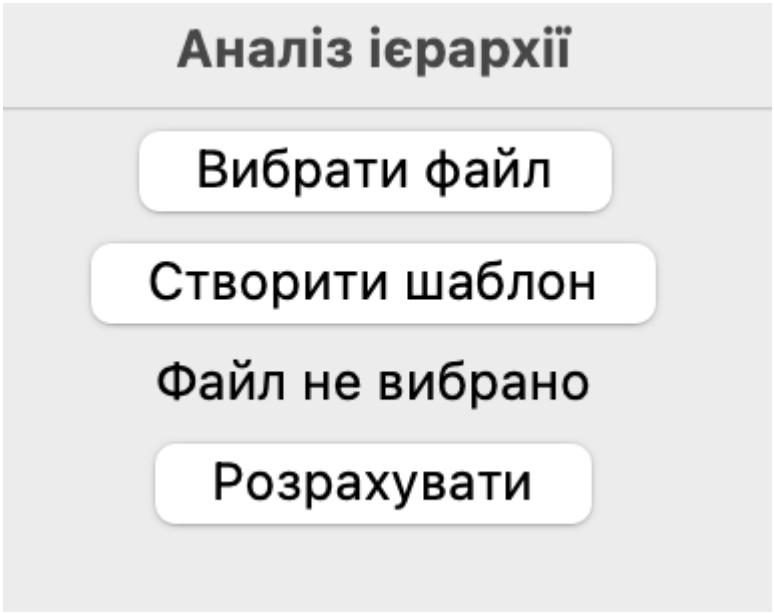

Рисунок 3.4.2 - початковий стан програми

Після створення json-файлу, можна переходити до запуску програми:

1. Натисніть кнопку "Вибрати файл"(Рисунок 3.4.2), щоб відкрити діалогове вікно вибору файлу.
2. Виберіть файл з даними JSON, який містить матриці порівняння критеріїв та альтернатив.
3. Після вибору файлу, ви побачите повідомлення про вибраний файл.

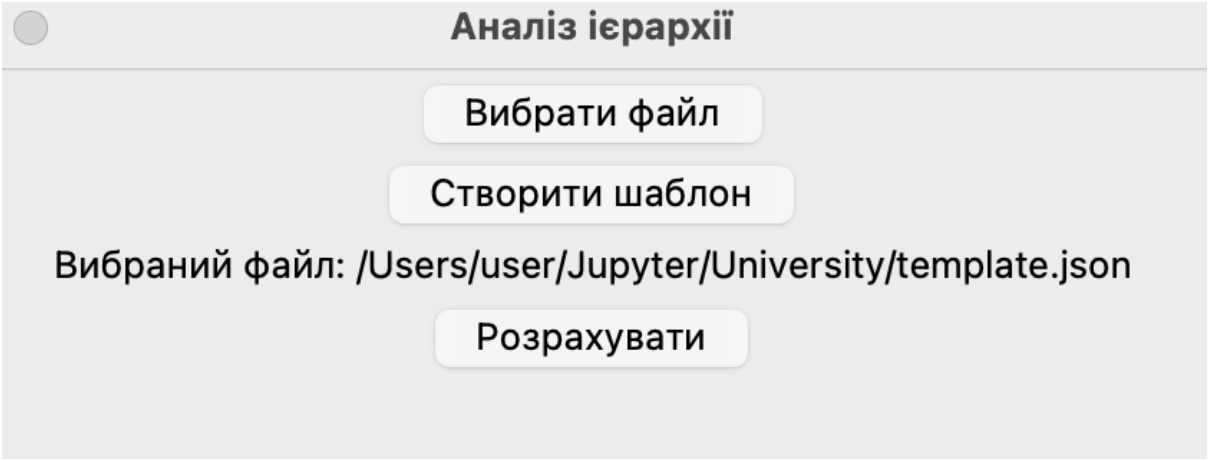

Рисунок 3.4.3 - обраний файл для обробки

4. Натисніть кнопку "Розрахувати", щоб виконати обчислення на основі вхідних даних.
5. Після обчислення програма виведе результати на екрані (Рисунок 3.4.4).
    a. "Найкраща альтернатива" вказує на альтернативу, яка має найвищий показник ваги.
    b. "Ваги альтернатив" показує вагу кожної альтернативи.
6. Деревова структура з'явиться на екрані, де кожен критерій буде мати свої альтернативи(Рисунок 3.4.4).
7. Результати запропонують відразу зберегти у файл за зручним вам текстовим форматом.
8. Вкажіть шлях та ім'я файлу, а потім натисніть кнопку "Зберегти". Результати будуть збережені у вибраний файл(Рисунок 3.4.5).

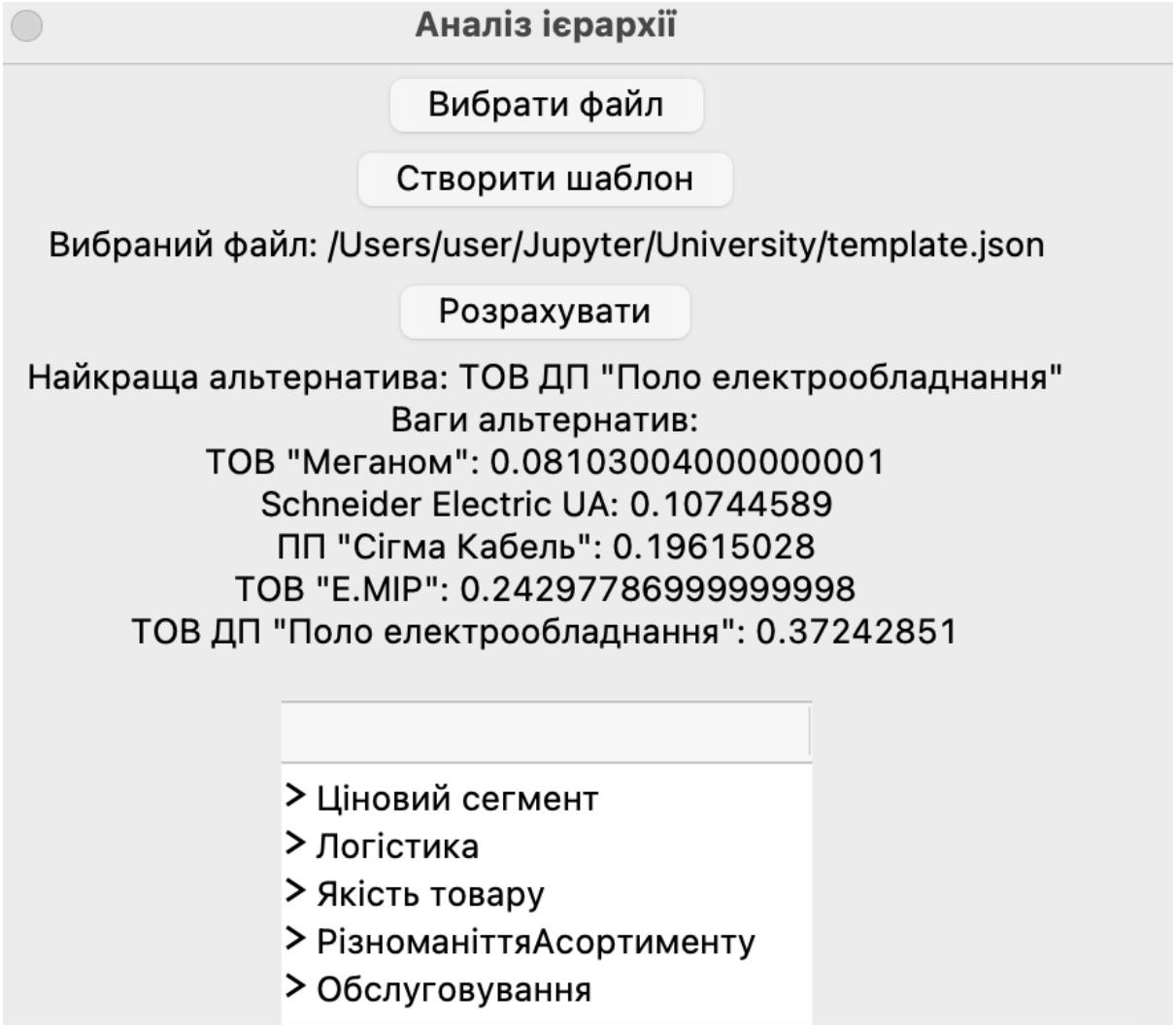

Рисунок 3.4.4 - результат роботи програми виведений на екран

Висновок, що виводиться в результаті роботи програми, містить інформацію про найкращу альтернативу і ваги альтернатив.

Будь ласка, переконайтеся, що дані в файлі JSON відповідають вказаній структурі та формату, щоб уникнути помилок при обробці даних.

```
Найкраща альтернатива: ТОВ ДП "Поло електрообладнання"

Ваги альтернатив:
ТОВ "Меганом": 0.08103004000000001
Schneider Electric UA: 0.10744589
ПП "Сігма Кабель": 0.19615028
ТОВ "Е.МІР": 0.24297786999999998
ТОВ ДП "Поло електрообладнання": 0.37242851

Рейтинг альтернатив за кожним критерієм:

Критерій: Ціновий сегмент
1. ТОВ ДП "Поло електрообладнання": 0.4054
2. ТОВ "Е.МІР": 0.2577
3. ПП "Сігма Кабель": 0.1664
4. Schneider Electric UA: 0.1146
5. ТОВ "Меганом": 0.056

Критерій: Логістика
1. ТОВ ДП "Поло електрообладнання": 0.347
2. ТОВ "Е.МІР": 0.2592
3. ПП "Сігма Кабель": 0.1801
4. Schneider Electric UA: 0.1461
5. ТОВ "Меганом": 0.0676

Критерій: Якість товару
1. ТОВ ДП "Поло електрообладнання": 0.4332
2. ТОВ "Е.МІР": 0.2259
3. ПП "Сігма Кабель": 0.1706
4. Schneider Electric UA: 0.1074
5. ТОВ "Меганом": 0.0629

Критерій: РізноманіттяАсортименту
1. ТОВ ДП "Поло електрообладнання": 0.4111
2. ТОВ "Е.МІР": 0.2501
3. ПП "Сігма Кабель": 0.1713
4. Schneider Electric UA: 0.1
5. ТОВ "Меганом": 0.0676

```

Рисунок 3.4.5 - результат роботи програми записаний у файл формату txt

3.6 Аналіз результатів

Результати роботи програми на основі методу аналізу ієрархії вказують на найкращу альтернативу та надають ваги альтернативам. У даному випадку:

Найкраща альтернатива: ТОВ ДП "Поло електрообладнання"

Ваги альтернатив:

- ТОВ "Меганом": 0.08103004000000001
- Schneider Electric UA: 0.10744589
- ПП "Сігма Кабель": 0.19615028
- ТОВ "E.MIP": 0.24297786999999998
- ТОВ ДП "Поло електрообладнання": 0.37242851

Цей результат означає, що на основі оцінки критеріїв та альтернатив, проведеної експертами, програма визначила, що ТОВ ДП "Поло електрообладнання" є найкращим контрагентом з найвищою вагою 0.37242851. Інші контрагенти також отримали ваги, проте їх значення менші, що свідчить про меншу пріоритетність цих альтернатив.

З іншої сторони, результати програми, отримані за допомогою методу аналізу ієрархії, засновані на суб'єктивних рішеннях експертів. Це означає, що програма визначила ТОВ ДП "Поло електрообладнання" як найкращий варіант з урахуванням наданих ваг альтернатив.

Важливо зазначити, що ця оцінка є обмеженою до вибраного кола клієнтів або експертів, і може варіюватися залежно від унікальних потреб і вимог кожного клієнта. ТОВ ДП "Поло електрообладнання" може бути локально відомою організацією, що спеціалізується на постачанні певних продуктів або послуг в своєму регіоні.

Тому перед прийняттям остаточного рішення щодо вибору контрагента, рекомендується врахувати інші фактори, такі як локальна репутація, якість продукції, ціновий сегмент та логістика. Важливо здійснити додатковий аналіз і зважити на індивідуальні потреби та обмеження вашої компанії перед прийняттям остаточного рішення.

3.7 Висновки до розділу

В даному розділі було розглянуто процес використання програми, яка застосовує метод аналізу ієрархії для вибору контрагента. Були надані вхідні

дані у форматі JSON, які включали параметри, критерії, альтернативи та матриці експертів.

Описано структуру програмного коду, який включав імпорт необхідних бібліотек, валідацію JSON-даних, функції для обчислення порівнянь, завантаження та обробки даних, а також відображення результатів у графічному інтерфейсі.

Надана інструкція користувача включала пояснення процесу заповнення JSON-файлу, вибір файлу для обробки, виконання обчислень та збереження результатів.

Після обчислення програма виводила на екран найкращу альтернативу та ваги альтернатив, а також створювала графічне представлення даних у вигляді дерева. Результати також могли бути збережені у текстовому файлі.

В аналізі результату було зазначено, що ТОВ ДП "Поло електрообладнання" було визначено як найкращий варіант з урахуванням ваг альтернатив застосованого методу аналізу ієрархії. Однак, було відзначено, що ця оцінка є суб'єктивною та обмеженою до вибраного кола клієнтів. Рекомендувалося додатково враховувати інші фактори та проводити додатковий аналіз перед прийняттям остаточного рішення щодо вибору контрагента.

Висновки

Засновуючись на проведених дослідженнях та розробці програмного модулю, можна зробити наступні висновки. Перш за все, розробка системи для вибору контрагента є надзвичайно актуальним завданням, що має великий потенціал для вдосконалення процесів у сфері ремонтних служб енергетичної інфраструктури. Використання методу аналізу ієрархій (MAI) явно продемонструвало свою ефективність як надійного інструменту для об'єктивного та систематичного прийняття рішень у процесі вибору контрагентів. Отримані результати досліджень та розробки, узгоджені зі станом сучасних досліджень, відкривають нові перспективи для поліпшення управлінських рішень та оптимізації використання ресурсів в галузі ремонтних служб енергетичної інфраструктури. Таким чином, дана робота є значним внеском у вирішення проблеми вибору контрагентів та має потенціал для практичного впровадження, сприяючи раціональному та ефективному управлінню ремонтними процесами в енергетичній інфраструктурі.

Джерела

1) https://dss.tg.ck.ua/ahp-help

2) https://docs.google.com/presentation/d/1OxQaZAGaelKnsVF3eUhvcdyObggGEvST/edit?usp=share_link&ouid=113982203120092786507&rtpof=true&sd=true

3) https://dss.tg.ck.ua/2019/06/22/3137

4) Ranking of risks for the existing and new building works [Електронний ресурс] / Chau Kwon Wing, Frankie Fanjie Zeng] – Режим доступу до ресурсу: https://www.researchgate.net/publication/333078282_Ranking_of_risks_for_the_existing_and_new_building_works

5) А. Т. Яровий, Є. М. Страхов «Багатовимірний статистичний аналіз»

6) В. Є. Бахрушин «Методи аналізу даних»

7) А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий «Програмування числових методів мовою Python»

8) https://pdt.tools/o-predprijatii/

9) https://github.com/PhilipGriffith/AHPy

10) On optimal completion of incomplete pairwise comparison matrices [Електронний ресурс] /Sándor Bozóki, János Fülöp, Lajos Rónyai] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S0895717710001159?via%3Dihub

11) A note on saaty's random indexes [Електронний ресурс] / H.A Donegan, F.J Dodd] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/089571779190098R?via%3Dihub

12) Decision making with the analytic hierarchy process [Електронний ресурс] /Thomas Saaty] – Режим доступу до ресурсу: https://www.inderscienceonline.com/doi/pdf/10.1504/IJSSCI.2008.017590

13) https://pdt.tools/

14) A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming [Електронний ресурс] /Sysed Hassan Ghodsypour, Christopher O'Brien] – Режим доступу до ресурсу: https://www.researchgate.net/publication/223291266_A_decision_support_system_for_supplier_selection_using_an_integrated_analytic_hierarchy_process_and_linear_programming

15) Вибір постачальників товарних ресурсів підприємства торгівлі [Електронний ресурс] Новікова Н.М.. Режим доступу до ресурсу: http://pev.kpu.zp.ua/journals/2018/4_09_uk/10.pdf

16) Supplier selection problem: A comparison of the total cost of ownership and analytic hierarchy process approaches [Електронний ресурс] M. Khurrum S. Bhutta, Faizul Huq. Режим доступу до ресурсу: https://www.researchgate.net/publication/235320999_Supplier_selection_problem_A_comparison_of_the_total_cost_of_ownership_and_analytic_hierarchy_process_approaches

17) Supplier Selection Using Multi-objective Programming: A Decision Support System Approach [Електронний ресурс] Charles Weber, Lisa Ellram. Режим доступу до ресурсу: https://www.emerald.com/insight/content/doi/10.1108/09600039310038161/full/html

18) Supplier Selection Criteria and Methods in Supply Chains: A Review / Om Pal, Amit Kumar Gupta, R. K. Garg.

19) http://pyanp.org/tutorials/limitmatrix.html

Додатки

Додаток А. Лістинг програми

```
import tkinter as tk
from tkinter import *
import json
import jsonschema
from ahpy import Compare
from tkinter import filedialog, ttk


def show_next_page():
    # Видаляємо всі віджети на поточній сторінці
    for widget in root.winfo_children():
        widget.destroy()

def check_entry_not_empty(entry):
    entry_value = entry.get()
    if not entry_value:
        return False
    else:
        return True

def create_window_issue():
    window = tk.Toplevel()
    window.title("Помилка!!")
    window.geometry("400x50")

    button = tk.Button(window, text="Close Window", command=window.destroy)
    button.pack()

def prepare_alternative_matrices(alternatives_matrices, alternatives_amount, criterias_amount):
    for i in range(criterias_amount):
        for j in range(alternatives_amount):
            for k in range(alternatives_amount):
                if j==k:
                    alternatives_matrices[i][j][k]=1
                else:
                    alternatives_matrices[i][j][k]=0
    return alternatives_matrices
```

```
        def check_alternative_matrices(alternatives_matrices, alternatives_amount,
criterias_amount):
            for i in range(criterias_amount):
                for j in range(alternatives_amount):
                    for k in range(alternatives_amount):
                        if alternatives_matrices[i][j][k]==0:
                            print("i fall in check: ",i,j,k)
        #                     create_window_issue()
                            return False
            return True

        def create_template_page():
            show_next_page()

            parameters = {}
            parameters["experts"] = 0
            parameters["criterias"] = 0
            parameters["alternatives"] = 0

            experts_label = tk.Label(root, text="Введіть к-сть експертів")
            experts_label.pack()
            experts_entry = tk.Entry(root)
            experts_entry.pack()

            criterias_label = tk.Label(root, text="Введіть к-сть критеріїв")
            criterias_label.pack()
            criterias_entry = tk.Entry(root)
            criterias_entry.pack()

            alternatives_label = tk.Label(root, text="Введіть к-сть альтернатив")
            alternatives_label.pack()
            alternatives_entry = tk.Entry(root)
            alternatives_entry.pack()

            button = tk.Button(root, text="Перевірити дані", command=lambda:
check_parameters_validity(
                parameters,
                experts_entry,
                criterias_entry,
                alternatives_entry))
            button.pack()
```

```
        def
check_parameters_validity(parameters,experts_entry,criterias_entry,alternatives_e
ntry):
            experts_amount = int(experts_entry.get())
            criterias_amount = int(criterias_entry.get())
            alternatives_amount = int(alternatives_entry.get())
            if ((experts_amount<=0) or (criterias_amount<=0) or
(alternatives_amount<=0)):
                issue_label = tk.Label(root, text="Введені дані некоректні(значення
менше або дорівнює нулю)")
                issue_label.pack()
            else:
                parameters["experts"] = experts_amount
                parameters["criterias"] = criterias_amount
                parameters["alternatives"] = alternatives_amount
                print("experts_amount: ",experts_amount)
                print("criterias_amount: ",criterias_amount)
                print("alternatives_amount: ",alternatives_amount)
                button_ready = tk.Button(root, text="Далі", command=lambda:
entry_criterias_page(
                parameters))
                button_ready.pack()

        def create_button_create_template():
        #     show_next_page()
            button = tk.Button(root, text="Створити шаблон",
command=create_template_page)
            button.pack()

        def entry_criterias_page(parameters):
            show_next_page()
            data = {}
            data["parameters"] = parameters
            criterias_amount = data["parameters"]["criterias"]
            criteria_entries = {}

            for i in range(criterias_amount):
                label_text = 'Введіть назву критерія {}'.format(i+1)
                label = tk.Label(root, text=label_text)
                label.pack()

                entry = tk.Entry(root)
                entry.pack()
```

```
            criteria_entries[i] = entry

        button = tk.Button(root, text="Перевірити критерії", command=lambda:
check_criterias_names_validity(
            criteria_entries,
            data))
        button.pack()


    def check_criterias_names_validity(criteria_entries, data):
        criterias_amount = data["parameters"]["criterias"]
        criteria_values = [''] * criterias_amount
        criterias_bools = [False] * criterias_amount
        for i in range(criterias_amount):
            criterias_bools[i] = check_entry_not_empty(criteria_entries[i])
            criteria_values[i] = criteria_entries[i].get()

        if False in criterias_bools:
            issue_label = tk.Label(root, text="Введені дані некоректні(присутнє
пусте поле)")
            issue_label.pack()
        else:
            data["criterias"] = criteria_values
            print(data["criterias"])
            button_ready = tk.Button(root, text="Далі", command=lambda:
entry_alternatives_page(
            data))
            button_ready.pack()



    def entry_alternatives_page(data):
        show_next_page()

        alternatives_amount = data["parameters"]["alternatives"]
        alternatives_entries = {}

        for i in range(alternatives_amount):
            label_text = 'Введіть назву альтернативи {}'.format(i+1)
            label = tk.Label(root, text=label_text)
            label.pack()

            entry = tk.Entry(root)
```

```
            entry.pack()
            alternatives_entries[i] = entry

        button = tk.Button(root, text="Перевірити альтернативи", command=lambda: check_alternatives_names_validity(
            alternatives_entries,
            data))
        button.pack()


    def check_alternatives_names_validity(alternatives_entries, data):
        alternatives_amount = data["parameters"]["alternatives"]
        alternatives_values = ['] * alternatives_amount
        alternatives_bools = [False] * alternatives_amount
        for i in range(alternatives_amount):
            alternatives_bools[i] = check_entry_not_empty(alternatives_entries[i])
            alternatives_values[i] = alternatives_entries[i].get()

        if False in alternatives_bools:
            issue_label = tk.Label(root, text="Введені дані некоректні(присутнє пусте поле)")
            issue_label.pack()
        else:
            data["alternatives"] = alternatives_values
            print(data["alternatives"])
            data = create_pattern_for_expert_matrices(data)

            button_ready = tk.Button(root, text="Далі", command=lambda: fill_expert_criterias_matrix(
            data,
            0))
            button_ready.pack()

    def create_pattern_for_expert_matrices(data):
        expert_matrices = [{} for _ in range(data["parameters"]["experts"])]
        data["expert_matrices"] = expert_matrices
        for i in range(data["parameters"]["experts"]):
            data["expert_matrices"][i]["expert"] = i+1
        return data
```

```
        def fill_expert_criterias_matrix(data, expert_id):
            show_next_page()
            criterias_amount = data["parameters"]["criterias"]
            criterias_names = data["criterias"]


            label_text = 'Експерт: {}'.format(expert_id+1)
            expert_label = tk.Label(root, text = label_text)
            expert_label.pack()

        #     label_matrix = [[{}]*criterias_amount]*criterias_amount
            label_matrix = [[None for _ in range(criterias_amount)] for _ in
range(criterias_amount)]


            name_label = tk.Label(root, text = "Матриця попарних порівнянь
критеріїв")
            name_label.pack()

        #     for i in range(data["parameters"]["experts"]):
        #         label_name_criteria = tk.Label(root, text=data["criterias"][i])
        #         label_name_criteria.pack(side=tk.LEFT)

            for i in range(criterias_amount):
                row_frame = tk.Frame(root)
                row_frame.pack()

                for j in range(criterias_amount):
                    if i==j:
                        label_matrix[i][j] = tk.Label(row_frame, text = str(1), borderwidth
= 1, relief = "solid", width = 5)
                        label_matrix[i][j].pack(side = tk.LEFT)
                    else:
                        label_matrix[i][j] = tk.Label(row_frame, text = str(0), borderwidth
= 1, relief = "solid", width = 5)
                        label_matrix[i][j].pack(side = tk.LEFT)



            label_warning1 = tk.Label(root, text = "Правило 1: Значення для
введення мають бути в проміжку 1-9")
            label_warning1.pack()
```

```
        label_warning2 = tk.Label(root, text = "Правило 2: Не можна порівнювати однакові елементи")
        label_warning2.pack()
        label_warning3 = tk.Label(root, text = "Правило 3: При перевірці в матриці не може бути значення 0")
        label_warning3.pack()

        label_dropdown = tk.Label(root, text = "Оберіть перший критерій для порівняння зі списку")
        label_dropdown.pack()
        dropdown_first_var = tk.StringVar(root)
        dropdown_first = tk.OptionMenu(root, dropdown_first_var, *criterias_names)
        dropdown_first.pack()

        label_dropdown = tk.Label(root, text = "Оберіть другий критерій для порівняння зі списку")
        label_dropdown.pack()
        dropdown_second_var = tk.StringVar(root)
        dropdown_second = tk.OptionMenu(root, dropdown_second_var, *criterias_names)
        dropdown_second.pack()

        entry = tk.Entry(root)
        entry.pack()

        button_set_value = tk.Button(root, text = "Вставити значення", command = lambda:set_value_to_matrix(
            criterias_names,
            row_frame,
            dropdown_first_var,
            dropdown_second_var,
            entry,
            label_matrix))
        button_set_value.pack()


        button_check_matrix = tk.Button(root, text = "Перевірити матрицю", command = lambda:check_matrix_and_go_further(
            data,
            row_frame,
            label_matrix,
            expert_id))
```

```
        button_check_matrix.pack()


    def check_matrix_and_go_further(data, row_frame, label_matrix, expert_id):
        criterias_amount = data["parameters"]["criterias"]
        expert_criterias_matrix = [[None for _ in range(criterias_amount)] for _ in range(criterias_amount)]
        for i in range(criterias_amount):
            for j in range(criterias_amount):
                expert_criterias_matrix[i][j] = float(label_matrix[i][j]["text"])
                if expert_criterias_matrix[i][j]==0:
                    create_window_issue()
                    return
        data["expert_matrices"][expert_id]["criteria_matrix"] = expert_criterias_matrix
        if data["parameters"]["experts"]!=expert_id+1:
            fill_expert_criterias_matrix(data, expert_id+1)
        else:
            print(data)
            alternatives_amount = data["parameters"]["alternatives"]
            alternatives_matrices = [[[None for _ in range(alternatives_amount)] for _ in range(alternatives_amount)] for _ in range(criterias_amount)]
            alternatives_matrices = prepare_alternative_matrices(alternatives_matrices,alternatives_amount,criterias_amount)
            fill_expert_alternatives_matrix(data,0,0,alternatives_matrices)

    def set_value_to_matrix(names, row_frame, dropdown_first_var, dropdown_second_var, entry, label_matrix):
        first_name = dropdown_first_var.get()
        first_name_index = names.index(first_name)
        second_name = dropdown_second_var.get()
        second_name_index = names.index(second_name)

        print("first_name: ", first_name)
        print("first_name_index: ", first_name_index)
        print("second_name: ", second_name)
        print("second_name_index: ", second_name_index)
        entry_value = int(entry.get())

        if first_name_index==second_name_index or entry_value<=0 or entry_value>9 or entry=="":
```

```
            create_window_issue()
        else:
            label_matrix[first_name_index][second_name_index]["text"] =
str(entry_value)
            label_matrix[second_name_index][first_name_index]["text"] =
str(round(1/entry_value,3))




    def fill_expert_alternatives_matrix(data, expert_id, criteria_id,
alternatives_matrices):
        show_next_page()
        criterias_amount = data["parameters"]["criterias"]
        criterias_names = data["criterias"]
        alternatives_amount = data["parameters"]["alternatives"]
        alternatives_names = data["alternatives"]


        label_text = 'Експерт: {}'.format(expert_id+1)
        expert_label = tk.Label(root, text = label_text)
        expert_label.pack()

    #    label_matrix = [[{}]*criterias_amount]*criterias_amount
        label_matrix = [[None for _ in range(criterias_amount)] for _ in
range(alternatives_amount)]


        name_label = tk.Label(root, text = "Матриця попарних порівнянь
альтернатив")
        name_label.pack()

    #    for i in range(data["parameters"]["experts"]):
    #        label_name_criteria = tk.Label(root, text=data["criterias"][i])
    #        label_name_criteria.pack(side=tk.LEFT)

        for i in range(alternatives_amount):
            row_frame = tk.Frame(root)
            row_frame.pack()

            for j in range(alternatives_amount):
                if i==j:
```

```
                    label_matrix[i][j] = tk.Label(row_frame, text = str(1), borderwidth
= 1, relief = "solid", width = 5)
                    label_matrix[i][j].pack(side = tk.LEFT)
                else:
                    label_matrix[i][j] = tk.Label(row_frame, text = str(0), borderwidth
= 1, relief = "solid", width = 5)
                    label_matrix[i][j].pack(side = tk.LEFT)



        label_warning1 = tk.Label(root, text = "Правило 1: Значення для
введення мають бути в проміжку 1-9")
        label_warning1.pack()
        label_warning2 = tk.Label(root, text = "Правило 2: Не можна
порівнювати однакові елементи")
        label_warning2.pack()
        label_warning3 = tk.Label(root, text = "Правило 3: При перевірці в
матриці не може бути значення 0")
        label_warning3.pack()

        label_dropdown = tk.Label(root, text = "Оберіть першу альтернативу
для порівняння зі списку")
        label_dropdown.pack()
        dropdown_first_var = tk.StringVar(root)
        dropdown_first = tk.OptionMenu(root, dropdown_first_var,
*alternatives_names)
        dropdown_first.pack()

        label_dropdown = tk.Label(root, text = "Оберіть другу альтернативу
для порівняння зі списку")
        label_dropdown.pack()
        dropdown_second_var = tk.StringVar(root)
        dropdown_second = tk.OptionMenu(root, dropdown_second_var,
*alternatives_names)
        dropdown_second.pack()

        entry = tk.Entry(root)
        entry.pack()

        button_set_value = tk.Button(root, text = "Вставити значення",
command = lambda:set_value_to_matrix(
            alternatives_names,
            row_frame,
```

```
            dropdown_first_var,
            dropdown_second_var,
            entry,
            label_matrix))
        button_set_value.pack()

        label_dropdown = tk.Label(root, text = "Задля перевірки матриці оберіть інший критерій зі списку, та натисніть кнопку 'Обрати критерій'")
        label_dropdown.pack()
        dropdown_third_var = tk.StringVar(root)
        dropdown_third_var.set(criterias_names[criteria_id])
        dropdown_third = tk.OptionMenu(root, dropdown_third_var, *criterias_names)
        dropdown_third.pack()
        label_dropdown = tk.Label(root, text = "Правило 4: Не можна перемкнутися на той самий критерій")
        label_dropdown.pack()


        button_check_matrix = tk.Button(root, text = "Перевірити матрицю", command = lambda:check_alternative_matrix_and_go_further(
            data,
            row_frame,
            label_matrix,
            expert_id,
            criteria_id,
            alternatives_matrices,
            dropdown_third_var))
        button_check_matrix.pack()

        label_dropdown = tk.Label(root, text = "Під час переходу до наступного експерта, відбудеться перевірка всіх матриць на наявність нулів")
        label_dropdown.pack()
        button_next_alternative_expert = tk.Button(root, text = "Перейти до наступного експерта", command = lambda:next_alternative_expert(
            data,
            row_frame,
            label_matrix,
            expert_id,
            criteria_id,
            alternatives_matrices))
        button_next_alternative_expert.pack()
```

```
        def next_alternative_expert(data,
                                    row_frame,
                                    label_matrix,
                                    expert_id,
                                    criteria_id,
                                    alternatives_matrices):
            criterias_amount = data["parameters"]["criterias"]
            alternatives_amount = data["parameters"]["alternatives"]
            expert_alternative_matrix = [[None for _ in range(criterias_amount)] for
_ in range(criterias_amount)]
            for i in range(alternatives_amount):
                for j in range(alternatives_amount):
                    expert_alternative_matrix[i][j] = float(label_matrix[i][j]["text"])
                    if expert_alternative_matrix[i][j]==0:
                        create_window_issue()
                        return
            alternatives_matrices[criteria_id] = expert_alternative_matrix
            print(alternatives_matrices)

            is_alternatives_matrices_ready                                    =
check_alternative_matrices(alternatives_matrices,alternatives_amount,criterias_am
ount)
            if is_alternatives_matrices_ready:
                data["expert_matrices"][expert_id]["alternatives_matrices"]    =
alternatives_matrices
                if data["parameters"]["experts"]!=expert_id+1:
                    alternatives_matrices = [[[None for _ in range(alternatives_amount)]
for _ in range(alternatives_amount)] for _ in range(criterias_amount)]
                    alternatives_matrices                                      =
prepare_alternative_matrices(alternatives_matrices,alternatives_amount,criterias_a
mount)
                    fill_expert_alternatives_matrix(data,
expert_id+1,0,alternatives_matrices)
                else:
                    save_data_to_json(data)
            else:
                print("i fall in next expert function")
                create_window_issue()
                return
```

```
        def check_alternative_matrix_and_go_further(data,
                                                    row_frame,
                                                    label_matrix,
                                                    expert_id,
                                                    criteria_id,
                                                    alternatives_matrices,
                                                    dropdown_third_var):
            criterias_amount = data["parameters"]["criterias"]
            alternatives_amount = data["parameters"]["alternatives"]
            expert_alternative_matrix = [[None for _ in range(criterias_amount)] for
_ in range(criterias_amount)]
            for i in range(alternatives_amount):
                for j in range(alternatives_amount):
                    expert_alternative_matrix[i][j] = float(label_matrix[i][j]["text"])
                    if expert_alternative_matrix[i][j]==0:
                        create_window_issue()
                        return


            next_criteria_name = dropdown_third_var.get()
            next_criteria_index = data["criterias"].index(next_criteria_name)
            if criteria_id == next_criteria_index:
                create_window_issue()
                return
            else:
                alternatives_matrices[criteria_id] = expert_alternative_matrix
                criteria_id = next_criteria_index
                fill_expert_alternatives_matrix(data, expert_id, criteria_id,
alternatives_matrices)




        def set_value_to_matrix(names, row_frame, dropdown_first_var,
dropdown_second_var, entry, label_matrix):
            first_name = dropdown_first_var.get()
            first_name_index = names.index(first_name)
            second_name = dropdown_second_var.get()
            second_name_index = names.index(second_name)

            print("first_name: ", first_name)
```

```
        print("first_name_index: ", first_name_index)
        print("second_name: ", second_name)
        print("second_name_index: ", second_name_index)
        entry_value = int(entry.get())

        if first_name_index==second_name_index or entry_value<=0 or
entry_value>9 or entry==":
            create_window_issue()
        else:
            label_matrix[first_name_index][second_name_index]["text"]          =
str(entry_value)
            label_matrix[second_name_index][first_name_index]["text"]          =
str(round(1/entry_value,3))


    def save_data_to_json(data):

        file_path = filedialog.asksaveasfilename(defaultextension='.json',
                                                 filetypes=[('JSON Files', '*.json')],
                                                 title='Save JSON File')

        if file_path:
            with open(file_path, 'w') as file:
                json.dump(data, file)
                print("JSON file saved successfully.")
        go_to_menu()




    # JSON schema для валідації
    schema = {
        "type": "object",
        "properties": {
            "parameters": {"type": "object"},
            "criterias": {"type": "array"},
            "alternatives": {"type": "array"},
            "expert_matrices": {"type": "array"},
        },
        "required": ["parameters", "criterias", "alternatives", "expert_matrices"]
    }
```

```
        def comparison_from_matrix(matrix, elements):
            comparisons = {}
            for i in range(len(matrix)):
                for j in range(i + 1, len(matrix[i])):
                    comparisons[(elements[i], elements[j])] = matrix[i][j]
            return comparisons

        def open_file():
            filepath = filedialog.askopenfilename()
            try:
                with open(filepath, "r") as file:
                    global data
                    data = json.load(file)
                    # виконати валідацію JSON
                    jsonschema.validate(data, schema)
                    print(criteria)
            except Exception as e:
                print(f"Помилка при відкритті файлу: {e}")
            file_label.config(text=f"Вибраний файл: {filepath}")


        def calculate():
            # Отримання загальних параметрів
            experts = data["parameters"]["experts"]
            global criteria
            criteria = data["criterias"]
            global alternatives
            alternatives = data["alternatives"]

            # Отримання матриці критеріїв і матриць альтернатив від кожного експерта
            criteria_matrices = [expert_matrix["criteria_matrix"] for expert_matrix in data["expert_matrices"]]
            alternatives_matrices = [expert_matrix["alternatives_matrices"] for expert_matrix in data["expert_matrices"]]

            # Отримання оцінок критеріїв від кожного експерта і їх узгодження
            expert_criteria = [
                Compare("Criteria", comparison_from_matrix(matrix, criteria)) for matrix in criteria_matrices
            ]
            average_criteria = Compare(
                "Criteria",
```

```
            {
                name: sum(c.target_weights[name] for c in expert_criteria) / len(expert_criteria)
                for name in expert_criteria[0].target_weights.keys()
            },
        )

        # Отримання оцінок альтернатив від кожного експерта і їх узгодження
        expert_alternatives = [
            [
                Compare(criteria[i], comparison_from_matrix(matrix, alternatives))
                for i, matrix in enumerate(matrices)
            ]
            for matrices in alternatives_matrices
        ]
        global average_alternatives
        average_alternatives = {
            crit_name: Compare(
                crit_name,
                {
                    name: sum(e.target_weights[name] for e in experts) / len(experts)
                    for name in experts[0].target_weights.keys()
                },
            )
            for crit_name, experts in zip(criteria, zip(*expert_alternatives))
        }

        # Визначення загальних ваг альтернатив
        global global_weights
        global_weights = {
            alt: sum(
                crit_weight * average_alternatives[crit].target_weights[alt]
                for crit, crit_weight in average_criteria.target_weights.items()
            )
            for alt in alternatives
        }

        # Визначення найкращої альтернативи
        global best_alternative
        best_alternative = max(global_weights, key=global_weights.get)
```

```
        result_text = f"Найкраща альтернатива: {best_alternative}\nВаги альтернатив:\n"
        for alt, weight in global_weights.items():
            result_text += f"{alt}: {weight}\n"
        result_label.config(text=result_text)

        tree = ttk.Treeview(root)
        tree.pack()

        for i, criteria in enumerate(data["criterias"]):
            tree.insert("", "end", "criterias" + str(i), text=criteria)
            for j, alternative in enumerate(data["alternatives"]):
                tree.insert("criterias" + str(i), "end", text=alternative)

        # Збереження результатів у файл
        save_results()

    # def save_results():
    #     """
    #     Ця функція зберігає результати в файл.
    #     """
    #     filepath = filedialog.asksaveasfilename(defaultextension=".txt",
    #                                             filetypes=(("Text Files", "*.txt"), ("All Files", "*.*")))
    #     try:
    #         with open(filepath, "w") as file:
    #             file.write(f"Найкраща альтернатива: {best_alternative}\n\n")
    #             file.write("Ваги альтернатив:\n")
    #             for alt, weight in global_weights.items():
    #                 file.write(f"{alt}: {weight}\n")
    #         print(f"Результати збережено у файлі: {filepath}")
    #     except Exception as e:
    #         print(f"Помилка при збереженні результатів: {e}")
    def save_results():
        """
        Ця функція зберігає результати в файл, включаючи рейтинг альтернатив за глобальними вагами
        та за кожним критерієм.
        """
        filepath = filedialog.asksaveasfilename(defaultextension=".txt",
                                                filetypes=(("Text Files", "*.txt"), ("All Files", "*.*")))
        try:
```

```
            with open(filepath, "w") as file:
                file.write(f"Найкраща альтернатива: {best_alternative}\n\n")
                file.write("Ваги альтернатив:\n")
                for alt, weight in global_weights.items():
                    file.write(f"{alt}: {weight}\n")

                # Рейтинг альтернатив за кожним критерієм
                file.write("\nРейтинг альтернатив за кожним критерієм:\n")
                for crit_name, crit_experts in average_alternatives.items():
                    file.write(f"\nКритерій: {crit_name}\n")
                    ranking = sorted(crit_experts.target_weights.items(), key=lambda
x: x[1], reverse=True)
                    for i, (alt, weight) in enumerate(ranking):
                        file.write(f"{i+1}. {alt}: {weight}\n")
            print(f"Результати збережено у файлі: {filepath}")
        except Exception as e:
            print(f"Помилка при збереженні результатів: {e}")


    def go_to_menu():
        show_next_page()
    #     global data
    #     data = None

    #     create_button_create_template()
        button = tk.Button(root, text="Створити шаблон",
command=create_template_page)
        button.pack()
        global file_label
        file_label = tk.Label(root, text="Файл не вибрано")
        file_label.pack()
        global file_button
        file_button = tk.Button(root, text="Вибрати файл", command=open_file)
        file_button.pack()

        # file_button = tk.Button(root, text="Створити шаблон",
command=show_criteria_page)
        # file_button.pack()

        global calculate_button
        calculate_button = tk.Button(root, text="Розрахувати",
command=calculate)
        calculate_button.pack()
```

```
    global result_label
    result_label = tk.Label(root, text="")
    result_label.pack()



root = tk.Tk()
root.title("Система вибору контрагента")
root.geometry("1260x900")
root.eval("tk::PlaceWindow . center")

data = None



# create_button_create_template()

button = tk.Button(root, text="Створити шаблон", command=create_template_page)
button.pack()
# file_button = tk.Button(root, text="Створити шаблон", command=show_criteria_page)
# file_button.pack()


file_label = tk.Label(root, text="Файл не вибрано")
file_label.pack()
file_button = tk.Button(root, text="Вибрати файл", command=open_file)
file_button.pack()

calculate_button = tk.Button(root, text="Розрахувати", command=calculate)
calculate_button.pack()

result_label = tk.Label(root, text="")
result_label.pack()


root.mainloop()
```

Додаток Б. Вміст JSON-файлу

```
{
```

```
            "parameters": {
              "experts": 4,
              "criteria": 5,
              "alternatives": 5
            },
"criteria": ["Ціновий сегмент", "Логістика", "Якість товару", "РізноманіттяАсортименту", "Обслуговування"],
"alternatives": [
"ТОВ \"Меганом\"",
"Schneider Electric UA",
"ПП \"Сігма Кабель\"",
"ТОВ \"Е.МІР\"",
"ТОВ ДП \"Поло електрообладнання\""],
"expert_matrices": [
{"expert":1,"criteria_matrix":
[[1,0.14285714285714285,0.2,0.5,0.3333333333333333],[7,1,0.2,0.16666666666666666,0.125],[5,5,1,0.2,0.125],[2,6,5,1,0.16666666666666666],[3,8,8,6,1]],
"alternatives_matrices": [
[[1,0.14285714285714285,0.16666666666666666,0.1111111111111111,0.25],
[7,1,0.16666666666666666,0.3333333333333333,0.25],
[6,6,1,0.1111111111111111,0.3333333333333333],[9,3,9,1,0.125],[4,4,3,8,1]],
[[1,0.14285714285714285,0.1111111111111111,0.2,0.1111111111111111],[7,1,0.16666666666666666,0.3333333333333333,0.5],[9,6,1,0.1111111111111111,0.14285714285714285],[5,3,9,1,0.3333333333333333],[9,2,7,3,1]],
[[1,0.1111111111111111,0.1111111111111111,0.5,0.1111111111111111],[9,1,0.16666666666666666,0.14285714285714285,0.16666666666666666],[9,6,1,0.25,0.25],[2,7,4,1,0.125],[9,6,4,8,1]],
[[1,0.5,0.2,0.3333333333333333,0.25],[2,1,0.14285714285714285,0.125,0.166666
```

66666666666],[5,7,1,0.125,0.14285714285714285],[3,8,8,1,0.16666666666666666],[4,6,7,6,1]],
[[1,0.5,0.2,1.0,0.16666666666666666],[2,1,0.16666666666666666,0.5,0.3333333333333333],[5,6,1,0.5,0.5],[1,2,2,1,0.3333333333333333],[6,3,2,3,1]]]},
{"expert": 2,
"criteria_matrix":
[[1,0.25,0.5,0.5,0.3333333333333333],[4,1,0.5,0.2,0.2],[2,2,1,0.5,0.3333333333333333],[2,5,2,1,0.16666666666666666],[3,5,3,6,1]],
"alternatives_matrices": [
[[1,0.2,0.16666666666666666,0.5,0.3333333333333333],[5,1,0.5,0.3333333333333333,1.0],[6,2,1,0.5,0.2],[2,3,2,1,0.2],[3,1,5,5,1]],
[[1,0.25,0.5,0.2,0.25],[4,1,0.3333333333333333,1.0,1.0],[2,3,1,0.3333333333333333,0.5],[5,1,3,1,0.3333333333333333],[4,1,2,3,1]],
[[1,0.16666666666666666,0.5,1.0,0.2],[6,1,0.2,1.0,0.25],[2,5,1,0.2,0.5],[1,1,5,1,0.5],[5,4,2,2,1]],
[[1,0.3333333333333333,0.16666666666666666,0.5,1.0],[3,1,0.5,1.0,0.2],[6,2,1,0.5,0.5],[2,1,2,1,0.16666666666666666],[1,5,2,6,1]],
[[1,0.5,0.2,0.3333333333333333,1.0],[2,1,0.5,0.25,0.3333333333333333],[5,2,1,1.0,0.3333333333333333],[3,4,1,1,1.0],[1,3,3,1,1]]]},
{"expert": 3,
"criteria_matrix":
[[1,0.5,0.2,0.3333333333333333,1.0],[2,1,0.5,0.25,0.3333333333333333],[5,2,1,1.0,0.3333333333333333],[3,4,1,1,1.0],[1,3,3,1,1]],
"alternatives_matrices": [
[[1,0.5,0.2,0.3333333333333333,1.0],[2,1,0.5,0.25,0.3333333333333333],[5,2,1,1.0,0.3333333333333333],[3,4,1,1,1.0],[1,3,3,1,1]],
[[1,0.5,0.2,0.3333333333333333,1.0],[2,1,0.5,0.25,0.3333333333333333],[5,2,1,1.0,0.3333333333333333],[3,4,1,1,1.0],[1,3,3,1,1]],

[[1,0.3333333333333333,1.0,0.2,0.25],[3,1,0.16666666666666666,0.5,0.5],[1,6,1,0.5,0.3333333333333333],[5,2,2,1,0.2],[4,2,3,5,1]],
[[1,0.3333333333333333,1.0,0.2,0.25],[3,1,0.16666666666666666,0.5,0.5],[1,6,1,0.5,0.3333333333333333],[5,2,2,1,0.2],[4,2,3,5,1]],
[[1,0.2,0.2,0.5,1.0],[5,1,0.2,0.5,0.5],[5,5,1,0.3333333333333333,1.0],[2,2,3,1,1.0],[1,2,1,1,1]]]},
{"expert": 4,
"criteria_matrix":
[[1,0.2,0.2,0.5,1.0],[5,1,0.2,0.5,0.5],[5,5,1,0.3333333333333333,1.0],[2,2,3,1,1.0],[1,2,1,1,1]],
"alternatives_matrices": [
[[1,0.25,0.25,0.16666666666666666,0.16666666666666666],[4,1,0.25,0.16666666666666666,1.0],[4,4,1,0.25,0.16666666666666666],[6,6,4,1,0.25],[6,1,6,4,1]],[[1,0.16666666666666666,0.5,0.5,0.5],[6,1,1.0,0.5,0.5],[2,1,1,1.0,0.25],[2,2,1,1,0.5],[2,2,4,2,1]],[[1,0.2,0.3333333333333333,0.3333333333333333,0.5],[5,1,0.25,0.16666666666666666,0.25],[3,4,1,0.3333333333333333,0.16666666666666666],[3,6,3,1,0.2],[2,4,6,5,1]],[[1,0.25,0.3333333333333333,0.2,0.2],[4,1,0.25,0.2,1.0],[3,4,1,0.25,0.2],[5,5,4,1,1.0],[5,1,5,1,1]],
[[1,1.0,1.0,1.0,1.0],[1,1,0.16666666666666666,0.16666666666666666,0.16666666666666666],[1,6,1,0.5,0.5],[1,6,2,1,0.2],[1,6,2,5,1]]]}]}

Додаток В. Шаблон для вхідних даних для JSON-файлу

```
{
  "parameters": {
    "experts": КІЛЬКІСТЬ_ЕКСПЕРТІВ,
    "criteria": КРИТЕРІЇ,
    "alternatives": АЛЬТЕРНАТИВИ
  },
  "criteria": [СПИСОК_КРИТЕРІЇВ],
```

```
 "alternatives": [СПИСОК_АЛЬТЕРНАТИВ],
 "expert_matrices": [
  {
   "expert": НОМЕР_ЕКСПЕРТА,
   "criteria_matrix": [[МАТРИЦЯ_ПОРІВНЯННЯ_КРИТЕРІЇВ]],
   "alternatives_matrices": [
    [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],    # ДЛЯ ПЕРШОГО КРИТЕРІЮ
    [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],    # ДЛЯ ДРУГОГО КРИТЕРІЮ
     ...
   ]
  },
  {
   "expert": НОМЕР_ЕКСПЕРТА,
   "criteria_matrix": [[МАТРИЦЯ_ПОРІВНЯННЯ_КРИТЕРІЇВ]],
   "alternatives_matrices": [
    [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],    # ДЛЯ ПЕРШОГО КРИТЕРІЮ
    [[МАТРИЦЯ_ПОРІВНЯННЯ_АЛЬТЕРНАТИВ]],    # ДЛЯ ДРУГОГО КРИТЕРІЮ
     ...
   ]
  },
  ...
 ]}
```

Додаток Г. Вміст результуючого TXT-файлу

Найкраща альтернатива: ТОВ ДП "Поло електрообладнання"

Ваги альтернатив:

ТОВ "Меганом": 0.08103004000000001

Schneider Electric UA: 0.10744589

ПП "Сігма Кабель": 0.19615028

ТОВ "Е.МІР": 0.24297786999999998

ТОВ ДП "Поло електрообладнання": 0.37242851