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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА НА ТЕМУ

«Програмний модуль підтримки процедури преференційного голосування»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: **бакалавр**

Виконав: студент 4 курсу, групи КН- 42

_____Григор'єв А.Г._____

(прізвище та ініціали)

Керівник: Гнатієнко Г.М.

(наук. ступінь, звання, прізвище та ініціали))

Випускна кваліфікаційна робота бакалавра допущена до захисту

рішенням кафедри *інтелектуальних технологій*

Протокол № від р.

зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Зав. кафедри інтелектуальних технологій

к.т.н., доц. Іларіонов О.Є.

(звання, прізвище та ініціали)

_____

(підпис)

« ____ » _____ 2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Григор'єву Арсентію Георгійовичу

1. Тема роботи: «Програмний модуль підтримки процедури преференційного голосування» затверджена наказом по університету від «11» листопада 2022 р. №4
2. Термін виконання проєкту (роботи): 31 травня 2023 року
3. Вихідні дані до роботи: розробити програмний модуль підтримки голосування на основі алгебраїчного методу
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
   1) Аналіз предметної області преференційного голосування. Постановка задачі

2) Проєктування програмного модулю підтримки преференційного голосування
3) Програмна реалізація програмного модулю підтримки преференційного голосування. Тестові приклади

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій):

   Мета, предмет та об'єкт кваліфікаційної роботи (1 слайд), Проблема голосування та вирішення (1 слайд). Актуальність кваліфікаційної роботи (1 слайд). Основний алгоритм кваліфікаційної роботи (1 слайд). Стек обраних технологій програмного модулю (1 слайд). Архітектура програмного модулю (5 слайдів). Структура бази даних (2 слайди). Результати роботи програми (1 слайд). Висновки (1 слайд).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Гнатієнко Г.М.		
2	Гнатієнко Г.М.		
3	Гнатієнко Г.М.		

7. Дата видачі завдання 15 лютого 2023 року

Керівник випускної кваліфікаційної роботи_____ / Г. М. Гнатієнко /

(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ / А. Г. Григор'єв/

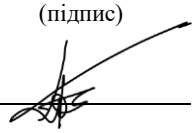
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Обговорення постановки завдання та змісту пояснювальної записки	25.01.2023 -22.02.2023	
2	Вибір та формування теми	23.02.2023 - 27.02.2023	
3	Аналіз предметної області	28.02.2023 – 11.03.2023	
4	Вибір методів рішення задачі	12.03.2023 –20.04.2023	
5	Створення програмного модулю	21.04.2023 -10.05.2023	
6	Оформлення пояснювальної записки	11.05.2023 – 29.05.2023	

Керівник випускної кваліфікаційної роботи _____ / Г. М. Гнатієнко /

(підпис) (ініціали та прізвище)

Студент  / А. Г. Григор’єв /

(підпис) (ініціали та прізвище

Анотація

Григор’єв Арсентій Георгійович виконав випускну кваліфікаційну роботу на тему «Програмний модуль підтримки процедури преференційного голосування» за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних методів голосування в колективі, розглянуто алгебраїчний метод обробки експертної інформації для підтримки голосування шляхом вибору упорядкованої підмножини кандидатів, розроблено серверне та клієнтське програмне забезпечення, що дозволяє проводити голосування та агрегацію даних з урахуванням експертних оцінок.

Ключові слова: підтримка голосування, упорядкована множина, алгебраїчний метод, експертна інформація, програмний модуль, комп'ютерні науки.

Summary

The degree project: «A software module for supporting preferential voting procedure» has been completed by **Hryhoriev Arsentii Heorhiyovych**, specialty 122 – «Computer Sciences».

In this graduation thesis, modern methods of voting in a collective are analyzed, the algebraic method of expert information processing is considered to support the preliminary voting process, and server-side and client-side software is developed that enables voting and data aggregation taking into account expert evaluations.

Keywords: supporting voting, subset of candidates, algebraic method, expert information, software module, computer sciences.

ЗМІСТ

ВСТУП

Сьогодні в умовах розвитку сучасного суспільства стає важливим забезпечення ефективної комунікації та ухвалення рішень в різноманітних галузях життєдіяльності людей. Одним з найважливіших інструментів у процесі прийняття рішень є голосування в колективі, що дозволяє агрегувати різні точки зору та досягати консенсусу з найбільшим задоволенням більшості учасників. Враховуючи значний вплив рішень, прийнятих на основі голосування, на успіх діяльності колективів, необхідно розробити ефективні та надійні інструменти підтримки такого процесу.

Метою даної випускної кваліфікаційної роботи є розробка програмного модуля підтримки процедури преференційного голосування на основі алгебраїчного методу обробки експертної інформації, що дозволить забезпечити більш об'єктивний та ефективний процес ухвалення рішень у малих колективах.

Актуальність теми обумовлена постійним розвитком та модернізацією інформаційних технологій, що сприяють оптимізації процесів ухвалення рішень та підвищенню ефективності роботи колективів. Застосування алгебраїчного методу обробки експертної інформації дозволить враховувати багаторівневу структуру оцінок експертів та досягти репрезентативності результатів голосування. Основні проектні рішення базуються на використанні сучасних програмних технологій та архітектурних рішень для забезпечення масштабованості, надійності та зручності використання розробленого програмного модуля.

Можливі галузі застосування результатів роботи охоплюють широкий спектр сфер, в яких необхідно забезпечити ефективне ухвалення рішень, таких як управління підприємствами, освітні заклади, державні установи, громадські організації та інші. Зокрема, програмний модуль може бути використаний для підтримки голосування в органах влади, що дозволить підвищити прозорість та

об'єктивність прийнятих рішень, а також сприятиме залученню громадськості до процесу прийняття рішень.

Об'єктом дослідження в даній випускній кваліфікаційній роботі є процес голосування в колективі як ключовий інструмент прийняття рішень.

Предметом дослідження є програмний модуль підтримки преференційного голосування в колективі, який реалізований на основі алгебраїчного методу обробки експертної інформації. Специфічними аспектами, що підлягають дослідженню в цьому контексті, є розробка алгоритмів для обробки даних голосування, проектування архітектури програмного модуля, а також його програмна реалізація та тестування.

Аналізуючи відомі вирішення проблеми та порівнюючи їх з пропонованим підходом, можна стверджувати, що розробка програмного модуля підтримки голосування в колективі на основі алгебраїчного методу обробки експертної інформації є актуальною та доцільною для розвитку відповідної галузі науки та виробництва. Впровадження такого модуля сприятиме підвищенню ефективності процесів прийняття рішень в колективах, а також покращенню якості управління та комунікації на різних рівнях влади та суспільства, особливо на користь України.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРЕФЕРЕНЦІЙНОГО ГОЛОСУВАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Аналітичний огляд

1.1.1 Сучасний стан досліджень в області прийняття рішень

Протягом останніх декількох десятиліть, науковці активно досліджують теорію прийняття рішень, що включає розробку методів, моделей та процесів, які допомагають приймати рішення в умовах невизначеності, ризику та багатоцільовості. Важливим аспектом цих досліджень є вивчення процесів прийняття рішень на різних рівнях: індивідуальному, груповому та організаційному.

Основні напрямки досліджень включають класичні методи, такі як принцип максимізації очікуваної корисності, а також більш сучасні підходи, що базуються на теорії ігор, байесівських методах та багатоцільовому оптимізації. Окремою галуззю є дослідження евристичних методів прийняття рішень, які спрощують складність задач і допомагають приймати рішення швидше, хоча іноді з втратою точності.

Науковці також активно досліджують роль психологічних та соціальних факторів у процесі прийняття рішень. Одним з ключових напрямків є вивчення когнітивних упереджень, які можуть негативно впливати на якість прийнятих рішень. Дослідження в цій області включають такі явища, як анкерування, надмірна впевненість та омана підтвердження.

В області колективного прийняття рішень важливим аспектом є розробка методів, що дозволяють забезпечити ефективне співробітництво та координацію між учасниками.

Це включає розробку систем підтримки прийняття рішень, які інтегрують експертні знання та думки різних учасників для досягнення консенсусу та вибору найкращого варіанту. Методи обробки експертної інформації, такі як алгебраїчний

метод, демпстер-шаферова теорія та аналітична ієрархія процесів, допомагають об'єднати різноманітні думки та оцінки у взаємосв'язаній структурі.

Технологічний розвиток вніс свій вклад у розвиток методів прийняття рішень. Штучний інтелект, машинне навчання та оптимізаційні алгоритми дозволяють створювати розумні системи підтримки прийняття рішень, які допомагають людям краще аналізувати ситуацію та вибирати оптимальні варіанти. Ці технології стали особливо актуальними для розробки програмних модулів, які допомагають у попередньому голосуванні в колективі, де координація та обробка інформації є ключовими факторами успіху.

Враховуючи ці сучасні дослідження та тенденції, ваша кваліфікаційна робота зосереджується на розробці програмного модуля підтримки попереднього голосування в колективі на основі алгебраїчного методу обробки експертних даних. Це дозволить вам інтегрувати сучасні знання з області прийняття рішень та створити ефективний інструмент для підтримки колективного вирішення проблем та досягнення консенсусу.

1.1.2 Проблеми та потреби в програмних модулях підтримки попереднього голосування

У сучасному світі прийняття рішень стає все більш складним процесом, який вимагає залучення різних експертів та учасників з різними компетенціями. У зв'язку з цим з'являється потреба в програмних модулях, які підтримують попереднє голосування та сприяють ефективному прийняттю рішень. Ось деякі проблеми та потреби, пов'язані з такими модулями:

1. Об'єднання різних думок та переконань: У різних галузях потрібно зібрати думки та переконання різних учасників процесу, щоб досягти найкращого варіанту. Програмні модулі, що підтримують попереднє голосування, допомагають враховувати різні точки зору та прийти до консенсусу.

2. Швидкість прийняття рішень: Важливою проблемою є швидкість прийняття рішень. За допомогою програмних модулів можна прискорити процес голосування та аналізу, а також забезпечити більш точне підрахунок голосів.

3. Ефективність комунікації: Для успішного прийняття рішень потрібна чітка та ефективна комунікація між учасниками. Програмні модулі можуть спростити обмін інформацією, допомагаючи учасникам зосередитися на важливих аспектах проблеми та забезпечити взаєморозуміння.

4. Забезпечення прозорості та об'єктивності: Програмні модулі для підтримки попереднього голосування можуть забезпечити прозорість та об'єктивність прийняття рішень. Це важливо для підтримки довіри між учасниками та створення об'єктивного процесу, який відображає різні точки зору та інтереси.

5. Конфіденційність та безпека: Охорона конфіденційності та безпеки експертних даних та голосів є важливим аспектом програмних модулів для підтримки попереднього голосування. Це включає захист від несанкціонованого доступу, забезпечення анонімності учасників та збереження інформації в безпечному середовищі.

У цілому, програмні модулі для підтримки попереднього голосування мають відповідати ряду потреб та викликів, пов'язаних зі спільним прийняттям рішень у різних галузях. Це включає забезпечення ефективної комунікації, прозорості, конфіденційності та адаптивності, що дозволяє учасникам досягати найкращих результатів у складних та динамічних умовах. Використання програмних модулів для підтримки попереднього голосування також сприяє широкому включенню різних учасників, забезпечуючи їх можливість внести свій вклад у процес прийняття рішень та відображення різноманітності думок та інтересів.

Однією з ключових проблем інших методів обробки експертних даних є те, що вони часто працюють з неточностями та неоднозначностями, які можуть призводити до помилок або неправильних оцінок. Деякі методи голосування, такі як більшість або кумулятивне голосування, можуть призводити до зміщення результатів, оскільки вони не враховують вплив різних рівнів експертизи учасників або не розглядають взаємозв'язки між альтернативами.

Програмний модуль, заснований на алгебраїчному методі обробки експертних даних, може допомогти вирішити ці проблеми. Алгебраїчний метод дозволяє об'єднати експертні думки, враховуючи їх рівень компетентності та взаємозв'язок факторів, що впливають на рішення. Такий підхід допомагає зменшити неточності та забезпечує більш об'єктивний аналіз різних альтернатив.

Крім того, програмний модуль на основі алгебраїчного методу може бути гнучким і адаптивним, що дозволяє йому швидко реагувати на зміни умов або вимог до рішення. Він може також використовуватися для моделювання різних сценаріїв та відслідковування змін впливу різних факторів на рішення, що сприяє більшій прозорості та розумінню процесу прийняття рішень.

Оскільки алгебраїчний метод обробки експертних даних забезпечує більш точні та обґрунтовані результати, організації можуть бути впевнені у виборі кращих альтернатив та використанні ресурсів найбільш ефективним чином. Це може призводити до більшої задоволеності клієнтів, збільшення прибутків та зниження ризиків, пов'язаних з прийняттям рішень.

З урахуванням вищезазначеного, розробка програмного модуля підтримки попереднього голосування, заснованого на алгебраїчному методі обробки експертних даних, є важливим та актуальним завданням. Впровадження такого модуля може допомогти організаціям покращити якість прийняття рішень, забезпечити ефективне використання ресурсів та досягнути більшої успішності в своїй діяльності.

1.1.3 Відомі методи та технології підтримки голосування

Підтримка голосування та прийняття рішень в організаціях та групах часто вимагає використання різних методів та технологій. Відомі методи та технології підтримки голосування можуть бути розглянуті з точки зору їх підходів до збору даних, обробки інформації та представлення результатів. В цьому пункті ми розглянемо деякі з них.

Відомі методи та технології підтримки голосування можна поділити на дві категорії: методи голосування та методи обробки експертної інформації.

1. Методи голосування
    - Голосування більшості: Простий метод, де варіант, що набирає найбільшу кількість голосів, перемагає.
    - Кондорсе: Визначає переможця на основі попарного порівняння кандидатів. Перемагає той, хто перемагає в попарних порівняннях найчастіше.
    - Борда: Кожен виборець присвоює кандидатам бали відповідно до їхнього рейтингу. Перемагає кандидат з найбільшою сумою балів.

2. Методи обробки експертної інформації

- Аналітичний ієрархічний процес (AHP): Багатокритеріальний метод, який вимагає попарного порівняння критеріїв та альтернатив за кожним критерієм.
- Електре: Метод, який вимагає встановлення порогів для критеріїв та порівняння альтернатив за кожним критерієм з урахуванням порогів.
- PROMETHEE: Метод, що базується на паретівському принципі, який дозволяє порівнювати альтернативи, враховуючи кілька критеріїв одночасно.

Алгебраїчний метод обробки експертних даних вважається одним з найточніших та найефективніших методів. Він використовує математичні та статистичні моделі для обробки даних та оцінювання альтернатив. Алгебраїчний метод може бути застосований в комбінації з іншими методами теорії прийняття рішень, такими як теорія ігор, теорія багатокритеріального вибору, теорія нечітких множин та інші. Алгебраїчний метод дозволяє розглядати нечіткість та неточність

експертних даних, що забезпечує більш точне та об'єктивне рішення на основі експертної інформації.

Деякі основні технології, що використовуються в теорії прийняття рішень та алгебраїчному методі, включають:

- Алгоритм Копленда: використовується для порівняння альтернатив на основі попарних порівнянь та визначення рейтингу кандидатів.
- Теорема Де Борда: стверджує, що в системі голосування, яка відповідає певним вимогам, не може існувати ідеального методу.
- Алгоритм Шульце: метод порівняння альтернатив, який забезпечує консенсус між учасниками голосування.
- Теорія нечітких множин: метод аналізу нечіткості та неточності, що дозволяє краще моделювати реальний світ та працювати з експертними даними.

Алгебраїчний метод є особливо важливим для програмних модулів підтримки попереднього голосування, оскільки він може адекватно обробляти нечіткість та неточність експертних даних. Такі модулі можуть забезпечити кращу підтримку прийняття рішень, надаючи засновані на алгебраїчному методі результати, які відображають реальний стан справ та враховують різні точки зору експертів.

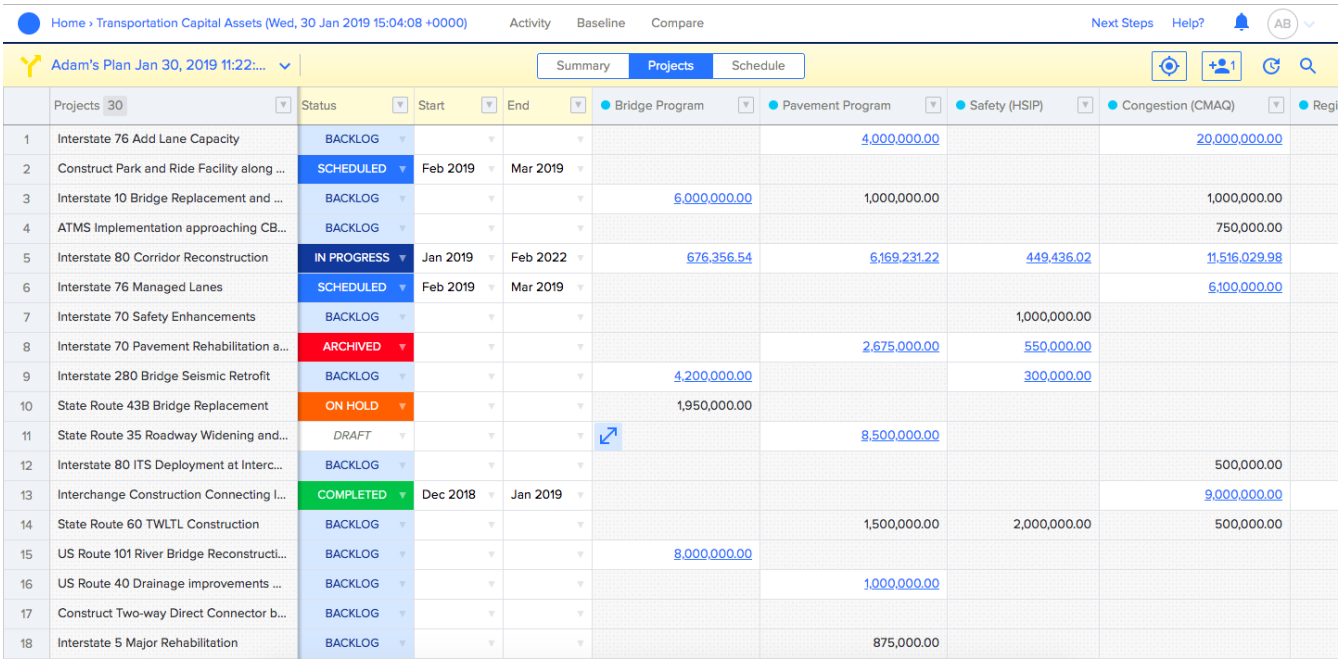
Враховуючи різноманітність методів та технологій, що існують для підтримки голосування та прийняття рішень, важливо визначити, який підхід найкраще підходить для конкретної ситуації або проблеми. Основні фактори, які слід враховувати, включають розмір групи або команди, структуру й взаємодію між учасниками, характер проблеми та критерії прийняття рішень.

Важливо також враховувати аспект людської сторони прийняття рішень, такі як емоції, суб'єктивні переконання та когнітивні упередження, які можуть впливати на об'єктивність та якість рішень. Використання технологій, що сприяють анонімності та незалежності учасників, може допомогти зменшити вплив таких факторів на процес голосування та прийняття рішень.

1.1.4 Огляд аналогів

Проведено аналіз предметної області створюваного проєкту, для цього його було порівняно з іншими програмними продуктами.

1. Decision Lens: це платформа для стратегічного прийняття рішень, яка використовує експертні знання та думки для аналізу альтернатив та пріоритезації проектів. Вона включає різні інструменти, такі як портфельний аналіз, оптимізація ресурсів та планування. Інтерфейс наведений на рис.1.



	Projects 30	Status	Start	End	Bridge Program	Pavement Program	Safety (HSIP)	Congestion (CMAQ)	Regi
1	Interstate 76 Add Lane Capacity	BACKLOG				4,000,000.00		20,000,000.00	
2	Construct Park and Ride Facility along ...	SCHEDULED	Feb 2019	Mar 2019					
3	Interstate 10 Bridge Replacement and ...	BACKLOG			6,000,000.00	1,000,000.00		1,000,000.00	
4	ATMS Implementation approaching CB...	BACKLOG						750,000.00	
5	Interstate 80 Corridor Reconstruction	IN PROGRESS	Jan 2019	Feb 2022	676,356.54	6,169,231.22	449,436.02	11,516,029.98	
6	Interstate 76 Managed Lanes	SCHEDULED	Feb 2019	Mar 2019				6,100,000.00	
7	Interstate 70 Safety Enhancements	BACKLOG					1,000,000.00		
8	Interstate 70 Pavement Rehabilitation a...	ARCHIVED				2,675,000.00	550,000.00		
9	Interstate 280 Bridge Seismic Retrofit	BACKLOG			4,200,000.00		300,000.00		
10	State Route 43B Bridge Replacement	ON HOLD			1,950,000.00				
11	State Route 35 Roadway Widening and...	DRAFT				8,500,000.00			
12	Interstate 80 ITS Deployment at Interc...	BACKLOG						500,000.00	
13	Interchange Construction Connecting I...	COMPLETED	Dec 2018	Jan 2019				9,000,000.00	
14	State Route 60 TWLTL Construction	BACKLOG				1,500,000.00	2,000,000.00	500,000.00	
15	US Route 101 River Bridge Reconstructi...	BACKLOG			8,000,000.00				
16	US Route 40 Drainage improvements ...	BACKLOG				1,000,000.00			
17	Construct Two-way Direct Connector b...	BACKLOG							
18	Interstate 5 Major Rehabilitation	BACKLOG				875,000.00			

Рисунок 1.1 - Інтерфейс Decision Lens

2. 1000Minds: це програмна платформа, яка використовує метод парних порівнянь для підтримки прийняття рішень, включаючи інтеграцію експертних оцінок та думок. Вона підходить для вирішення проблем у таких галузях, як охорона здоров'я, екологія, освіта та бізнес. Інтерфейс наведений на рис. 2.

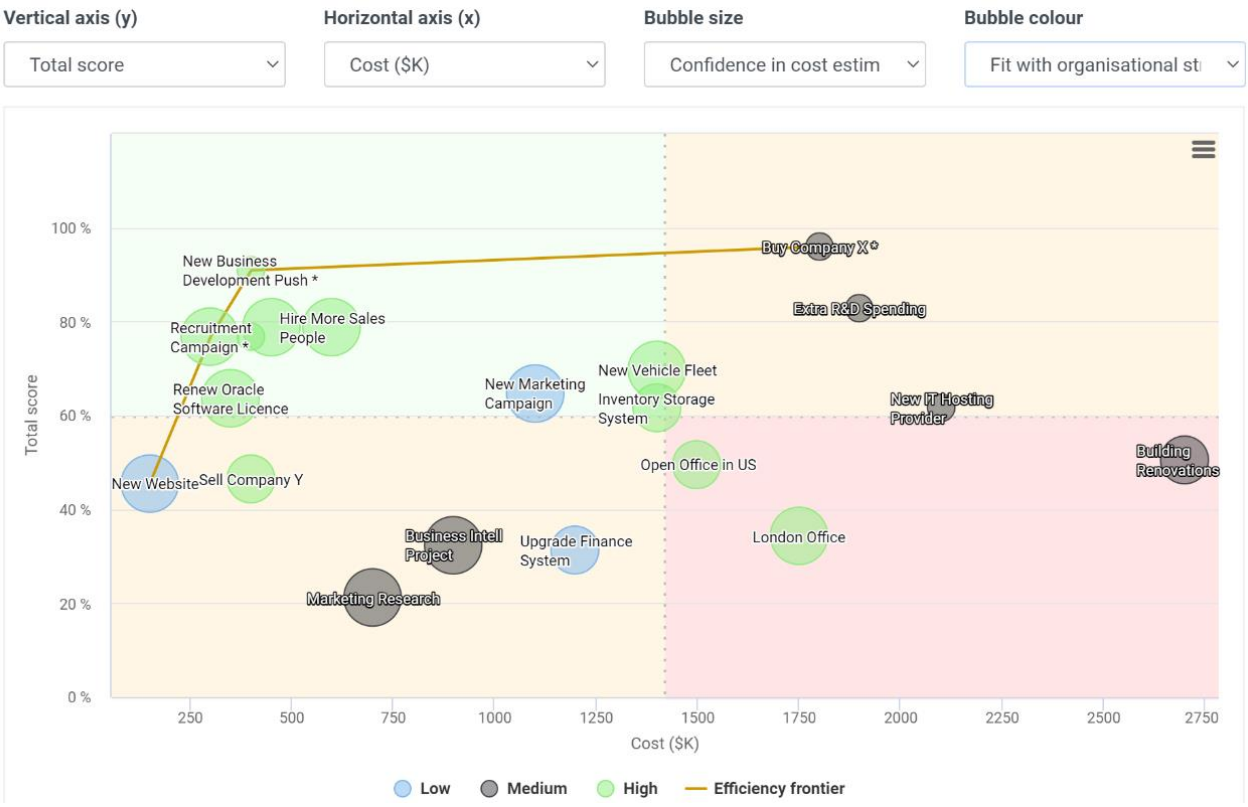


Рисунок 1.2 - Інтерфейс 100Minds

3. Expert Choice: це програмне рішення, засноване на методі аналізу ієрархій (AHP), яке допомагає у процесі прийняття рішень шляхом інтеграції експертних думок та оцінок. Воно дозволяє групам ефективно порівнювати та визначати пріоритети альтернативних варіантів. Інтерфейс наведений на рис. 3.

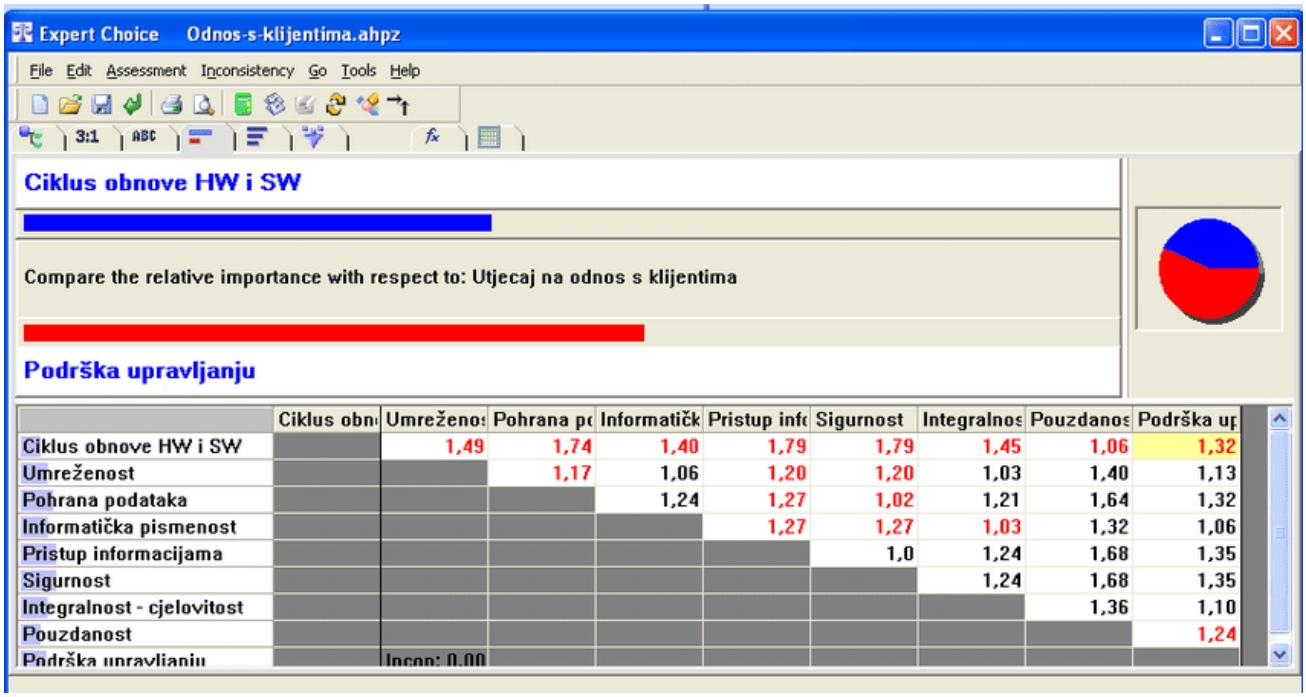


Рисунок 1.3 - Інтерфейс програми 1000Minds

Таблиця 1.1 – порівняння програмних застосунків

	Expert Choice	Decision Lens	1000Minds
Зручний інтерфейс	-	-	+
Парний порівняльний аналіз	+	+	+
Аналіз чутливості	-	+	-
Вбудована система оцінки	+	+	+
Інтеграція з експертною інформацією	+	+	+
Колаборація між користувачами	-	+	+
Підтримка налаштувань критеріїв	+	-	+

Звітність та візуалізація даних	-	+	-

1.1.5 Висновки аналітичного огляду

В аналітичному огляді було досліджено сучасний стан рішень у сфері підтримки прийняття рішень та програмних модулів для попереднього голосування. З урахуванням аналізу існуючих програмних продуктів, таких як Expert Choice, Decision Lens та 1000Minds, було виявлено, що кожен з них має свої переваги та недоліки.

Хоча деякі з цих програмних рішень надають можливість інтеграції експертної інформації та колаборації між учасниками, їх функціональність може бути обмежена у відношенні до алгебраїчного методу обробки експертних даних, що розглядається в даній роботі. Крім того, інтерфейси користувачів, налаштування критеріїв та візуалізація даних можуть відрізнятися, що може вплинути на зручність та продуктивність роботи з цими програмами.

На підставі проведеного огляду, можна зробити висновок про необхідність проектування оригінального програмного модуля підтримки прийняття рішень. Такий модуль повинен забезпечувати інтеграцію з алгебраїчним методом обробки експертних даних, що допоможе враховувати думки різних учасників для вибору найкращого варіанту. Крім того, оригінальний модуль має бути зручним для користувачів, мати гнучкі налаштування критеріїв та ефективні засоби візуалізації даних, що сприятиме успішному застосуванню рішення в різних областях.

1.2 Постановка задачі

1.2.1 Формулювання мети та задачі дипломної роботи

Мета: Створення програмного модулю на основі алгебраїчного методу обробки експертних даних, який допомагає враховувати думки різних учасників у процесі вибору найкращого варіанту.

Постановка задачі: Розглядається задача визначення ранжування об’єктів (побудови лінійного порядку) на основі заданих експертами множинних порівнянь об’єктів фіксованої довжини та розробка програмного модулю, що використовує алгебраїчний метод обробки експертних даних для визначення множини кандитатів у голосуванні.

Модуль, що розробляється, повинен виконувати ряд задач:

- Збір та обробка експертних даних;
- Інтеграція алгебраїчного методу для вагового ранжування альтернатив;
- Забезпечення зручного користувацького інтерфейсу для введення даних та відображення результатів;
- Забезпечення сумісності з іншими програмними рішеннями для підтримки прийняття рішень.

Для реалізації поставленої мети були виділені наступні задачі та дослідження:

- Вивчення існуючих програмних рішень та їх аналіз для виявлення переваг та недоліків;
- Розробка математичних моделей та методів для алгебраїчного методу обробки експертних даних;
- Проектування архітектури програмного модуля та розробка алгоритмів для реалізації поставлених задач;
- Відладка та тестування програмного модуля;
- Оформлення супроводжуючої документації та презентаційних матеріалів.

1.2.2 Системні вимоги до програмного модуля підтримки попереднього голосування

Системні вимоги до програмного модуля визначають параметри та обмеження, які необхідно враховувати при розробці та реалізації модуля. Вони включають:

- Сумісність з різними операційними системами: програмний модуль повинен бути сумісний з найпоширенішими операційними системами, такими як Windows, macOS та Linux, для забезпечення широкого доступу користувачів.
- Підтримка різних форматів даних: модуль повинен мати змогу імпортувати та експортувати дані у різних форматах, таких як CSV, XLSX, JSON тощо, для полегшення обміну даними з іншими програмними продуктами.
- Модульна архітектура: програмний модуль повинен мати модульну структуру, що дозволить легко додавати або модифікувати функціонал, забезпечуючи гнучкість та адаптивність до змінюваних вимог користувачів.
- Безпека даних: програмний модуль повинен мати вбудовані механізми захисту даних, щоб забезпечити конфіденційність та цілісність інформації, яка обробляється.
- Масштабованість: програмний модуль повинен бути проектований з урахуванням можливості масштабування, щоб підтримувати велику кількість користувачів та обробляти великі обсяги даних.
- Локалізація: програмний модуль повинен підтримувати локалізацію, що дозволяє адаптувати його до різних мов та культурних особливостей користувачів.
- Інтерфейс користувача: програмний модуль повинен мати простий, інтуїтивно зрозумілий та зручний для користувача інтерфейс, що сприяє швидкому навчанню та використанню програми.
- Технічна підтримка: програмний модуль повинен супроводжуватися відповідною технічною підтримкою, що допоможе користувачам вирішувати можливі проблеми під час використання та налаштування програми.

Врахування цих системних вимог допоможе забезпечити створення ефективного, надійного та корисного програмного модуля для підтримки попереднього голосування, що задовольнятиме потреби користувачів та сприятиме якісному прийняттю рішень.

1.2.3 Функціональні вимоги

Функціональні вимоги до програмного модуля:

- Реєстрація та авторизація користувачів: модуль повинен надавати можливість реєстрації нових користувачів і авторизації існуючих для доступу до функціоналу.
- Створення голосування: модуль повинен дозволяти користувачам створювати нові голосування з вказівкою варіантів вибору та параметрів голосування.
- Запрошення експертів: модуль повинен надавати можливість запрошувати експертів для участі в голосуванні за допомогою електронної пошти або інших засобів комунікації.
- Проведення голосування: модуль повинен дозволяти експертам голосувати за вибраний варіант та надавати можливість зміни вибору протягом встановленого часу голосування.
- Обробка результатів голосування: модуль повинен забезпечувати обробку результатів голосування за допомогою алгебраїчного методу та інших доступних методів.
- Візуалізація результатів: модуль повинен відображати результати голосування у вигляді графіків, діаграм та таблиць для зручного аналізу користувачами.
- Експорт результатів: модуль повинен надавати можливість експорту результатів голосування у різних форматах, таких як CSV, Excel, PDF тощо.

1.2.4 Нефункціональні вимоги

Нефункціональні вимоги до програмного модуля:

- Зручність використання: програмний модуль повинен мати інтуїтивно зрозумілий і зручний інтерфейс користувача.

- Масштабованість: програмний модуль повинен мати можливість працювати з великою кількістю користувачів та голосувань одночасно.
- Безпека: програмний модуль повинен забезпечувати безпеку даних користувачів та результатів голосування, використовуючи сучасні методи шифрування та аутентифікації.
- Сумісність з різними платформами: програмний модуль повинен підтримувати роботу на різних операційних системах та пристроях, таких як комп'ютери, смартфони та планшети.
- Продуктивність: програмний модуль повинен відрізнятися високою швидкістю обробки результатів голосування та відгуком інтерфейсу.
- Надійність: програмний модуль повинен працювати стабільно та без збоїв, а також мати систему резервного копіювання для попередження втрати даних.
- Локалізація: програмний модуль повинен надавати підтримку багатьох мов, що дозволить користувачам з різних країн працювати з ним комфортно.
- Технічна підтримка: програмний модуль повинен мати систему технічної підтримки для вирішення можливих проблем користувачів та отримання зворотного зв'язку.
- Документація: програмний модуль повинен мати чітку та детальну документацію для користувачів, що допоможе їм швидко освоїти функціонал і вирішити можливі питання.
- Інтеграція: програмний модуль повинен мати можливість інтеграції з іншими системами та сервісами, що дозволить розширити його функціонал та забезпечити більш гнучке використання.

1.2.5 Вимоги до окремих видів забезпечення та/або їх компонентів

Вимоги до апаратного забезпечення:

- сумісність з різними пристроями (комп'ютери, планшети, смартфони);
- наявність достатньої обчислювальної потужності для швидкої обробки даних.

Вимоги до системного забезпечення:

- сумісність з різними операційними системами (Windows, macOS, Linux, Android, iOS);
- можливість оновлення та патчів для підтримки безпеки та стабільності програмного модуля.

Вимоги до мережевого забезпечення:

- надійна та безпечна передача даних між користувачами та серверами;
- забезпечення доступності програмного модуля 24/7.

Вимоги до програмного забезпечення:

- гнучка архітектура для можливості розширення функціоналу та інтеграції з іншими системами;
- модульність для легкої модифікації окремих компонентів;
- відкритий API для розробки додаткових рішень на основі програмного модуля.

Вимоги до інтерфейсу користувача:

- інтуїтивно зрозумілий та зручний інтерфейс для користувачів з різним рівнем досвіду;
- адаптивний дизайн, що забезпечує зручне відображення на різних пристроях;
- підтримка локалізації для використання додатка на різних мовах.

1.2.6 Вхідні дані

Для роботи програмного модуля підтримки попереднього голосування необхідна наступна вхідна інформація:

- дані про учасників голосування (ідентифікатори, ролі, відомості про експертів);
- список варіантів рішень для голосування;
- критерії оцінки варіантів рішень та їх ваги;
- експертні оцінки варіантів рішень, зібрані від учасників голосування;

- параметри алгебраїчного методу обробки експертної інформації (якщо потрібно).

1.2.7 Вихідні дані

Після обробки вхідної інформації програмний модуль надає наступну вихідну інформацію:

- зведена оцінка варіантів рішень, з урахуванням думок усіх учасників голосування;
- рейтинг варіантів рішень, відсортований за зведеною оцінкою;
- аналіз відхилень між оцінками різних експертів, що допомагає виявити можливі проблеми або розбіжності у поглядах;
- рекомендації щодо вибору оптимального варіанту рішення на основі зведеної оцінки та рейтингу;
- можливість відстеження змін у результатах голосування в разі оновлення вхідної інформації (наприклад, нові оцінки або критерії).

1.3 Висновки до першого розділу

В результаті проведеної аналітичної роботи в рамках першого розділу, було досліджено проблему підтримки прийняття рішень на основі алгебраїчного методу обробки експертних даних. Було обґрунтовано актуальність цієї проблеми, оскільки вона стосується рішень, які забезпечують залучення різних експертних думок і сприяють прийняттю кращих рішень.

Було проведено аналіз програмних аналогів, таких як Expert Choice, Decision Lens та 1000Minds, та їх функціональності. З огляду на переваги та недоліки цих програмних продуктів, було визначено, що необхідно розробити оригінальний програмний модуль підтримки прийняття рішень, який найбільш ефективно задовольнятиме потреби користувачів.

В ході роботи було сформульовано чітке формулювання задачі випускної кваліфікаційної роботи, що включає розробку програмного модуля підтримки попереднього голосування на основі алгебраїчного методу обробки експертних даних. Були визначені системні вимоги, функціональні та нефункціональні вимоги до створюваного програмного модуля, а також вимоги до окремих видів забезпечення та їх компонентів.

Було сформовано вхідну і вихідну інформацію для програмного модуля. Все це дає солідну основу для розробки програмного модуля, який має потенціал стати надійним інструментом підтримки прийняття рішень, здатним враховувати думки різних експертів і вибирати найкращі варіанти в різних ситуаціях.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ ПІДТРИМКИ ПРЕФЕРЕНЦІЙНОГО ГОЛОСУВАННЯ

2.0 Вибір та обґрунтування мови програмування, технологій та бібліотек

Програмний модуль буде побудований на наступному стеку:

- Клієнтська частина: React Typescript, Framer Motion та Material UI.
- Серверна частина: Express, MongoDB та GraphQL.

React і TypeScript: React є популярною бібліотекою JavaScript для побудови інтерфейсів користувача, зокрема односторінкових веб-додатків. TypeScript - це надмножина JavaScript, яка додає строгу типізацію, що дозволяє розробникам виявляти помилки типів на етапі компіляції, покращуючи якість коду та забезпечуючи більшу стабільність додатку. Використання React і TypeScript спрощує розробку, робить код більш структурованим та підтримуваним.[15]

Material UI: це популярна бібліотека компонентів React. яка дозволяє розробникам швидко створювати привабливі та консистентні інтерфейси користувача, що покращує взаємодію користувачів з додатком.

Express: Express - це швидкий та гнучкий веб-фреймворк для Node.js, який полегшує розробку веб-додатків, надаючи набір корисних функцій для обробки HTTP-запитів та відповідей.

MongoDB: MongoDB - це нереляційна база даних, яка дозволяє зберігати дані у форматі JSON. Вона пропонує гнучкість у структурі даних, масштабованість та швидкість роботи.

GraphQL: це запитова мова та середовище виконання для API, яке дозволяє клієнтам точно вказувати, які дані їм потрібні, забезпечуючи більш ефективне використання ресурсів та зменшуючи кількість запитів до сервера. Використання GraphQL дозволить підтримувати надійні, гнучкі та ефективні API для вашого веб-додатку.

Framer Motion: це бібліотека анімації для React, яка допомагає легко створювати різні анімаційні ефекти та переходи між сторінками веб-додатку. Використання Framer Motion підвищить естетичність веб-додатку та забезпечить більш гладкий та приємний досвід користувача.

2.1 Функціональний аналіз

2.1.1 Дерево функцій

Для початку розробимо дерево функцій для відображення базових клієнтських функцій та адміністративних.

Вершина дерева функцій – модуль додатку, який включає в собі основні функції: клієнтські та адміністративні функції.

До клієнтських функцій входить авторизація користувача в системі та повний комплекс функцій для керування голосуванням, включаючи створення голосування, додавання об'єктів, налаштування параметрів голосування, ініціювання експертної оцінки, а також обробка результатів голосування.

Адміністративні функції дозволяють розширювати можливості додатку за рахунок додавання нових функцій, критеріїв та методів обробки експертних даних. Ці функції спрямовані на покращення та оптимізацію роботи додатку, а також на реалізацію специфічних потреб користувачів.

Таким чином, дерево функцій відображає основні напрямки роботи додатку та різноманітність функціональних можливостей, що надаються користувачам для проведення голосувань та обробки експертної інформації.

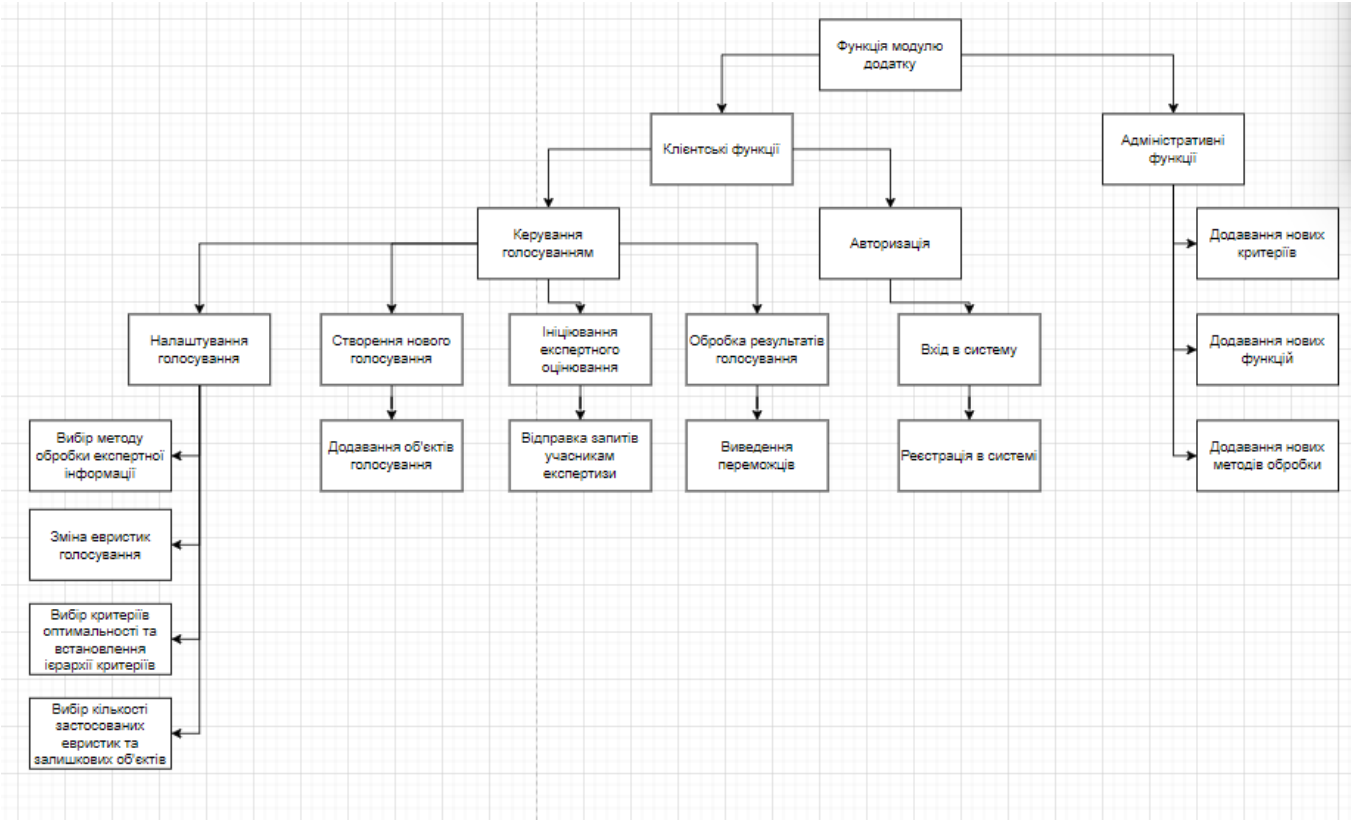


Рисунок 2.1 – Дерево функцій

2.1.2 Діаграма event-driven process chain

Опишемо функціонування програмного модуля для організації голосування з експертною оцінкою на основі діаграми EPC. Початок роботи з додатком полягає в тому, що користувач відкриває додаток в браузері на своєму телефоні чи комп'ютері. Користувач реєструється, створюючи новий акаунт або входячи в існуючий.

Після авторизації, користувач вводить дані для створення нового голосування, додає об'єкти голосування та вибирає метод обробки експертної інформації. Користувач також має змогу змінити евристики голосування, вибрати кількість застосованих евристик та залишкових об'єктів, а також вибрати критерії оптимальності та встановити ієрархію критеріїв.

Коли голосування налаштовано, користувач ініціює експертне оцінювання, розсилаючи запити учасникам експертизи. Після отримання результатів від усіх експертів, система обробляє результати голосування.

Адміністратор має можливість працювати з додатком, додавати нові функції, критерії та методи обробки експертних даних. Для цього адміністратор відкриває налаштування додатку, де може внести необхідні зміни.

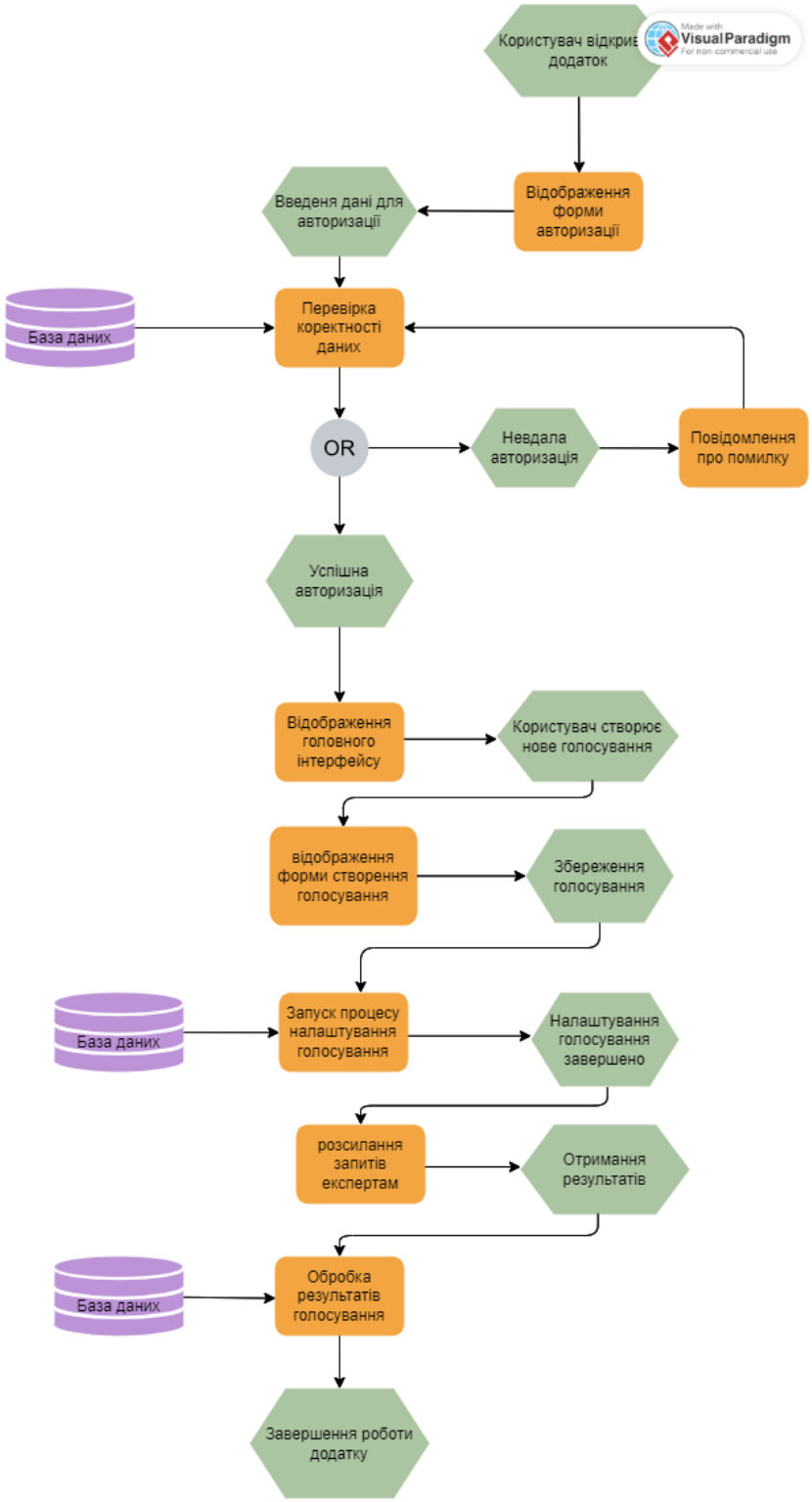


Рисунок 2.2 – діаграма EPC для клієнтських функцій

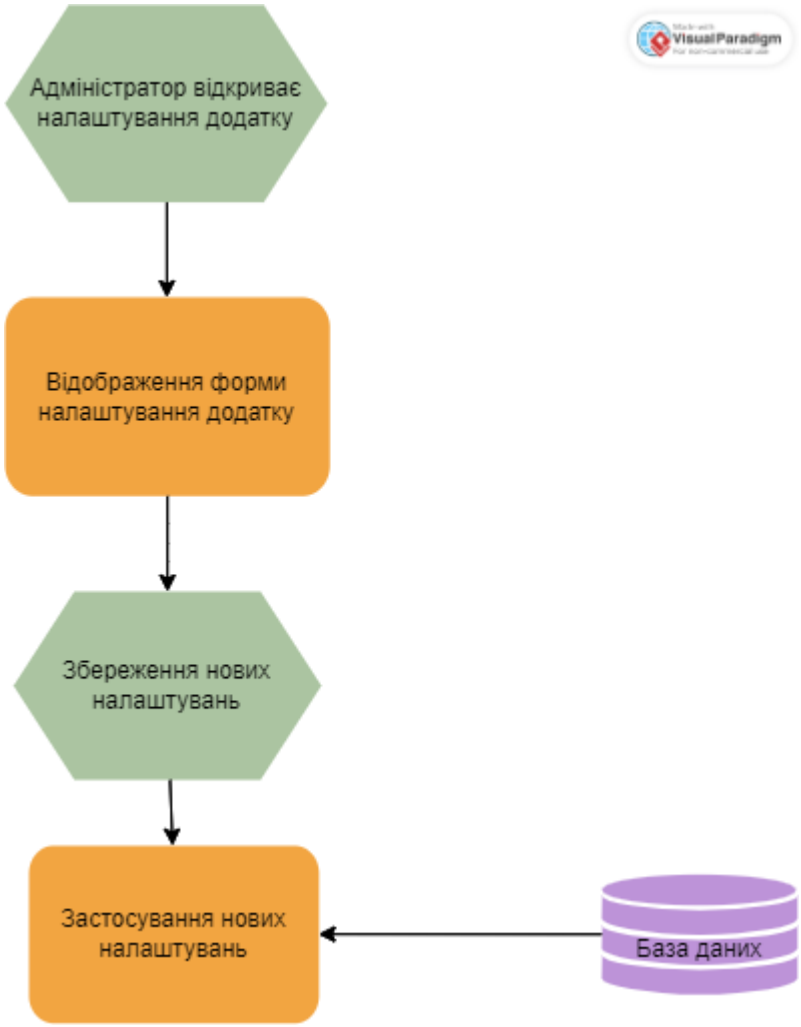


Рисунок 2.3 – діаграма EPC для адміністративних функцій

2.1.3 Діаграма Swimlane

Розробимо діаграму swimlane, що відображає розподіл ролей та процесів між користувачем, системою, експертом та адміністратором у контексті створення, проведення та обробки результатів голосування. Діаграма допомагає візуалізувати, як різні ролі взаємодіють між собою, що сприяє кращому розумінню робочого процесу та спрощує його оптимізацію. Вона також демонструє можливі розгалудження в процесі проведення експертизи, що дозволяє врахувати різні сценарії роботи з експертами та адаптації до нових експертних даних.

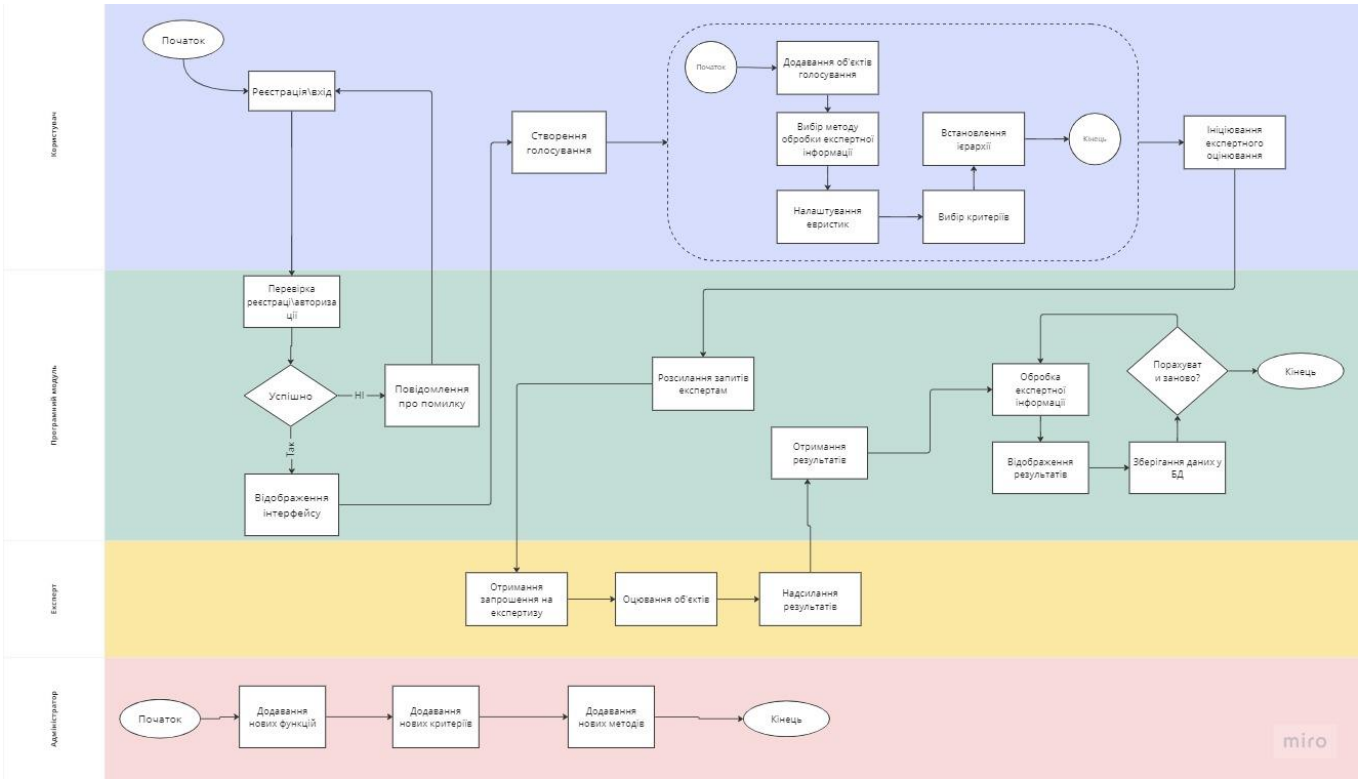


Рисунок 2.4 – діаграма Swimlane програмного модулю

2.2 Аналіз основних процесів предметного середовища

Основні процеси вашого предметного середовища для дипломної роботи пов'язані з розробкою та впровадженням системи голосування з можливістю експертного оцінювання. Користувач взаємодіє з веб-інтерфейсом для створення нового голосування, додавання об'єктів голосування, вибору методів обробки експертних даних, ініціювання експертного оцінювання та обробки результатів голосування.

Основні цілі системи:

- Автоматизація процесу голосування з експертним оцінюванням
- Можливість додавання об'єктів голосування та критеріїв оптимальності
- Забезпечення взаємодії з експертами для отримання їх оцінок

Складові елементи:

- Підсистема керування голосуванням
- Підсистема обробки експертних даних

- Підсистема адміністративного управління

Опис профілів зацікавлених сторін:

Внутрішні зацікавлені сторони:

- Розробник додатку
- Адміністратор

Зовнішні зацікавлені сторони

- Користувач (ініціатор голосування)
- Експерти

2.2.1 Діаграми IDEF0

Структурно-функціональне моделювання цього процесу згідно стандарту IDEF0 в форматі "to-be" включає такі елементи:

Назва процесу: Проведення голосування на основі експертних даних

Вхідні дані: Об’єкти голосування, налаштування голосування, експертні оцінки

Вихідні дані: Результати голосування (5 переможців)

Управління: Інформація у базі даних, правила роботи з сервісом

Механізми: Технічне забезпечення, користувач, веб-додаток

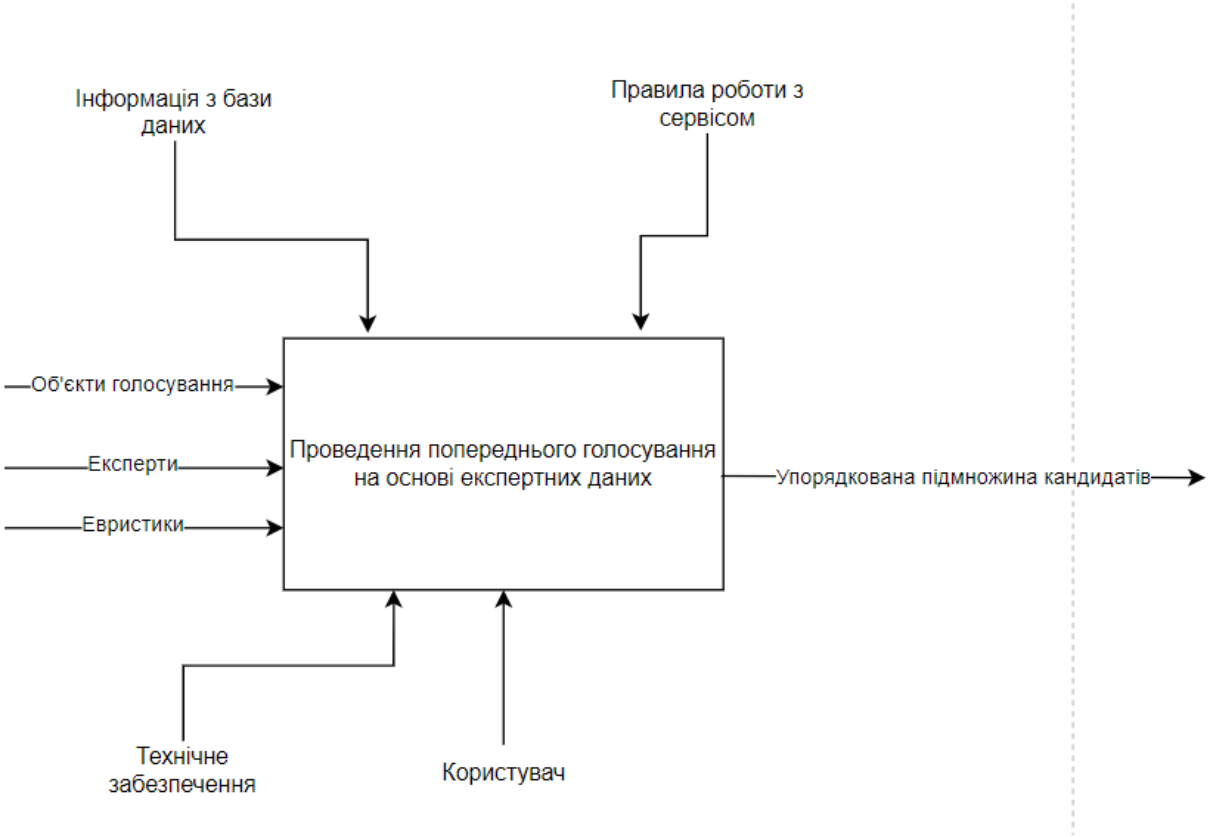


Рисунок 2.5 – контекстна діаграма

Наступний крок – проведення декомпозиції основного процесу

Діаграма декомпозиції першого рівня представлена на рис 2.6

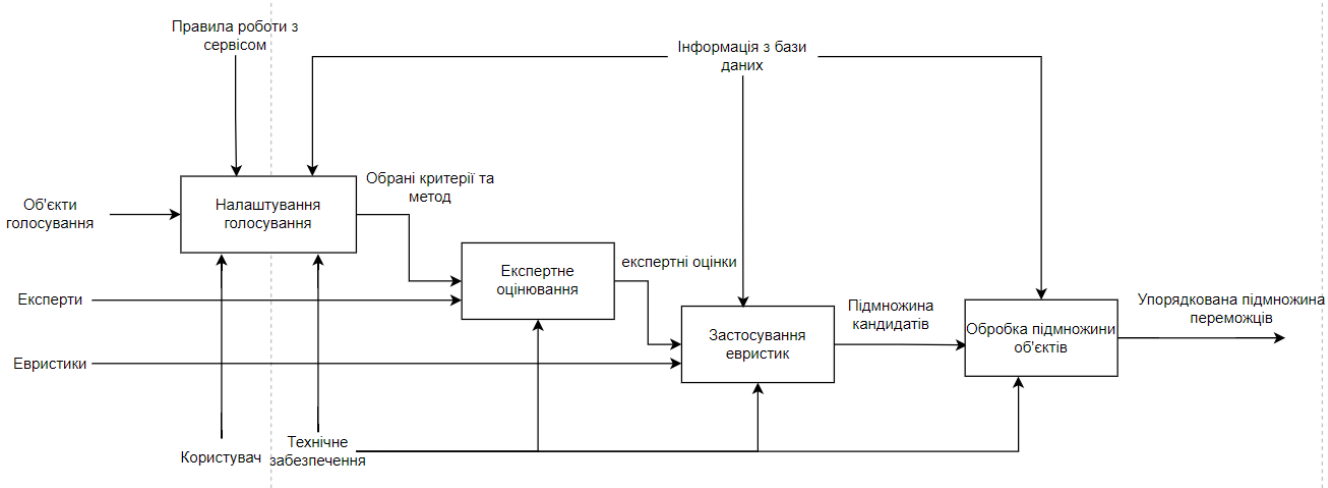


Рисунок 2.6 – Діаграма декомпозиції першого рівня

Проведемо декомпозицію процесу «Експертне оцінювання»(рис. 2.7)

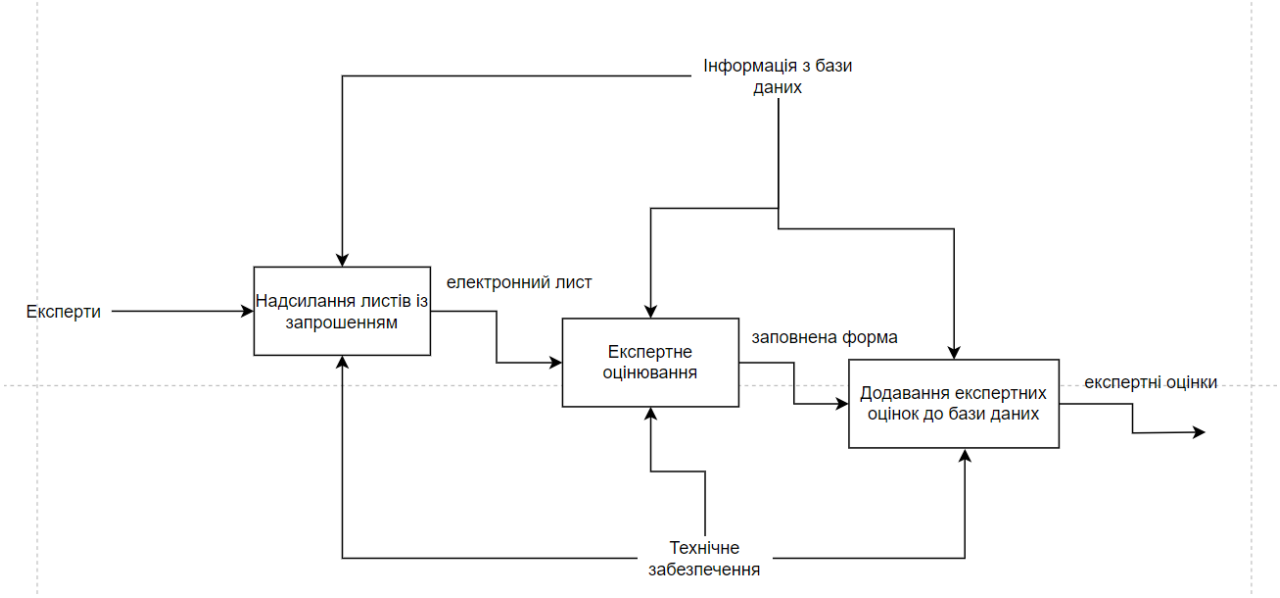


Рисунок 2.7 – Діаграма декомпозиції процесу «Експертне оцінювання»

2.3 Архітектура інтелектуальної системи

2.3.1 Структура програмного модулю

Застосунок буде складатися з двох частин – клієнтської системи та сервера, що будуть спілкуватися по протоколу HTTP.

Рис 2.7 – структура програмного модулю

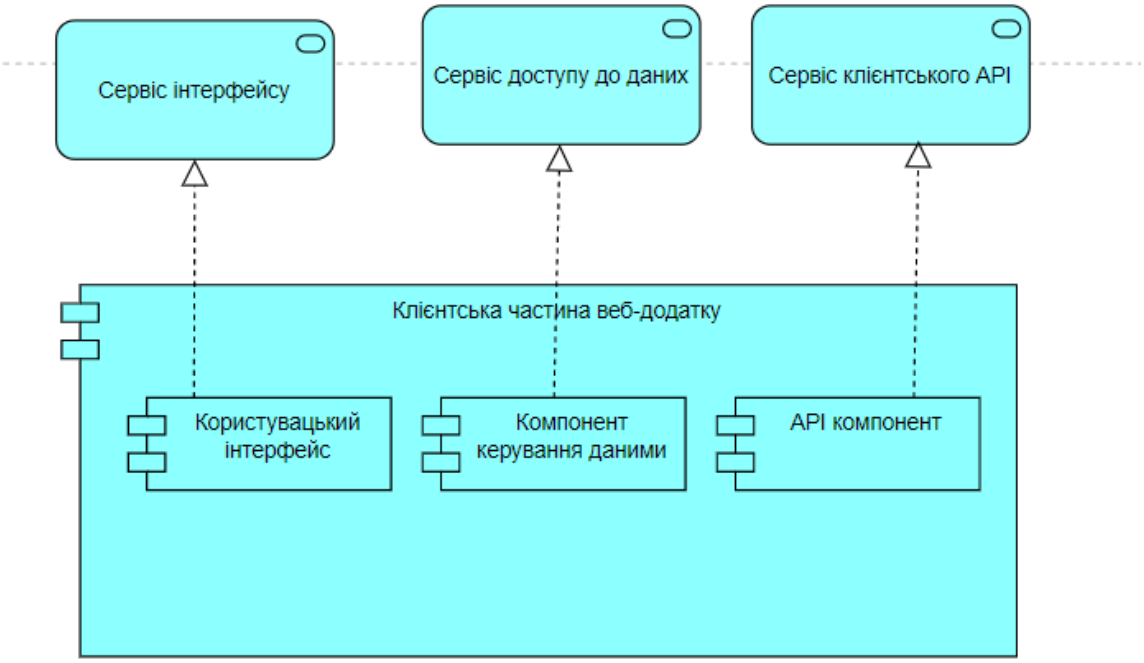


Рис 2.8 – структура клієнтської частини

Клієнтська частина:

- Користувацький інтерфейс (Application Component)

1. Сервіс інтерфейсу: відповідає за створення віджетів, візуалізацію даних та взаємодію з користувачем.

- Компонент керуванням та доступу до даних (Application Component)

2. Сервіс обробки даних: відповідає за збереження даних у локальному сховищі та доступ до них.

- Компонент API (Application Component)

3. Сервіс клієнтського API: відповідає за взаємодію з серверними API.

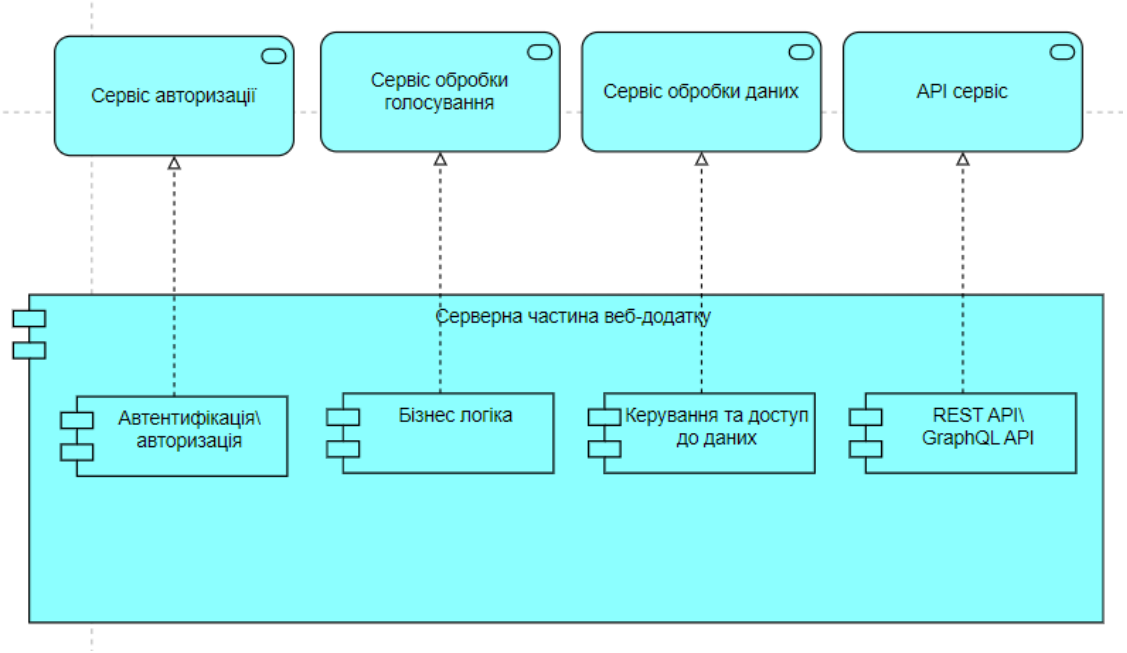


Рис 2.9 – структура серверної частини

Серверна частина:

- Автентифікація\авторизація (Application Component)
  - Сервіс авторизації: відповідає за аутентифікацію та авторизацію користувачів.
- Бізнес логіка (Application Component)
  - Сервіс обробки голосування: відповідає за обробку даних голосування, відповідно до евристик і правил.
- Керування та доступ до даних (Application Component)
  - Сервіс обробки даних: відповідає за доступ до даних та їх зберігання в базі даних.
- REST API або GraphQL API (Application Component)
  - API Сервіс: відповідає за обробку HTTP-запитів від клієнтської частини та повернення відповідей у JSON або GraphQL форматі.

2.3.2 Діаграма класів

Діаграма класів (class diagram) використовується для представлення статичної структури моделі системи в термінології класів об'єктно - орієнтованого програмування.

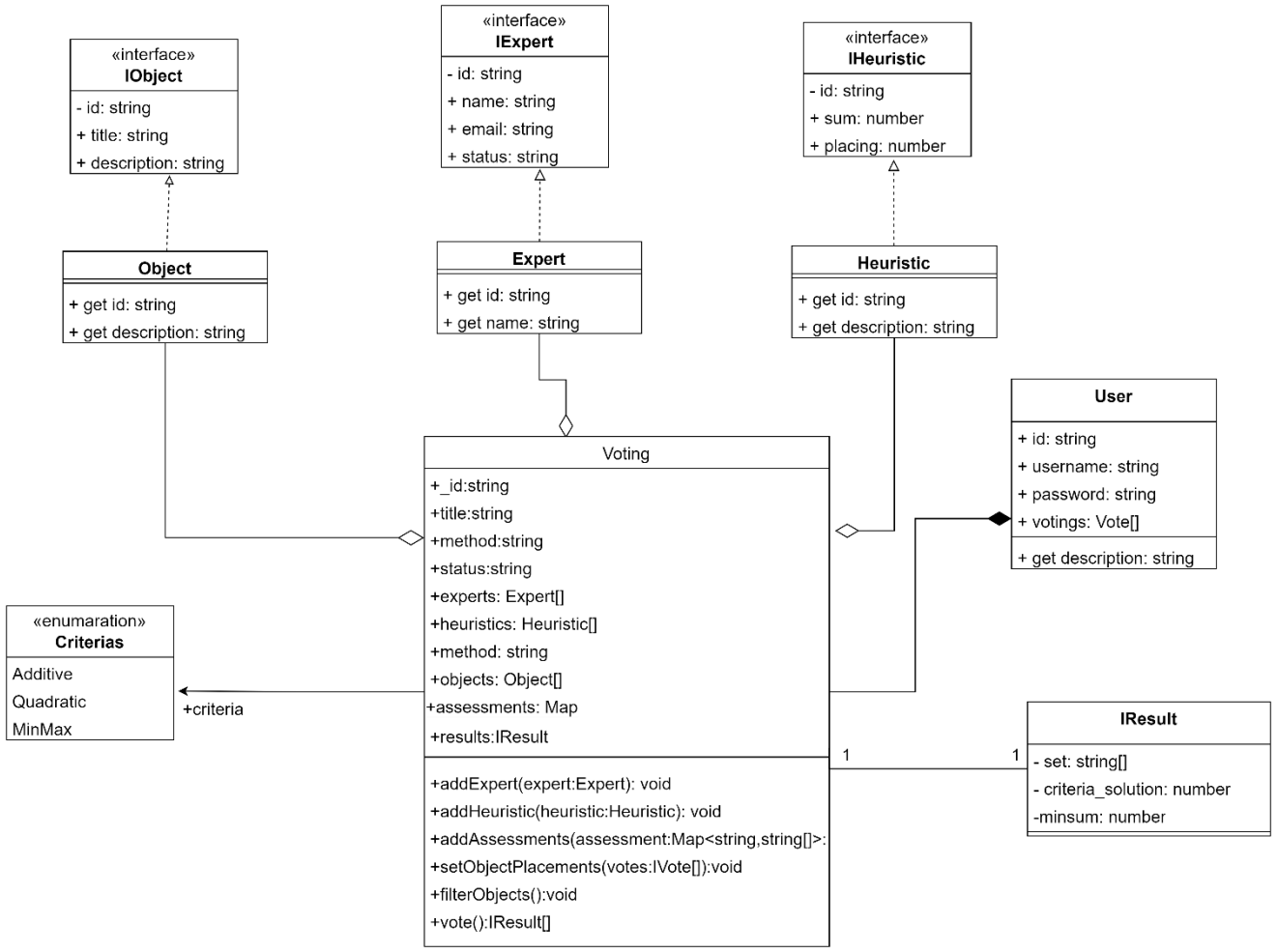


Рис 2.10 – діаграма класів

2.3.3 Діаграма розгортання

Діаграма розгортання — діаграма на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду.

Діаграму розгортання системи можна описати наступним чином. Користувач взаємодіє із системою шляхом використання веб-додатку для голосування. Додаток

відправляє запити та отримує відповіді від web-сервера. Сервер надає користувацький інтерфейс.

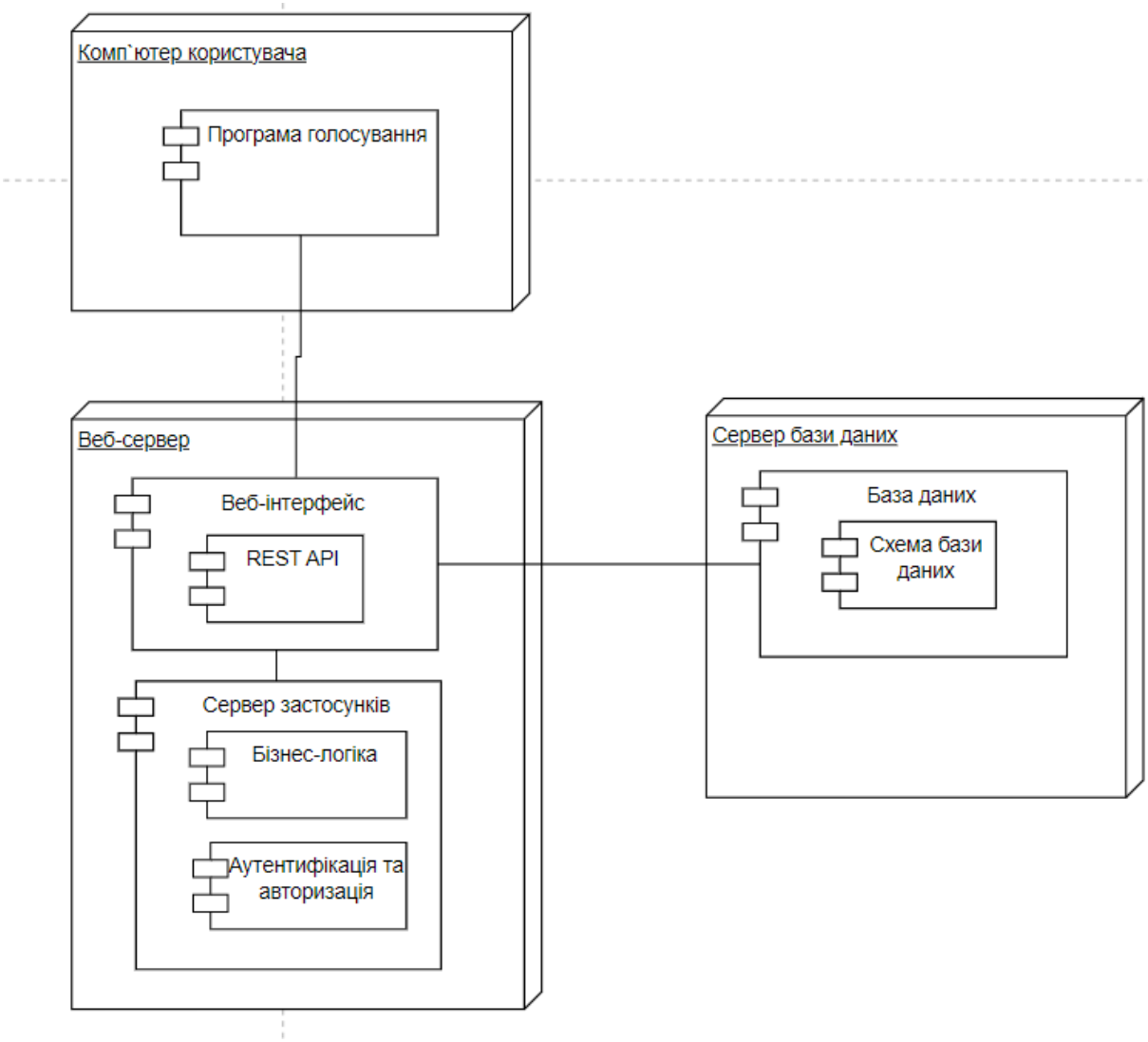


Рис 2.11 – діаграма розгортання

2.3.4 Діаграма варіантів використання

Розробимо діаграму варіантів використання. Діаграма варіантів використання відображає основні дії та процеси, які користувач може виконувати в рамках програмного модуля. Вона містить акторів, варіанти використання та відносини між ними, що допомагає краще зрозуміти функціональність системи.

Звичайний користувач – основний користувач програмного модуля, який може:

- Авторизуватися або зареєструватися у системі, відновлюючи пароль за потреби
- Створювати нове голосування, додавати експертів, критерії та об'єкти, та надсилати запрошення експертам
- Відслідковувати статус голосування
- Переглядати результати голосування
- Застосовувати евристики для відбору підмножини переможців

Адміністратор – користувач з розширеними правами доступу, який може:

- Авторизуватися у системі та відновлювати пароль
- Керувати користувачами (створення, редагування, видалення)
- Переглядати список всіх голосувань
- Редагувати голосування (зміна статусу, додавання/видалення експертів, введення нових евристик)
- Перевіряти результати голосування
- Керувати системою (наприклад, налаштування сервера, оновлення програмного модуля, керування безпекою)

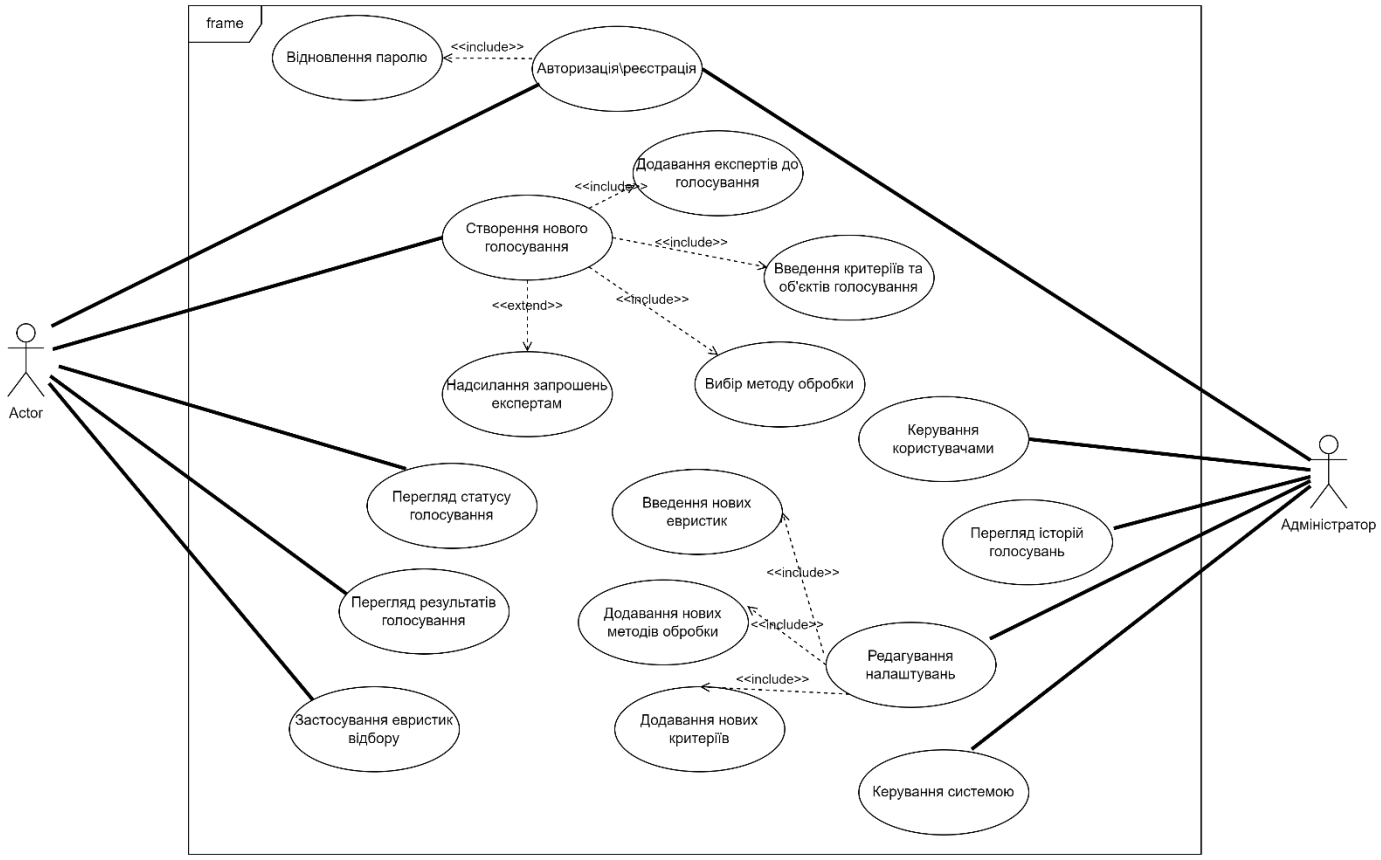


Рис 2.12 – діаграма варіантів використання

2.3.2 Розробка моделі даних

Для даного програмного модуля, була обрана нереляційна, документоорієнтовна база даних – MongoDB. Створимо концептуальну та логічну моделі бази даних для відображення сутностей та їх відношень.

На рис. 2.13 представлена концептуальна модель бази даних

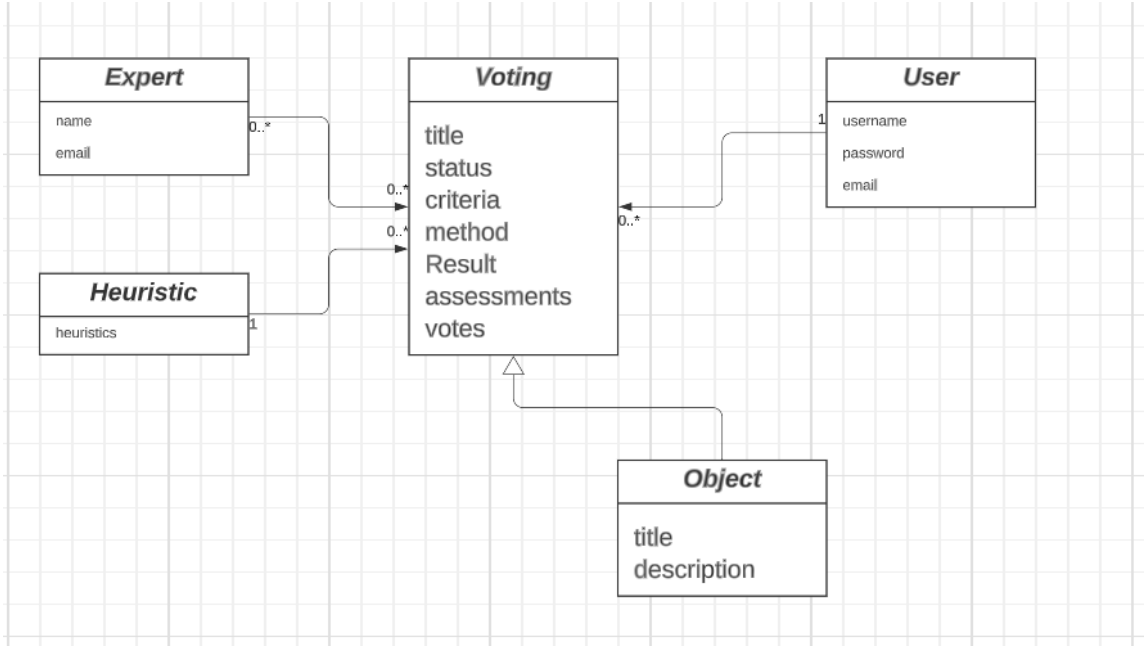


Рисунок 2.13 – концептуальна модель бази даних

На основі концептуальної моделі бази даних розробимо логічну модель бази даних, що буде відображати відносини у документорієнтовній моделі бази даних. На рис 2.14 зображена логічна модель бази даних.

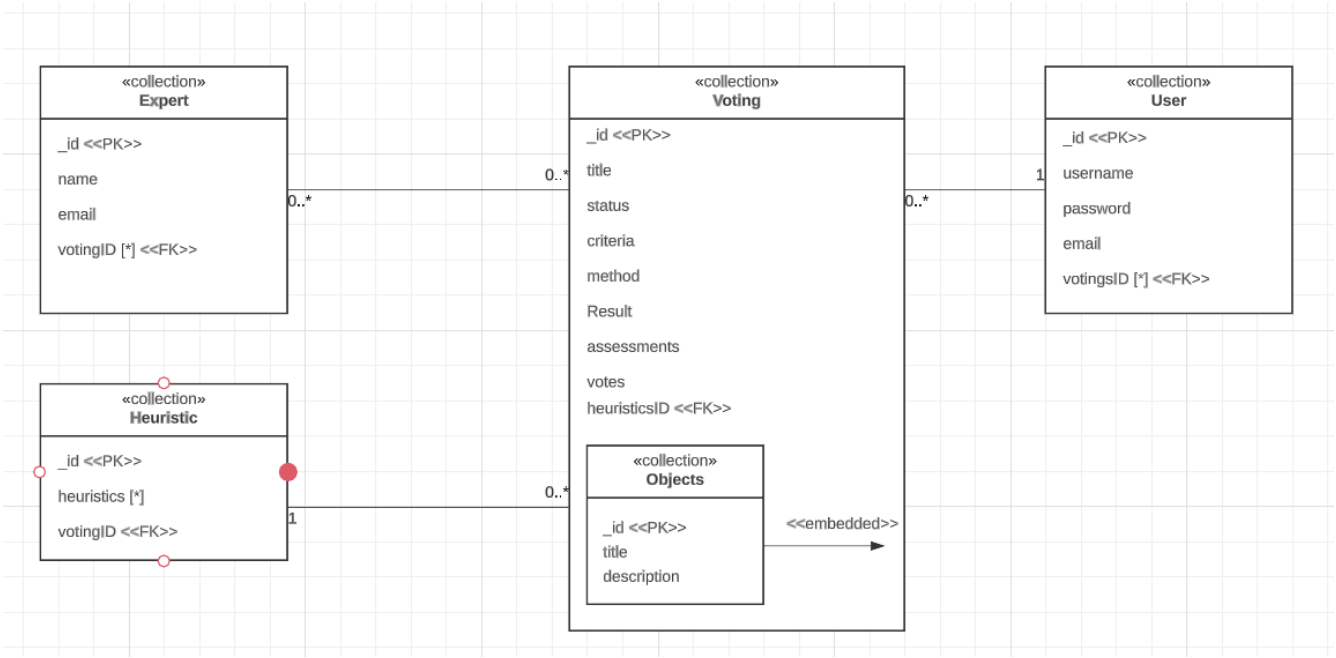


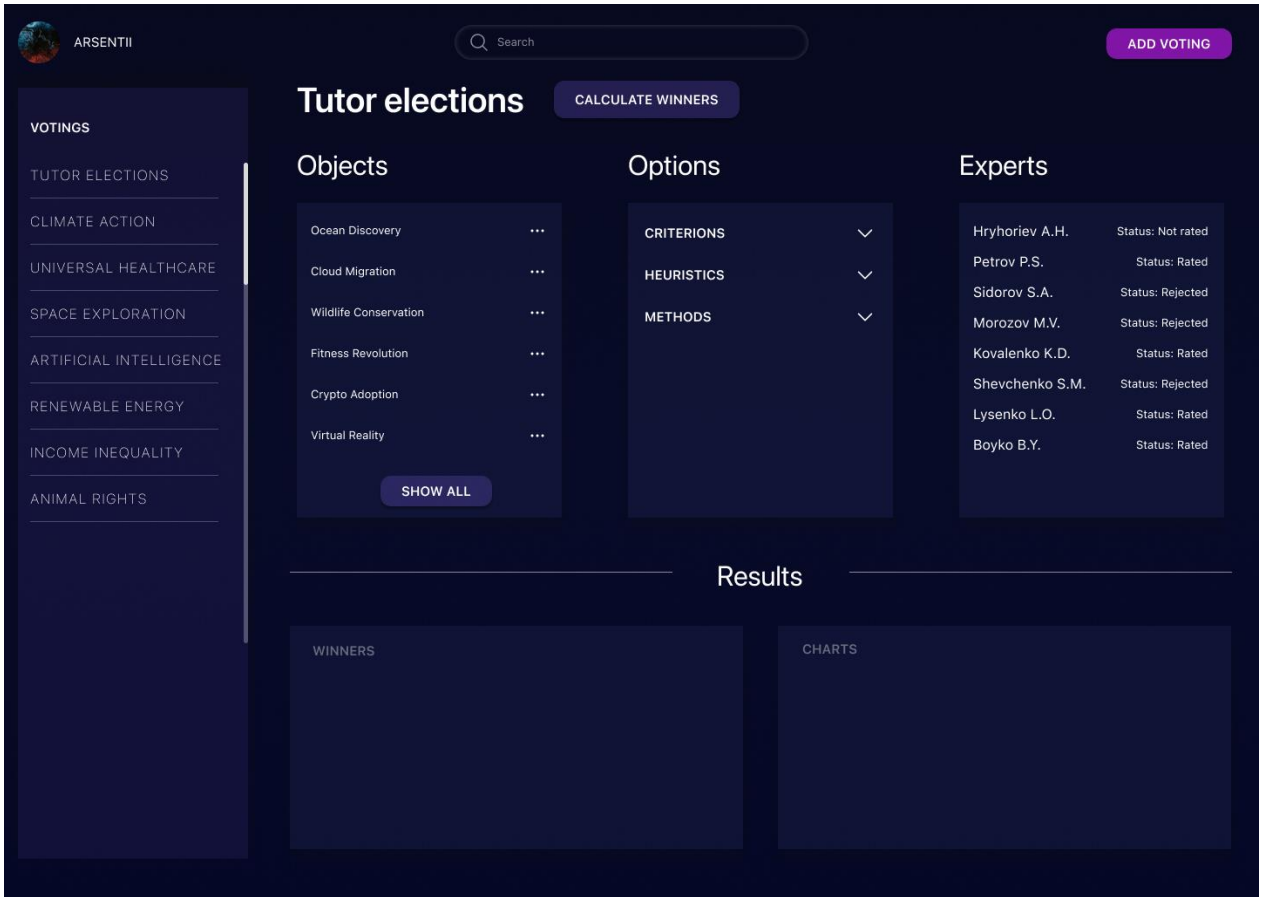
Рисунок 2.14 – логічна модель бази даних

2.3.3 Інтерфейс програмного модулю

Розробимо каркас інтерфейсу нашого програмного модулю. Програмний модуль буде скаладится з компонентів хедеру, сайдбару та головної частини

Рис 2.15 – макет програмного модулю

Далі, деталізуємо інтерфейс програмного модулю, додамо зображення та текст.



2.16– дизайн інтерфейсу програмного модулю

2.4 Математичне забезпечення: аналіз та вибір моделей та методів розв’язання задачі

2.4.1 Задача ранжування

Задача ранжування об'єктів - це процес впорядкування множини об'єктів відповідно до певних критеріїв чи характеристик. Ця задача зазвичай виникає у ситуаціях, коли потрібно відібрати або прийняти рішення щодо кількох альтернатив, які можуть бути оцінені на основі одного або декількох критеріїв.

Ціль ранжування полягає у встановленні відносної важливості або пріоритетності об'єктів в множині. Результатом ранжування є список або послідовність об'єктів, відсортованих від найкращого до найгіршого (або навпаки) згідно з певними критеріями.

Одним з ключових аспектів ранжування об'єктів є забезпечення об'єктивності та справедливості процесу відбору. Для цього може бути використано експертні оцінки, коли декілька експертів незалежно відповідають на питання про ранжування об'єктів, а потім їх відповіді агрегуються для отримання загального ранжування. Експертні оцінки можуть допомогти врахувати різні точки зору та знання експертів, що може забезпечити більш точне та репрезентативне рішення.

Постановка задачі

Нехай k експертами здійснюється попередня експертиза n деяких проєктів, об’єктів чи претендентів на посаду з множини А. За умовами експертизи, члени колективу (ЧК) на першому етапі мають задати множинні порівняння (МП) v об’єктів, причому $v \ll n$. Додатковою вимогою може бути фіксоване значення v для усіх експертів. Верхнє обмеження значення v обумовлене психо-фізіологічними можливостями людини. Воно не може бути більшим, наприклад, ніж 9, зважаючи на число Міллєра 7 ± 2.

Будемо позначати вибрану $i-$ м ЧК підмножину претендентів, на якій ЧК встановлює відношення порядку у вигляді МП, через $A_i^v, i \in I = \{1, \dots, k\}, v << n$.

Зазначимо, що на початковому етапі кількість ЧК і кількість об’єктів співпадають:

$k = n$. Не зменшуючи загальності, будемо вважати, що

$$A_j^v = \{a_{i_1}^j > \cdots > a_{i_2}^j > \cdots a_{i_k}^j\}, \text{ для усіх ЧК: } j \in I. \quad (2.1)$$

Така процедура може розглядатися як проміжний етап при переході від неформального лідерства, тобто процесу впливу на колектив за допомогою своїх здібностей, вміння, до формального – процес впливу на людей з позиції займаючої посади.

Тобто, в результаті таких виборів людина об’єднує два аспекти і стає лідером:

- особистість, за якою ЧК визнають право брати на себе найбільш відповідальні рішення, що зачіпають їхні інтереси;

- особа, на яку офіційно покладені функції управління колективом і організації його діяльності;
- особа, признана групою авторитетною, що користується визнанням та авторитетом в групі.

Необхідно знайти результуюче ранжування альтернатив, узгоджене з заданими МП експертів з урахуванням метрики неспівпадання рангів та різних критеріїв. Тобто, необхідно знайти обгрунтований порядок, найближчий до заданих експертних МП. Такі підходи можуть бути застосовані у проблемних колективах. Метою процедури попереднього ранжування об'єктів можуть бути:

- мінімізація рівня конфлікта, який назріває, або перейшов уже в активну стадію;
- м'яка перевірка рівня підтримки потенційних лідерів у колективі
- оцінка наявних угрупувань в колективі, тобто структурування його полярності і одержання таким чином більше інформації про колектив, розподіл симпатій ЧК, популярність та кількість лідерів тощо;
- визначення коефіцієнтів компетентності ЧК, причому очевидно, що експерти з найбільшою відносною компетентністю є найбільшими реалістами і можуть бути безпосередньо залучені до вирішення конфлікта.

Формалізація задачі визначення результуючого ранжування

Найпоширенішим методом знаходження результуючого ранжування альтернатив є обчислення медіани заданих ранжувань. Ця група методів узагальнення експертної інформації є найбільш достовірною та математично обґрунтованою. Розв'язки задачі, які визначаються при застосуванні різних метрик та різних критеріїв, є медіанами заданих експертами лінійних порядків. Зокрема, це пов'язано з тим, що теорія вимірювання передбачає для порядкових шкал визначення середніх величин у вигляді медіан. Тому методи середніх рангів тощо у таких випадках виглядають щонайменше наближеними та сумнівними: вони не можуть забезпечити упевненість у адекватному виборі лідера.

Якщо застосовувати бальне оцінювання для розв’язання цієї задачі, обов’язково матиме місце відхилення, тому що усі ЧК є взаємопов’язаними між собою. Адже організація є живим організмом, і опосередковано думки кожного ЧК не є автономними. Тому оцінювання можливих претендентів у балах означало б не врахування, руйнування взаємозв’язків між ЧК.

У таких випадках, що розглядається, замість бального оцінювання застосовуєтья оцінка відношень у вигляді:

- матриць попарних порівнянь між об’єктами;
- ранжувань об’єктів;
- множинні порівняння між об’єктами;
- неповні ранжування на множині об’єктів.

Підходи до визначення результуючого ранжування об’єктів

Відстані між ранжуваннями альтернатив визначаються з використанням метрики Кука неспівпадання рангів (місць, позицій) альтернатив,

$$d\left(R^j, R^l\right) = \sum_{i \in I} \left| r_i^j - r_i^l \right|, \quad (2.2)$$

, де r_i^j – ранг $i-$ї альтернативи у ранжуванні $l-$го експерта, $R^l, l \in L, 1 \leq r_i^l \leq n.$

Метрика Кука виду (2.2) є популярною в задачах ранжування альтернатив. Для її застосування при розв’язанні поставленої задачі аналізу неповних ранжувань введемо евристики та будемо визначити на їх основі відстані від заданих експертами ранжувань до опорного ранжування.

2.4.2 Застосування евристик у задачах експертного оцінювання

Евристика – це метод прийняття рішень, який використовує практичні, але не гарантовано оптимальні або раціональні, засоби досягнення результату. Це методи вирішення творчих, нестандартних, креативних проблем в умовах невизначеності, яка використовують метод спроб та помилок, а також результати експериментів для знаходження рішень. Евристичні методи спрямовані на зменшення варіантів

перебору і підвищують ймовірність одержання прийнятного, хоч і не завжди оптимального, розв’язку задачі. З допомогою евристичних методів часто знаходять вихід із надто складних чи не передбачуваних ситуацій. Потреби в евристиці виникають при вирішенні проблем, для яких ще не розроблено теорію, і які описуються неповними чи недостовірними даними.

Евристики представляють собою методики або "правила пальця", які допомагають ухвалювати рішення або розв'язувати складні задачі. Вони часто використовуються, коли повне аналітичне рішення є непрактичним або неможливим. Хоча евристики не завжди призводять до оптимального рішення, вони дозволяють знайти досить хороше рішення з меншими витратами часу або ресурсів

Основні види евристик включають:

- Евристика доступності. Вона полягає в ухваленні рішення на основі інформації, яка найлегше доступна або найбільш свіжа в пам'яті. Наприклад, люди можуть припустити, що авіакатастрофи є більш поширеними, ніж автомобільні аварії, через те, що авіакатастрофи частіше з'являються в новинах.
- Евристика пристосування (анкорингу). Люди використовують початкову інформацію як "якір" і потім коригують своє рішення на основі нової інформації. Наприклад, при оцінці вартості товару люди часто виходять з його роздрібної ціни і потім коригують свою оцінку вверх або вниз на основі додаткової інформації.
- Евристика представництва. Люди оцінюють ймовірність події на основі того, наскільки схожа ця подія на існуючі моделі або сценарії. Наприклад, при оцінці ймовірності того, що людина є програмістом, люди можуть розглядати, наскільки характеристики цієї людини схожі на стереотипні характеристики програміста.

Евристики можуть бути класифіковані за декількома критеріями: походженням (досвід, інтуїція, випадковість), типом дії (пошук, оптимізація, навчання) та областю застосування (рішення задач, управління, дизайн тощо).

У нашому випадку, використовуються евристики для експертного оцінювання, які допомагають зменшити кількість об'єктів для подальшого аналізу. Вони базуються на досвіді експертів і статистичних даних про попередній вибір.

Евристики використовуються для вибору найкращих об'єктів з множини експертних оцінок. Вони базуються на позиції об'єктів у порівнянні вибраному експертом.

Наприклад: Евристика E1.Участь в одному множинному порівнянні на 3 місці. E1 видаляє об'єкт, який знаходиться на 3-му місці в одному порівнянні, E2 - на 2-му місці, і так далі.

Ці евристики є проблемно-орієнтованими, оскільки вони спеціально розроблені для цього типу задач. Вони дозволяють значно зменшити обсяг даних для подальшого аналізу і є досить ефективними в рамках даної системи.

Застосування цих евристик дає змогу користувачеві керувати процесом вибору, встановлюючи кількість евристик для застосування та обмежуючи кількість елементів, що мають залишитися в множині A2 для прямого перебору. Такий підхід підвищує гнучкість системи та робить її більш адаптивною до потреб користувача.

Додатково, розглядаючи приклади евристик, можна побачити, що вони дозволяють відбирати об'єкти на основі їхньої "сили" в експертному виборі. Об'єкти, які частіше з'являються на верхніх позиціях у порівняннях, мають більший шанс залишитися в підмножині переможців. Це відображає інтуїтивну логіку, що об'єкти, які вибираються експертами як кращі, ймовірно, мають більше переваг.

Застосування евристик стає ключовим елементом у процесі прийняття рішень. Вони не тільки допомагають упоратися з великою кількістю даних, але і забезпечують адаптивність системи, дозволяючи користувачу контролювати процес на основі своїх потреб та вимог.

2.4.3 Класичні правила вибору

Розглянемо класичні правила вибору, які використовуються для ранжування альтернатив чи кандидатів на основі експертного оцінювання. Проаналізуємо основні принципи, формули та переваги та недоліки кожного методу, що допоможе зрозуміти їх застосування та обмеження в різних ситуаціях прийняття рішень.

Правило Кондорсе

Метод Кондорсе (також відомий як метод парних порівнянь) - це класичний метод голосування, запропонований французьким математиком і філософом маркізом де Кондорсе у 18 столітті. Цей метод базується на попарних порівняннях кандидатів за кожним критерієм і визначенні переможця на основі загального числа перемог в попарних зіставленнях.[3]

Метод Кондорсе можна описати наступним чином:

Проводиться голосування, в якому виборці розставляють кандидатів в порядку своїх уподобань.

Для кожної пари кандидатів (A, B) порівнюється число виборців, які віддають перевагу кандидату A над кандидатом B, і навпаки.

Якщо існує кандидат, який перемагає у всіх попарних порівняннях, цей кандидат вважається переможцем голосування за методом Кондорсе.

Формула для методу Кондорсе може бути записана наступним чином:

$P(A, B)$ = число виборців, які віддають перевагу кандидату A над кандидатом B

$P(B, A)$ = число виборців, які віддають перевагу кандидату B над кандидатом A

Кандидат A вважається переможцем за методом Кондорсе, якщо для всіх кандидатів B виконується нерівність: $P(A, B) > P(B, A)$.

Переваги методу Кондорсе полягають у його здатності враховувати всі можливі попарні порівняння кандидатів, що дозволяє отримати більш повну

інформацію про виборчі уподобання. Крім того, цей метод зазвичай приводить до вибору кандидата з найбільшою підтримкою виборців в агрегованому вигляді.

Правило Борда

У цьому правилі за останнє місце кандидата йому нараховується 0 балів (очок), за передостаннє -1, … , за перше – (m-1).

Розглянемо профіль (таблиця 1). Маємо: $n_a = 24, n_b = 22, n_c = 27, n_d = 29$. Перемагає кандидат, що набрав найбільшу кількість балів, у нашому випадку – це кандидат d.

5	3	5	4	S
a	a	b	c	3
d	d	c	d	2
c	b	d	b	1
b	c	a	a	0

Таблиця 1 – профіль для правила Борда

Правило Сімпсона

Правило Сімпсона, також відомий як метод мінімального протистояння, є ще одним класичним методом голосування. Він базується на ідеї попарного порівняння кандидатів, але, на відміну від методу Кондорсе, він фокусується на найслабших результатів кожного кандидата в попарних протистояннях. Метод Сімпсона визначає переможця як того кандидата, який має найменше слабке протистояння серед усіх кандидатів.[3]

Щоб застосувати метод Сімпсона, спочатку треба провести попарні порівняння між кандидатами, підраховуючи кількість голосів, які кожен кандидат отримує в кожному протистоянні. Запишемо результати у матрицю попарних протистоянь P, де $P(i, j)$ відображає кількість голосів на користь кандидата i проти кандидата j.

Наступним кроком є визначення слабкості кожного кандидата. Для кожного кандидата i знайдемо його найгірший результат у матриці P, тобто мінімальне

значення серед всіх $P(i,j)$ для фіксованого i. Слабкість кандидата і буде дорівнювати цьому мінімальному значенню: $W(i) = min\{P(i,j) \mid \text{для всіх } j \neq i\}$.

Переможцем за методом Сімпсона вважається кандидат з найбільшою слабкістю, тобто кандидат і, для якого $W(i)$ є максимальним значенням серед усіх кандидатів: переможець = $argmax\{i \mid W(i)\}$.

Правило відносної більшості

На перше місце 8 виборців поставили кандидата а (n_a= 8), 5 виборців – b (n_b= 5) і 4 виборці – с (n_c= 4), n_d= 0 . Перемагає той кандидат, за якого проголосувала більшість виборців (у даному випадку – а , випадок рівності голосів поки що не розглядаємо). Цей метод поважає волю більшості у тому сенсі, що якщо за певного кандидата проголосувала абсолютна більшість (тобто більше половини виборців (часто говорять – "50%+1"), то він і перемагає)3

Правило відносної більшості з вибуванням

Або ("відносна більшість у два тури", "абсолютна більшість"). За подібним правилом відбуваються вибори президента України. Якщо деякий кандидат набрав більше половини голосів, то він – переможець. Інакше у другий тур проходять два кандидати, що набрали відносну більшість голосів (тому – "відносна у два тури")[3].

2.4.4 Критерії якості розв’язків

1. Адитивний критерій якості розв'язків:

Адитивний критерій якості розв'язків базується на сумуванні значень часткових критеріїв для кожного рішення або об'єкта. Він вважається простим та інтуїтивно зрозумілиьм способом агрегації даних з різних критеріїв.

Основна формула адитивного критерію якості розв'язків:

$$F(x) = w_1 * f_1(x) + w_2 * f_2(x) + \cdots + w_n * f_n(x) \quad (2.3)$$

де $F(x)$ - агреговане значення критерію якості для рішення x; $f_i(x)$- значення часткового критерію i для рішення x; w_i- вага часткового критерію i (сума всіх ваг дорівнює 1).

2. Мінімаксний критерій якості розв'язків:

Мінімаксний критерій якості розв'язків враховує найгірший можливий випадок для кожного рішення або об'єкта. Він шукає таке рішення, яке максимізує мінімальне значення критерію якості серед усіх рішень. Мінімаксний критерій вважається консервативним, оскільки він фокусується на уникненні найгіршого результату.

Основна формула мінімаксного критерію якості розв'язків:

$$F(x) = \max(\min(f_1(x), f_2(x), \dots, f_n(x))) \quad (2.4)$$

, де $F(x)$ - агреговане значення критерію якості для рішення x; $f_i(x)$ - значення часткового критерію i для рішення x.

3. Квадратичний критерій якості розв'язків:

Квадратичний критерій якості розв'язків базується на сумі квадратів відхилень часткових критеріїв від їхніх оптимальних значень. Він вважається більш розумним та чутливим до варіацій значень часткових критеріїв, оскільки він штрафує відхилення від оптимальних значень критеріїв сильніше, ніж адитивний критерій. Квадратичний критерій також відомий як метод найменших квадратів.

Основна формула квадратичного критерію якості розв'язків:

$$F(x) = w_1 * \left(f_1(x) - f_{1opt}\right)^2 + w_2 * \left(f_2(x) - f_{2opt}\right)^2 + \dots + w_n * \left(f_n(x) - f_{nopt}\right)^2 \quad (2.5)$$

, де $F(x)$ - агреговане значення критерію якості для рішення x; $f_i(x)$ - значення часткового критерію i для рішення x; f_{iopt}- оптимальне значення часткового критерію i; w_i- вага часткового критерію i (сума всіх ваг дорівнює 1).

Ці три критерії якості розв'язків допомагають визначити найкращі рішення або об'єкти за допомогою різних підходів до агрегації інформації з часткових критеріїв.

2.4.6 Алгебраїчний метод обробки експертних даних

Для визначення колективної числової оцінки алгебраїчним методом використовується експертиза 7: $\Omega = \widetilde{\Omega} = E^1$, експерти ізольовані, обернений зв'язок відсутній. Відстань d між числовими оцінками a і b визначається як $d(a,b) = |a-b|$. У якості колективної оцінки a приймаються, наприклад, оцінки:

- $a \in Arg\ min_{a\ \epsilon\varepsilon^1} \sum_{i=1}^{n} a_i d(a,a_i)$ (утилітарний критерій) (2.6)
- $\alpha\epsilon Arg\ min_{\alpha\epsilon E^1} max_{i=\overline{1,n}} a_i d(a,a_i)$ (егалітарний критерій) (2.7)

Для визначення колективного ранжування алгебраїчним методом експерти задають матриці $A^i = (a^i_{jk})$, у яких $a^i_{jk} = 1$ тоді й лише тоді, коли об'єкт i передує об'єкту k; якщо об'єкти j і k рівноцінні або $j = k, a_{jk} = 0$; якщо $a_{jk} = 1$ $(j \neq k)$, то $a_{kj} = -1$.

Ранжування A і відповідну йому матрицю A будемо позначати одним символом.

Ранжування C знаходиться між ранжуванням A і B, якщо для $\forall i,j = \overline{1,m}$ $a_{ij} \leq c_{ij} \leq b_{ij}$ або $a_{ij} \geq c_{ij} \geq b_{ij}$.

Відстань між ранжуваннями вводиться аксіоматично:

A1. $d(A,B) \geq 0$, причому $d(A,B) = 0 \leftrightarrow A = b$;

A2. $d(A,B) = d(B,A)$ (симетричність);

A3. $d(A,B) + d(B,C) \geq d(A,C)$, причому рівність досягається тоді й лише тоді, коли ранжування B знаходиться між ранжуваннями A і C (аксіома трикутника).

A4. При однакових перестановках об'єктів у ранжуваннях A і B відстань між отриманими ранжуваннями $d(A',B') = d(A,B)$ (інваріантність відносно позначень);

A5. Якщо двоє ранжувань відрізняються одне від одного лише на частині об'єктів, то відстань між початковими ранжуваннями дорівнює відстані між ранжуваннями лише цих об'єктів;

A6. Мінімальна додатня відстань між ранжуваннями дорівнює 1.

Аксіоми А1-А6 однозначно визначають відстань (відстань Хемінга) $d(A,B)$ при будь-якій довжині ранжувань $m \geq 2$, а формула: $d(A,B) = 0.5 \times \sum_{i,j=1}^{m} \left| a_{ij} - b_{ij} \right|$, визначає єдину відстань $d(A,B)$, що задовольняє аксіомам А1-А6.

Експертиза 8: $\Omega = \widetilde{\Omega} =$ { матриці A_i, елементи яких визначені вище}, експерти ізольовані, обернений зв'язок відсутній. У якості відстані береться відстань Хемінга, колективне ранжування визначається критеріями:

- $A^{KS} \in Argmin_{A \in \tilde{A}} \sum_{i=1}^{n} a_i d\left(A, A^i\right)$ (медіана Кемені-Снелла); (2.8)
- $A^{VG} \in Argmin_{A \in \tilde{A}} max_{i=\overline{1,m}} \sum_{i=1}^{n} a_i d\left(A, A^i\right)$ (компроміс); (2.9)
- $A^{SZ} \in Argmin_{A \in \tilde{A}} \sum_{i=1}^{n} a_i d^2\left(A, A^i\right)$ (середнє значення); (2.10)

Вище $\tilde{A}$ – множина матриць $m \times m$ з елементами $a_{ij} \in \{+1, -1, 0\}$, що відповідають ранжуванням (тобто матриці ациклічні). Як видно – критерій 1 відповідає принципу утилітаризма, критерій 2 – егалітаризма.

Усі ці критерії визначають колективне ранжування як аргумент, що мінімізує суму відстаней до всіх експертних ранжувань, причому ваги a_i можуть враховувати надійність відповідного експерта.

Проте, практичне використання цих критеріїв вимагає розв'язання задачі комбінаторної оптимізації, що є NP-складною. Тому зазвичай замість знаходження точного рішення застосовуються різні евристичні алгоритми.

Один з таких алгоритмів - це алгоритм Borda, який використовує рейтингові бали для кожного об'єкта. Колективне ранжування тоді визначається як ранжування, отримане сортуванням об'єктів за сумою їх рейтингових балів.

Можливо також застосування більш складних евристик, таких як алгоритми генетичного пошуку, ант колонії або імітація відпалу, для вирішення даної задачі.

Кемені-Снелла та критерій середнього значення працюють на принципі мінімізації суми відстаней, тоді як компромісний критерій зосереджується на мінімізації максимальної відстані.

Ранжування Кемені-Снелла відображає середнє ранжування за всіма експертами і відображає загальне вподобання колективу. Середнє значення критерію намагається згладити вплив окремих відхилень від групи, зосереджуючись на середньому значенні відстаней. Ці два критерії найкраще використовувати, коли метою є знаходження загального консенсусу серед експертів.

З іншого боку, компромісний критерій намагається знайти ранжування, яке максимально наближене до всіх інших, зосереджуючись на мінімізації максимальної відстані. Цей критерій є особливо корисним, коли метою є уникнення найбільшого конфлікту або розбіжності з будь-яким з експертів.

Таким чином, за допомогою алгебраїчного методу обробки експертних даних можна отримати гнучкі і адаптивні результати, що відображають різні аспекти колективного вподобання. Це надає вам можливість вибрати найбільш відповідний підхід до агрегування ранжувань в залежності від конкретних цілей та контексту вашої задачі.

2.5 Висновки другого розділу

У другому розділі було розглянуто та розроблено важливі аспекти створення програмного модулю. У пункті про вибір та обгрунтування мови програмування та технологій було представлено стек, на якому буде побудований програмний модуль, включаючи клієнтську частину на React Typescript, Tailwind, та серверну частину на MongoDB, Next.js. Функціональний аналіз допоміг у структуруванні програмного модуля за допомогою дерева функцій, діаграми Event-driven process chain та діаграми SwimLane. Це дало змогу чітко визначити основні цілі системи, такі як автоматизація процесу голосування з експертним оцінюванням, можливість додавання об'єктів голосування та критеріїв оптимальності, а також забезпечення взаємодії з експертами для отримання їх оцінок. Система складається з трьох підсистем: керування голосуванням, обробки експертних даних та адміністративного управління. За допомогою діаграм IDEF0 було проведено аналіз

основних процесів предметного середовища, що дозволило розробити архітектуру інтелектуальної системи, включаючи структуру програмного модуля, діаграму класів, діаграму розгортання, діаграму варіантів використання та інтерфейс програмного модулю. У пункті Математичного забезпечення було описано усі необхідні формули та моделі розв’язків поставленої задачі, було описано переваги та недоліки разних класичних методів вибору, описано варіанти критеріїв розв’язк

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЮ ПІДТРИМКИ ПРЕФЕРЕНЦІЙНОГО ГОЛОСУВАННЯ. ТЕСТОВІ ПРИКЛАДИ

3.1 Компоненти програмного модуля

Для розробки було обрано Next.js, який є фреймворком із серверним рендерингом, щоб уникнути навантаження на клієнтську частину. Програмний модуль складається із серверної частини, яка розташована в src/pages/api/*, папка api відповідає за усі доступні ендпоїнти програмного модуля. Уся клієнтська частина та компоненти розташована у папці src/*. Структура додатку є наступною (рис 3.0)

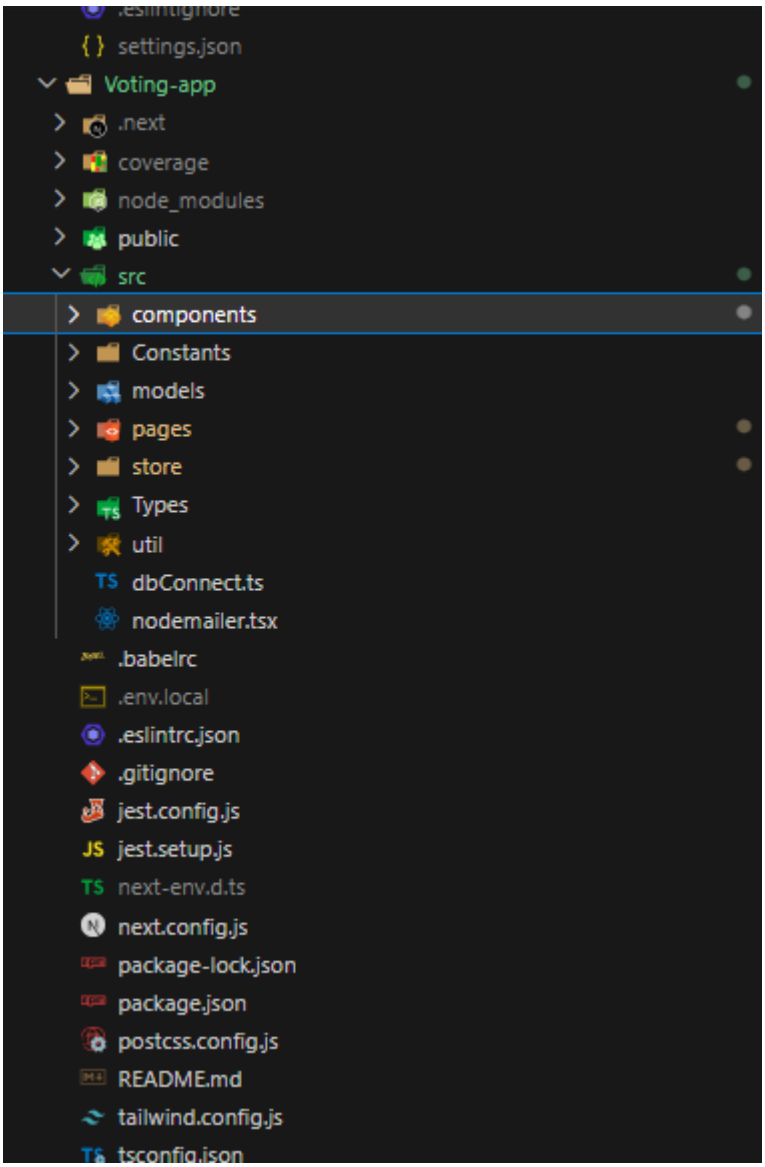


Рисунок 3.0 – Структура каталогів проєкту

Усі налаштування проєкту зроблені у відповідних конфіг файлах, підключення до бази даних відбувається у модулі dbConnect.ts. Відправка листів реалізована за допомогою бібліотеки nodemailer та реалізується в модулі nodemailer.tsx.

Каталог ‘public’ призначений для зберігання файлів клієнтської частини (зображень, шрифтів). Каталог ‘util’ використовується для обчислень, в ньому представлені усі класи програмного модуля та допоміжні функції.

3.2 Описання класів та функцій програмного модуля

Програмний модуль буде складатися з 4 основних класів: Heuristic, VotingObject, Expert та Vote.

Клас Expert - він представляє експерта, який бере участь у голосуванні.

Атрибути:

- _id: Унікальний ідентифікатор експерта.
- name: Ім'я експерта.
- email: Email адреса експерта.
- status: Статус експерта.

Методи:

- id(): Повертає ID експерта.
- description(): Повертає опис експерта у форматі "Ім'я (Email): Статус".

Клас VotingObject - ін представляє об'єкт голосування.

Атрибути:

- _id: Унікальний ідентифікатор об'єкта.
- title: Назва об'єкта.
- description: Опис об'єкта.

Методи:

- name(): Повертає назву об'єкта у форматі "Назва (Опис)".

- id(): Повертає ID об'єкта.

Клас Heuristic - ін представляє евристику, що використовується під час голосування.

Атрибути:

- _id: Унікальний ідентифікатор евристики.
- sum: Сума голосів евристики.
- placing: Розташування об’єкту голосування.

Методи:

- name(): Повертає опис евристики у форматі "Exclude Object with sum votes: sum and with placement taken: placing".
- id(): Повертає ID евристики.

Клас Vote - він представляє процес голосування.

Атрибути:

- objects: об’єкти голосування
- experts: експерти, що приймають участь у голосуванні
- heuristics: евристики, що застосовуються для отримання підмножини переможців
- results: результати попереднього голосування
- assessments: оцінки експертів
- votes: голоси, отримані для кожного об’єкта.

Методи:

- Results(): геттер класу, повертає результати
- Objects(): геттер класу, повертає об’єкти голосування
- Heuristics(): геттер класу, повертає евристики голосування
- Experts(): гетер класу, повертає список експертів голосування
- Assessments(): гетер класу, повертає оцінки експертів
- Votes(): гетер класу, повертає голоси для кожного об’єкту

- addExpert(expert: Expert): Додає нового експерта до масиву експертів experts.
- addObject(object: VotingObject): Додає новий об'єкт до масиву об'єктів objects.
- addHeuristic(heuristic: Heuristic): Додає нову евристику до масиву евристик heuristics.
- setAssessments(assessments: Map<string, string[]>): Встановлює оцінки експертів для об'єктів у вигляді мапи assessments.
- setObjectPlacements(votes: IVote[]): Встановлює голоси експертів для об'єктів у вигляді масиву votes.
- applyHeuristics(): Застосовує евристики до голосів votes та фільтрує їх згідно з заданими умовами.
- filterObjects(): Фільтрує об'єкти objects на основі відфільтрованих голосів.
- vote(): Виконує процес голосування, обчислює матрицю, застосовує критерій оцінки, обчислює результат голосування та повертає його у вигляді масиву результатів IResult.

Опишемо допоміжні функції програмного модуля:

- Функція transformAssessments(assessments: Map<string, string[]>, objects: VotingObject[]) приймає Map assessments, що містить оцінки експертів для об'єктів, та масив objects з об'єктами голосування. Функція перетворює ці оцінки в матрицю, де рядки відповідають об'єктам голосування, стовпці - експертам, а значення - місцю об'єкту в оцінках експерта. Функція повертає отриману матрицю.
- Функція generateCombinations(objects: Object[]) приймає масив об'єктів objects та генерує всі можливі комбінації цих об'єктів. Кожна комбінація представлена масивом чисел, що відповідають індексам об'єктів. Функція повертає масив всіх згенерованих комбінацій.

- Функція calculateMatrix(a: number[][], b: number[][]): number[][] | undefined приймає дві матриці a та b у вигляді двовимірних масивів чисел. Функція обчислює матрицю, яка є результатом множення матриці a на матрицю b (елементи матриці a використовуються як індекси для вибору елементів матриці b). Якщо кількість стовпців матриці a не дорівнює кількості рядків матриці b, функція повертає undefined. В іншому випадку функція повертає отриману матрицю.
- Функція findMinimaxValue(array: number[][]) приймає матрицю array у вигляді двовимірного масиву чисел. Функція знаходить мінімаксне значення (мінімальний максимальний елемент) у кожному рядку матриці та повертає об'єкт, що містить мінімаксне значення та відповідний рядок матриці.
- Функція findQuadraticCriterionValue(array: number[][]) приймає матрицю array у вигляді двовимірного масиву чисел. Функція знаходить мінімальну суму квадратів значень у кожному рядку матриці та повертає об'єкт, що містить мінімальну суму квадратів та відповідний рядок матриці.
- Функція findAdditiveCriterionValue(array: number[][]) приймає матрицю array у вигляді двовимірного масиву чисел. Функція знаходить мінімальну суму значень у кожному рядку матриці та повертає об'єкт, що містить мінімальну суму та відповідний рядок матриці.

3.3 Функціонування програмного модуля

Користувач заходить на сайт та бачить перед собою головний екран додатку (рис 3.1), для початку роботи програми, потрібно створити нове голосування або залогінитись.

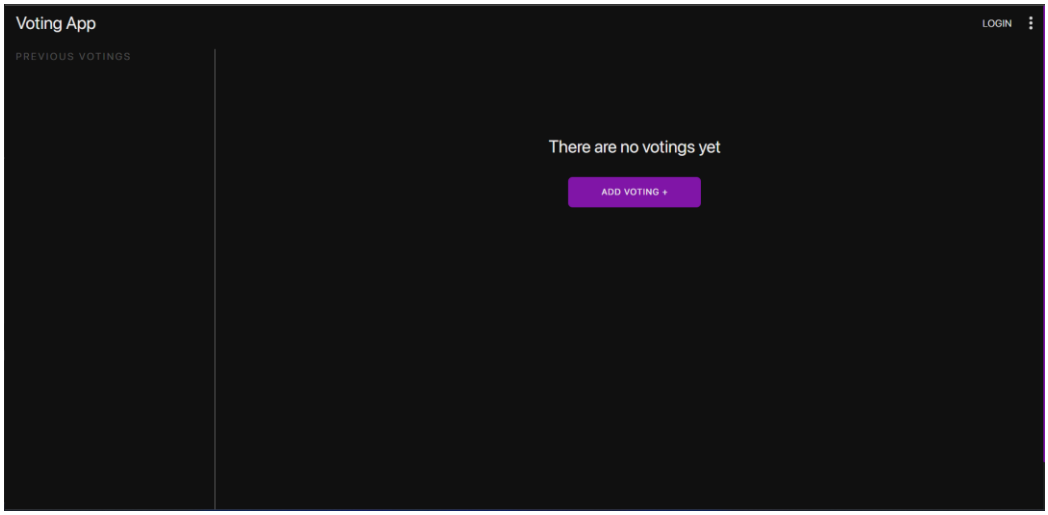


Рисунок 3.1 – головний екран програмного модуля

Якщо користувач залогінився та має у базі даних вже раніше створені голосування, то він бачить головний екран з попередньо створеними голосуваннями. (рис 3.2)

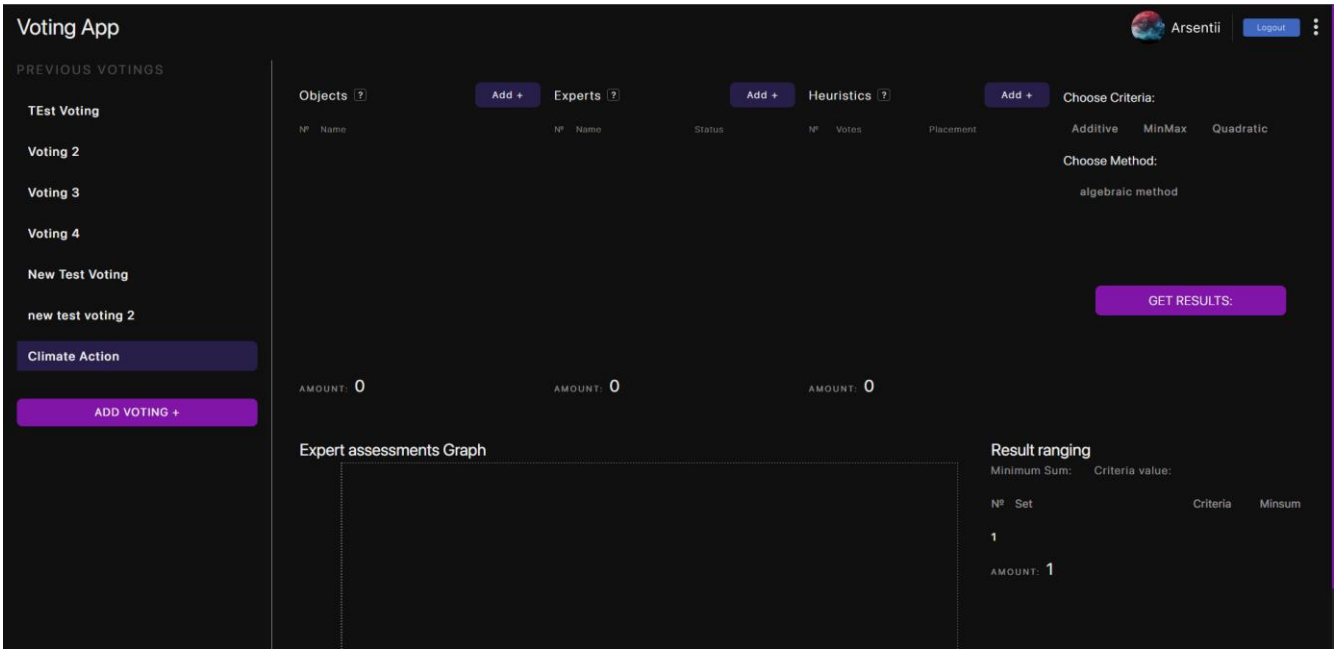


Рисунок 3.2 – головний екран додатку для залогіненого користувача

Далі потрібно створити нове голосування, натиснувши кнопку “Add Voting+” на сайдбарі. Після натискання даної кнопки відкривається вікно додавання голосування (рис.3.3)

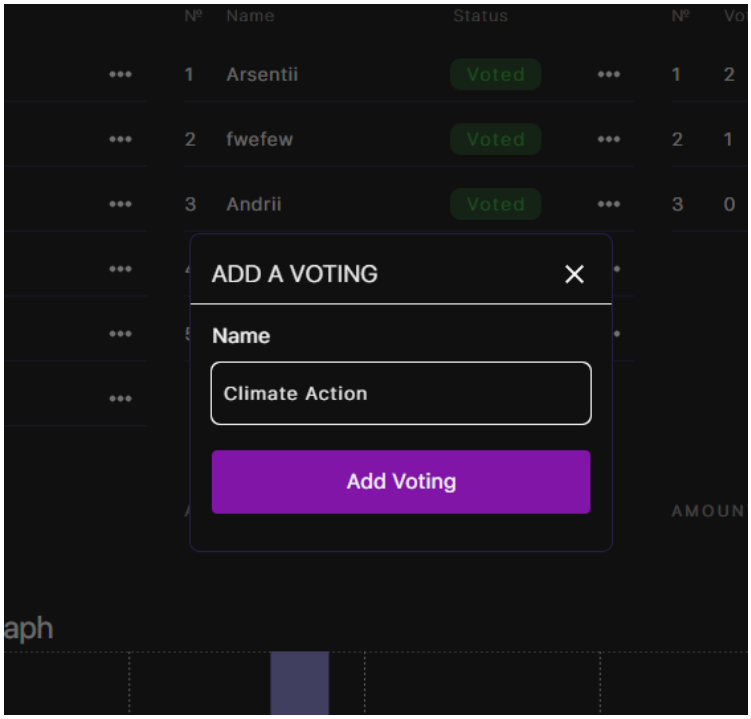


Рисунок 3.3 – вікно додавання голосування

Після додавання голосування, користувач буде бачити пустий інтерфейс головного екрану із головними елементами, які мають бути заповнені (рис 3.4)

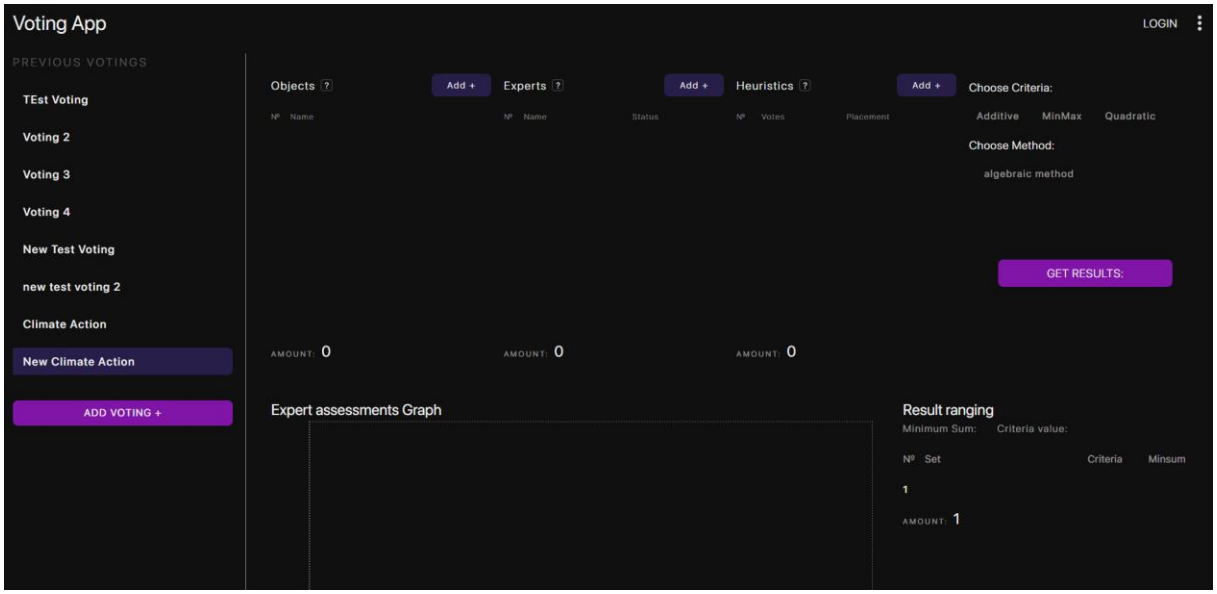


Рисунок 3.4 – головний інтерфейс додатку з новим голосуванням

Далі користувач розпочинає додавати об’єкти до голосування, натискаючи кнопку “Add+” у компоненті Objects, та отримує наступне вікно додавання об’єкту (рис 3.5)

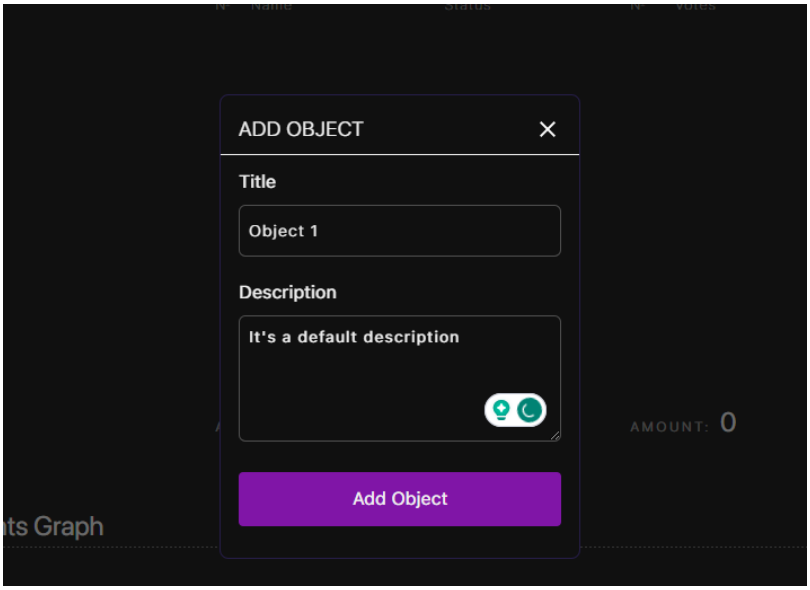


Рисунок 3.5 – вікно додавання об’єктів

Після додавання об’єкту, робиться запит на сервер, додається даний об’єкт до бази даних, та модифікується стейт інтерфейсу, новостворений об’єкт з’являється у відповідному компоненті (рис 3.6)

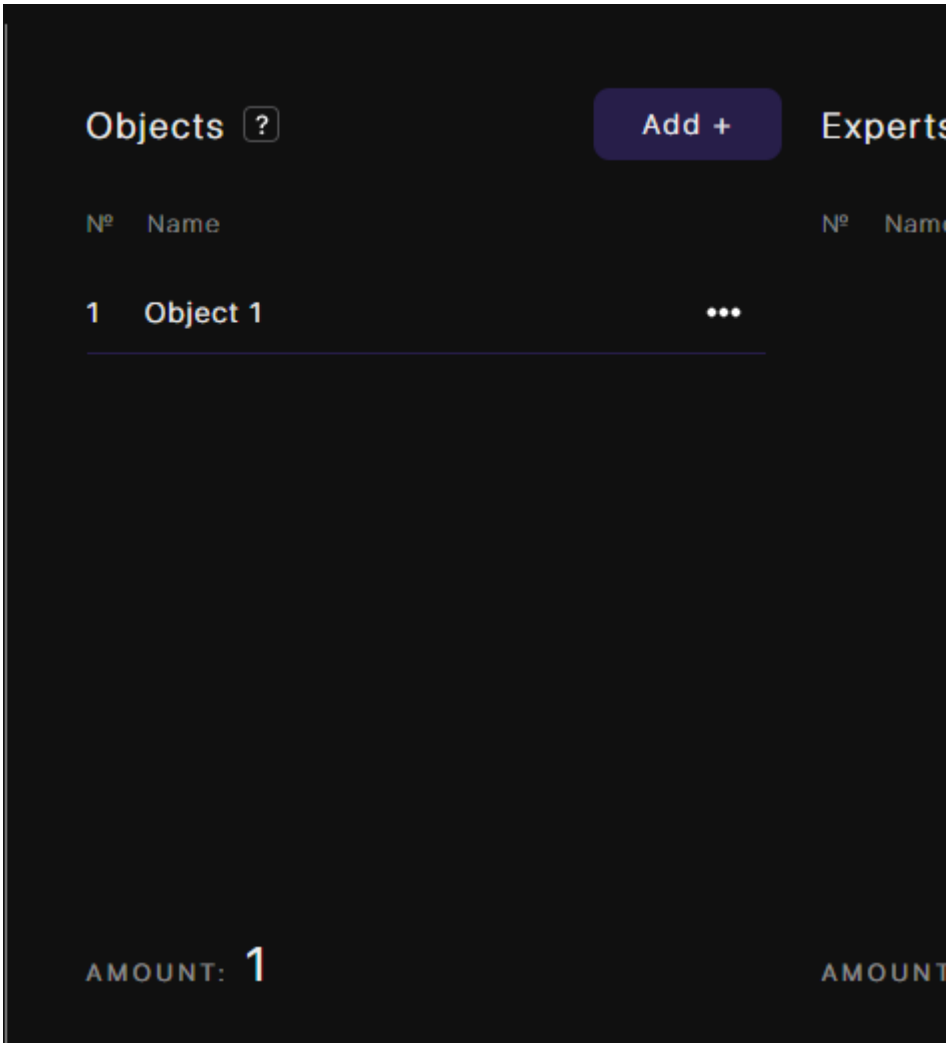


Рисунок 3.6 – новостворений об’єкт з’явився у відповідному компоненті, кількість знизу показу = 1

Заповнюємо голосування об’єктами і переходимо до кроку додавання експертів голосування, натискаємо кнопку “Add +” у компоненті експертів та з’являється відповідне вікно(рис 3.7)

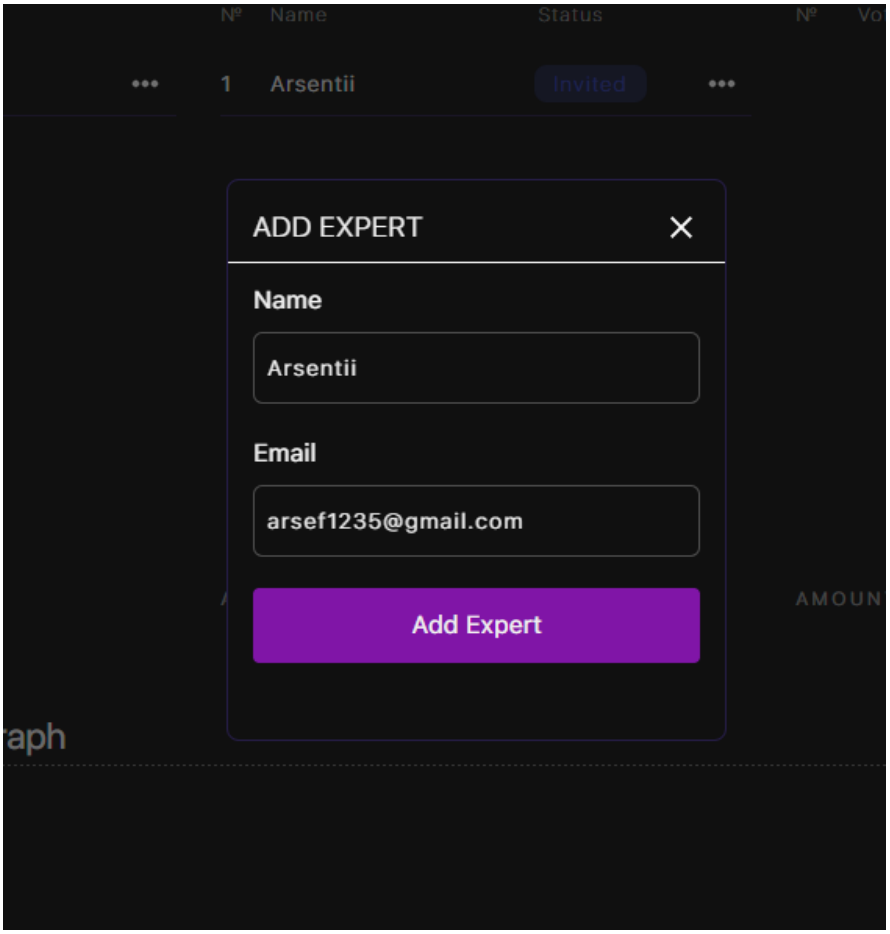


Рисунок 3.7 – вікно додавання експертів до голосування

Усі дані, які вводяться у форму валідуються спочатку на клієнті, потім на сервері, у разі введення неправильних даних – отримуємо повідомлення про помилку(рис 3.8).

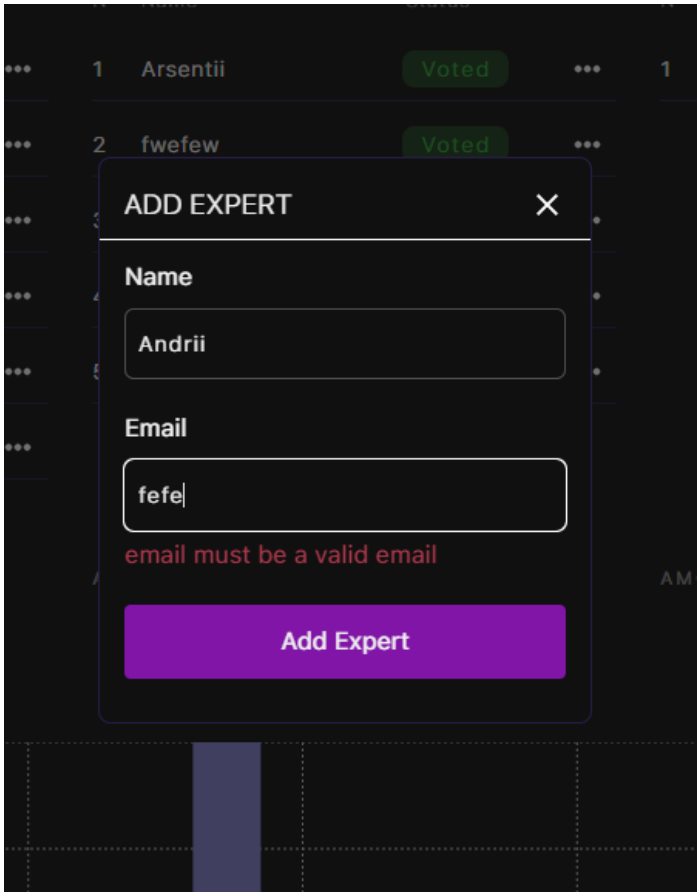


Рисунок 3.8 – вікно додавання експертів із неправильно введеними даними

Після додавання експерту, його статус буде «Invited», що означає, що експерт запрошений до голосування.(рис 3.9)

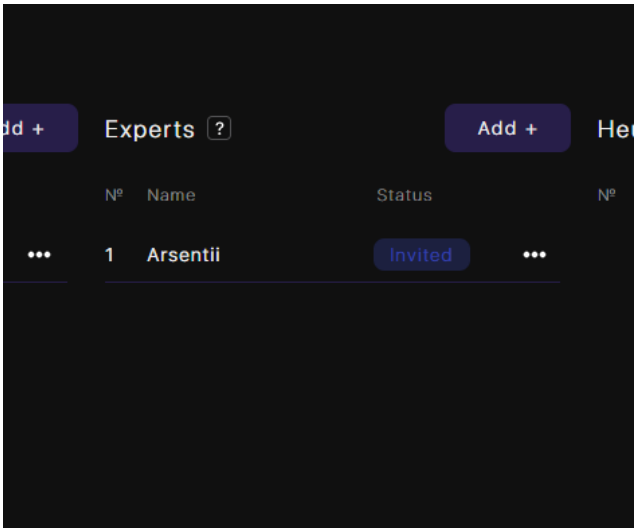


Рисунок 3.9 – доданий експерт із статусом «Invited»

Далі запрошеному експерту має прийти лист на вказану пошту із запрошенням перейти за посиланням та проголосувати(рис 3.10)

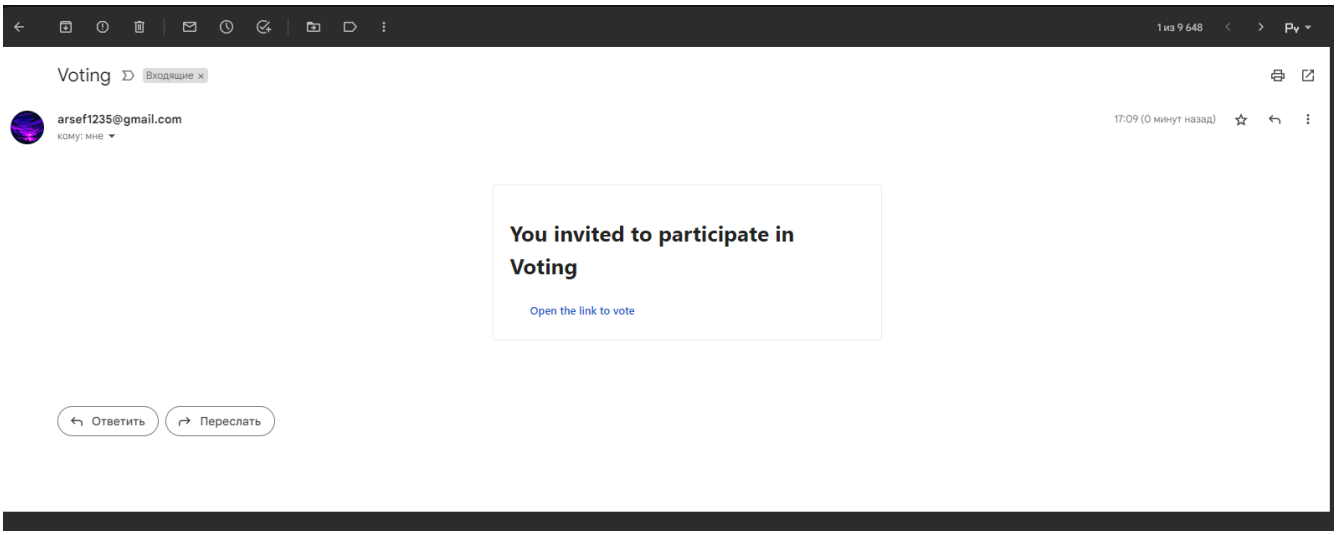


Рисунок 3.10 – лист, що приходить на пошту експерту

Перейшовши за посиланням, експерт переходить на сторінку для голосування, у посилання ми передаємо айді нашого голосування, та айді нашого експерта (рис 3.11)

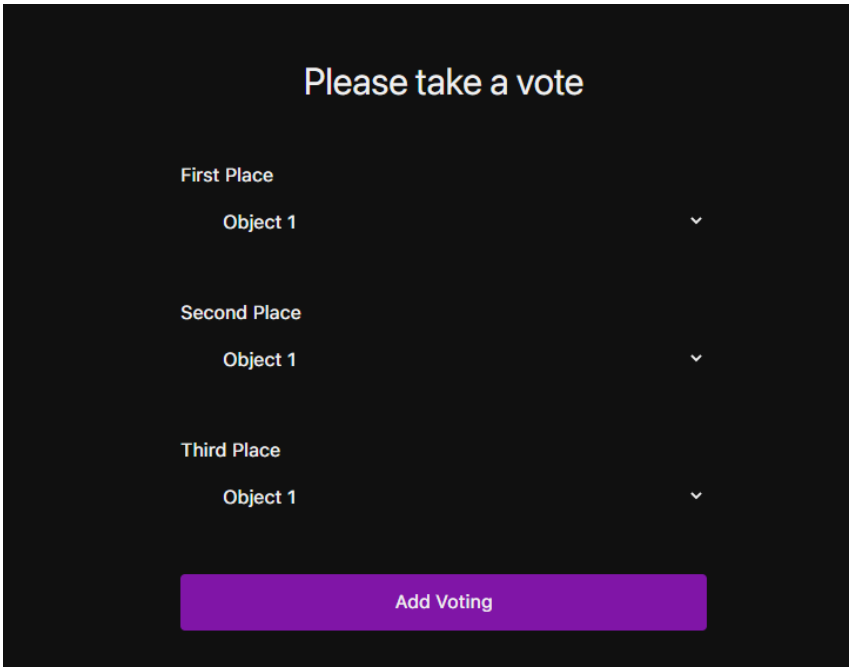


Рисунок 3.11 – сторінка із голосуванням

Експерт може розташувати об’єкти за пріоритетними місцям, поставити щось на перше, друге та третє місце, однакові об’єкти поставити не можна, на клієнті

відбувається перевірка. Після обрання об’єктів, експерт натискає кнопку “Add Voting” та завершує голосування, з’являється текст “Thanks for your vote” (рис 3.12), зникають параметри, що передавались разом із посиланням, щоб уникнути повторного голосування від того самого експерта, також відбувається перевірка на сервері, чи було отримано голосування від експерта чи ні.

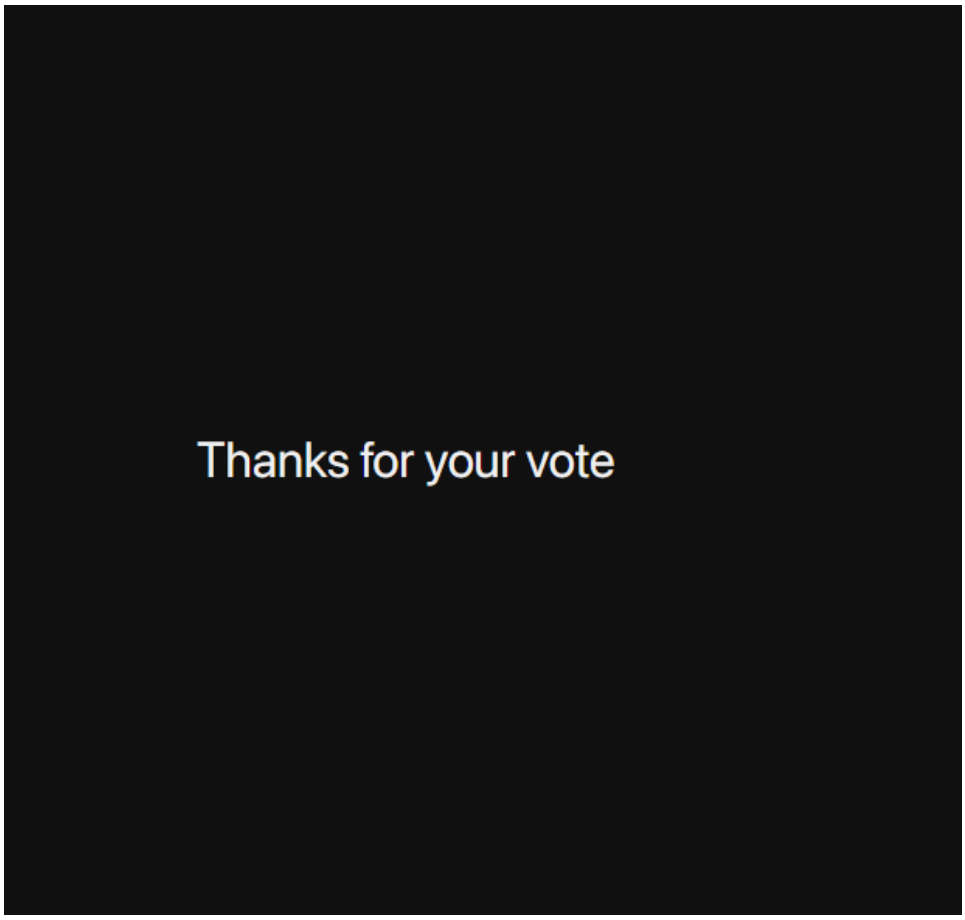


Рисунок 3.12 – сторінка голосування після відправки форми голосування

Відразу після голосування експерта, ми заносимо відповідні голоси у базу даних та надсилаємо оновлені дані на клієнт, тепер ми можемо бачити розподілення голосів експерта на графіку голосування (рис 3.13)

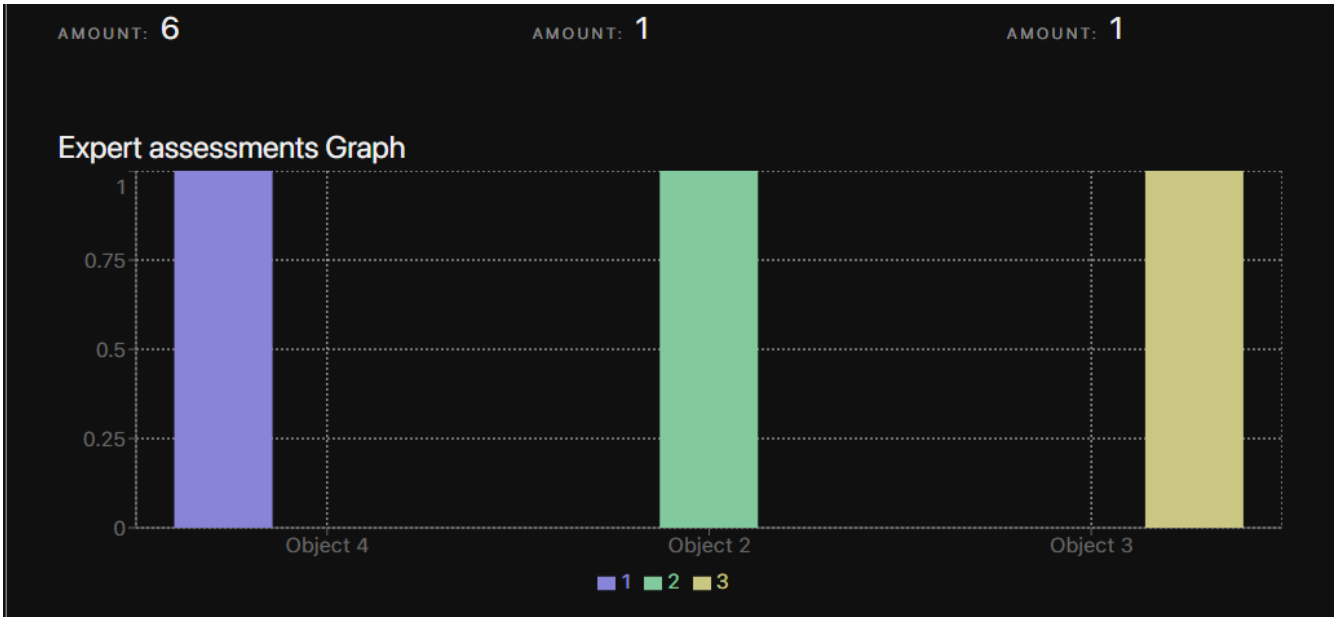


Рис 3.13 – головний інтерфейс програмного модуля. Компонент графіку

Статус експерта стає “Voted”, оскільки він проголосував (рис 3.14)

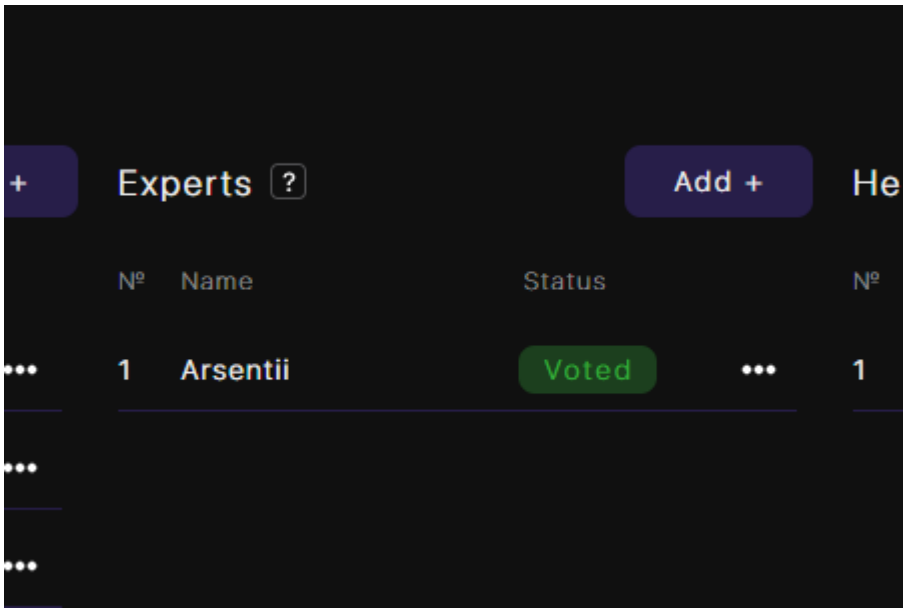


Рисунок 3.14 – компонент експертів, статус експерта змінено на “Voted”

Додаємо більше експертів, очікуємо на їх голосування, після голосування, можемо побачити наступний графік розподілених голосів (рис 3.15)

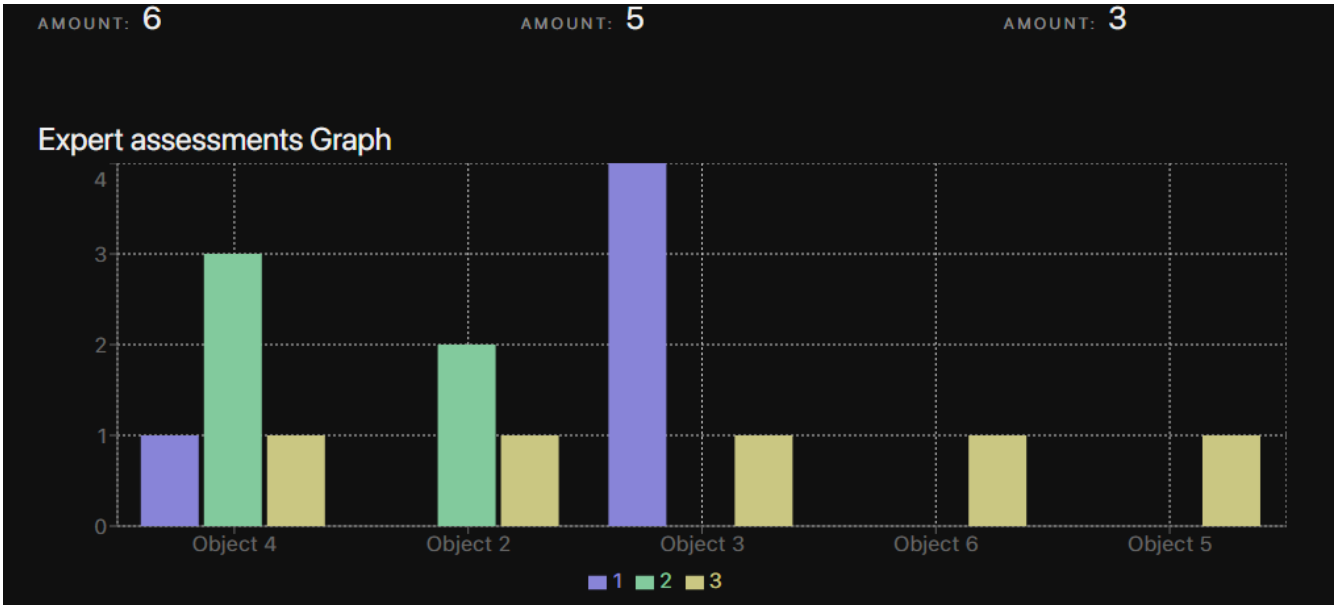


Рисунок 3.15 – головний інтерфейс додатку. Графік розподілення голосів

При наведенні на будь-який елемент графіку можемо побачити детальний опис (рис 3.16)

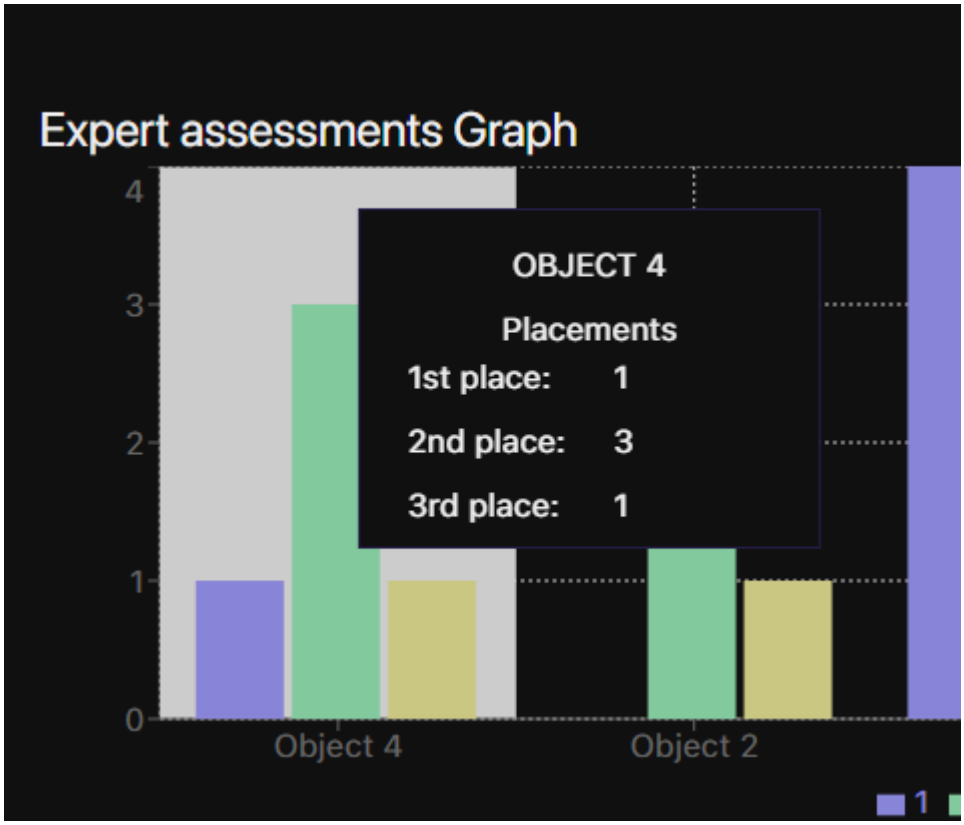


Рисунок 3.16 – вікно із детальною інформацією по голосам для даного об’єкта

Далі додаємо евристики для відбору підмножини переможців, щоб вилучити із голосування очевидних «лузерів» голосування, наприклад об’єкти, що отримали лише 2 голоси за 3 місце, або об’єкти, що отримали 0 голосів. Вікно додавання евристик виглядає наступним чином (3.17)

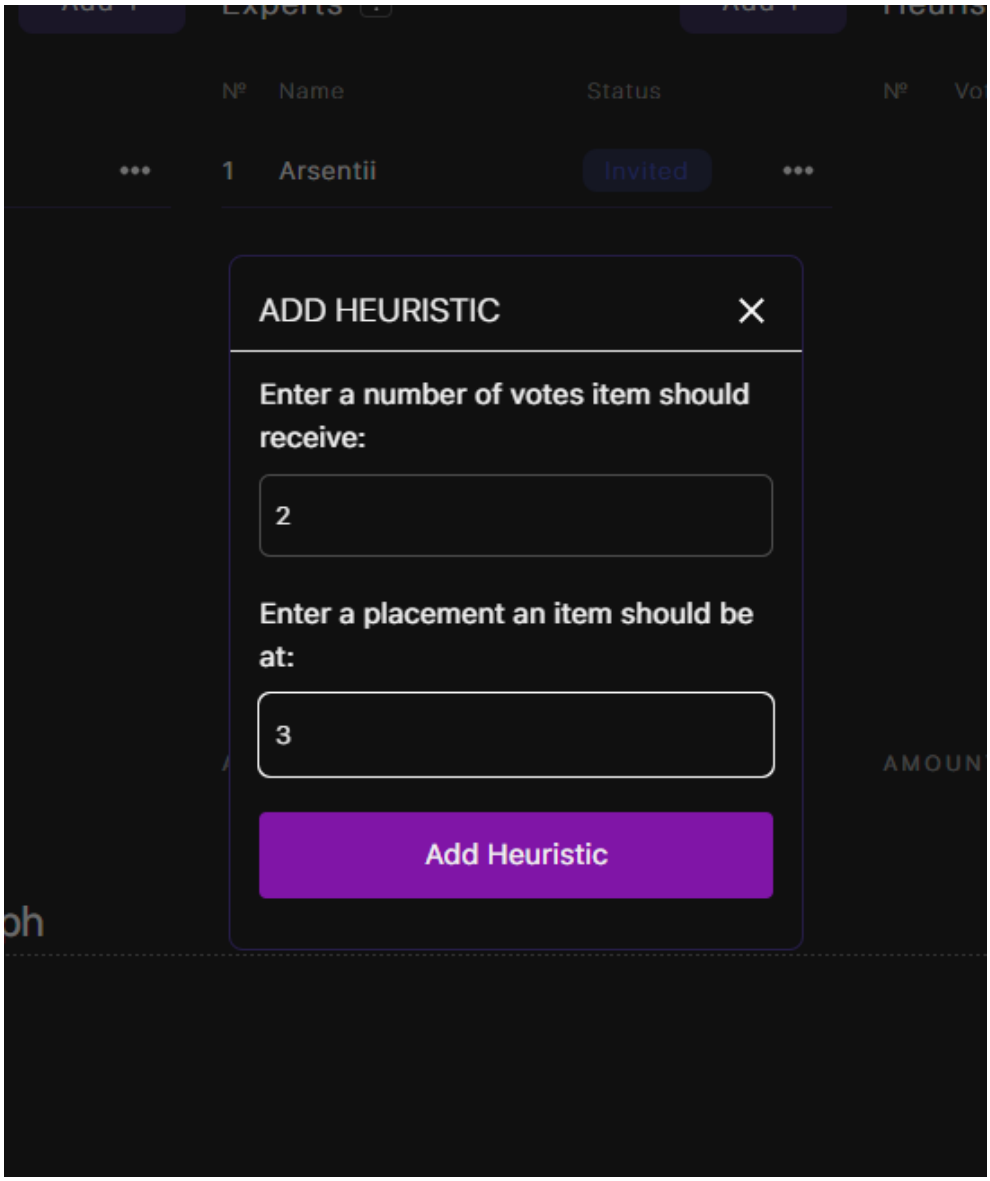


Рисунок 3.17 – вікно додавання евристик для відбору підмножини переможців

Після відправки форми евристика додається до голосування та відображається на інтерфейсі користувача. (рис 3.18).

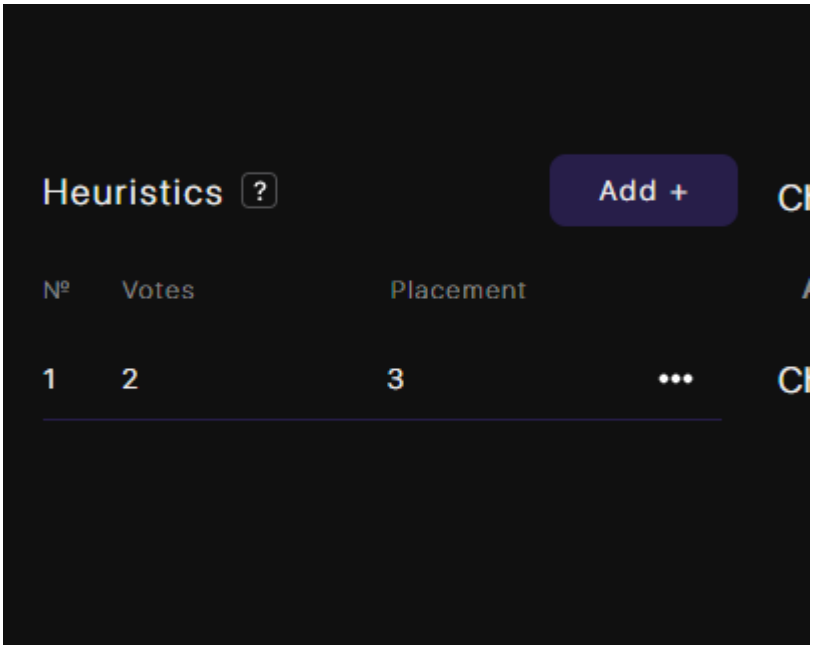


Рисунок 3.18 – головний інтерфейс користувача. Компонент евристик із доданою евристикою

Далі користувач обирає критерій за яким хоче отримати розв’язок та метод обробки експертної інформації.Після обрання користувач натискає кнопку “Get Results” для отримання результуючою ранжування переможців для нашого попереднього голосування. (рис 3.19).

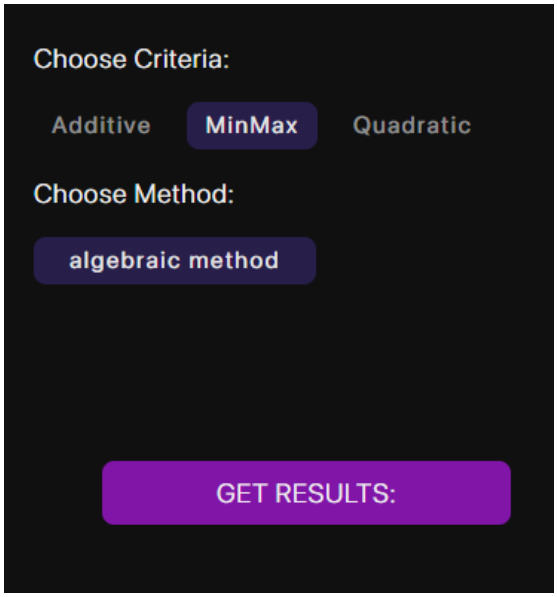


Рисунок 3.19 – головний інтерфейс додатку. Обрані критерії та метод розв’язку

Далі відбувається запит на сервер із обраними критеріями і методами, створюються відповідні класи та відбувається алгебраїчний метод визначення ранжування переможців із застосування вибраного критерію, в даному випадку критерій MinMax. Отримуємо результати у відповідному компоненті. (рис 3.20).

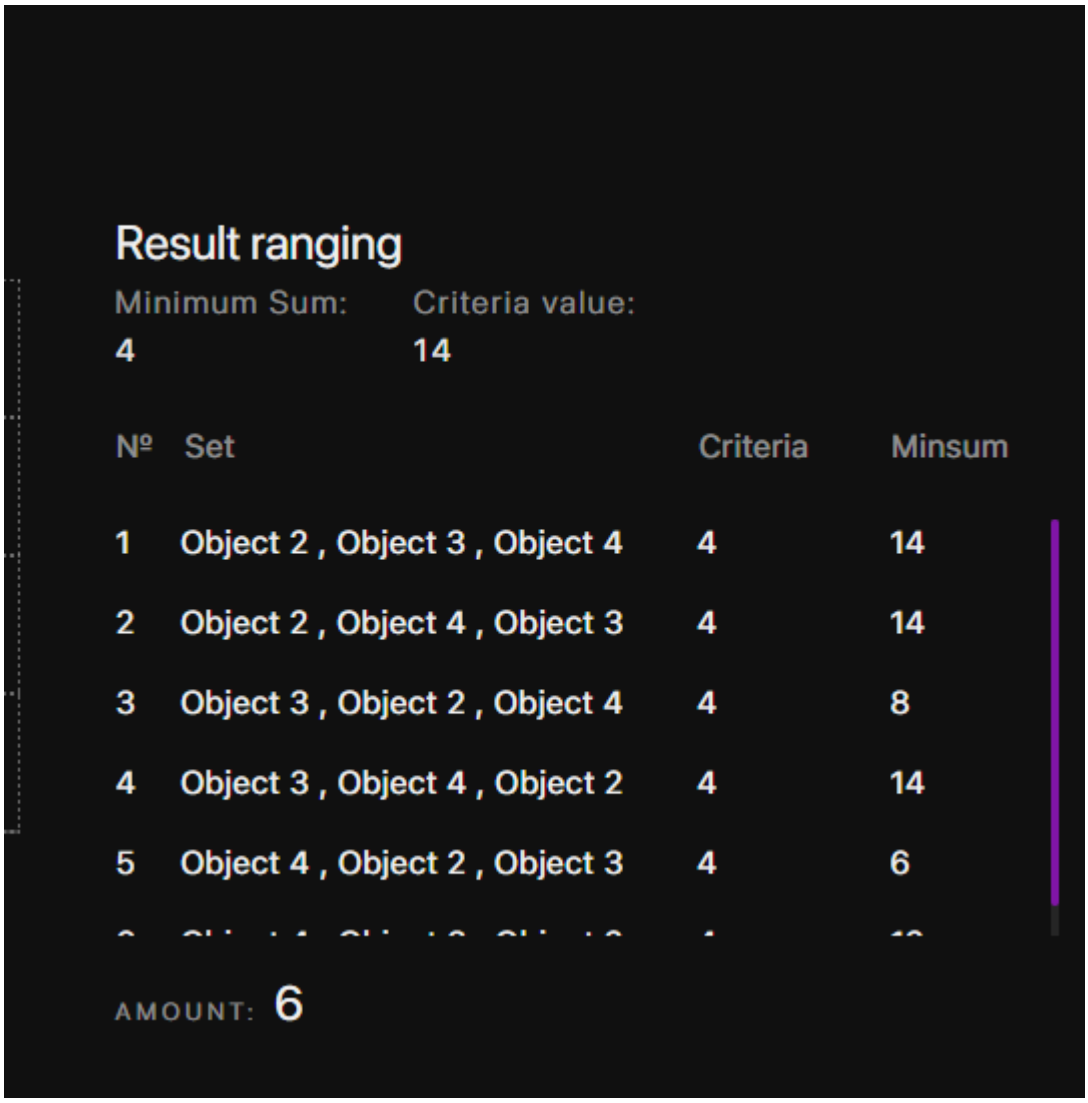


Рисунок 3.20 – головний інтерфейс застосунку. Результат роботи програми

У даному випадку, ми отримали 6 ранжувань об’єктів, кожен з розв’язків може бути кінцевим варіантом голосування, тобто можливо обрати перші 3 об’єкти з кожного ранжування і вони будуть переможцями голосування.

Мета попереднього голосування в отриманні потенційних переможців перед проведенням справжнього голосування, побачити лідерів голосування, які будуть математично пораховані, щоб результати були об’єктивними та чесними.

3.4 Тестування програмного модуля

Тестування програмного модуля є невід'ємною частиною процесу розробки програмного забезпечення. Воно спрямоване на перевірку якості та функціональності модуля з метою забезпечення його правильної роботи та відповідності вимогам.

Тестування в процесі розробки програмного забезпечення має наступні загальні визначення:

- Перевірка коректності: Тестування дозволяє перевірити, чи працює програмний модуль так, як очікується, та чи поводиться він відповідно до специфікацій та вимог.
- Виявлення помилок: Тестування допомагає виявити помилки, дефекти та неполадки в програмному модулі. Це дозволяє розробникам вносити необхідні корективи та покращувати якість продукту.
- Забезпечення стабільності: Тестування спрямоване на забезпечення стабільності та надійності програмного модуля. Воно допомагає виявити потенційні проблеми та забезпечити безперебійну роботу системи під час її експлуатації.
- Підтвердження вимог: Тестування дозволяє перевірити, чи відповідає програмний модуль вимогам та очікуванням замовника чи користувача. Це гарантує, що функціонал програмного модуля відповідає поставленим завданням.
- Вдосконалення якості: Тестування сприяє вдосконаленню якості програмного модуля. Шляхом ітеративного виконання тестів та виправлення виявлених проблем, розробники можуть підвищити якість коду, підсилюючи надійність та ефективність модуля.

Усі критичні функції програмного модуля були покриті тестами, щоб перевірити правильність роботи програми. Були протестовані усі класи, їх геттери та сеттери,

також було протестовано допоміжних функцій, їх було ізольовано та простетовано на тестових даних.(додаток А)

Описані тести перевіряють наступні функціональності модуля Vote:

- "should add expert correctly": Цей тест перевіряє, чи правильно додається експерт до модуля Vote. Він створює новий експерт і додає його до модуля Vote, а потім перевіряє, чи містить список Experts нового експерта.
- "should return id of vote correctly": Цей тест перевіряє, чи повертається правильно ідентифікатор голосу експерта. Він створює нового експерта з певним ідентифікатором і перевіряє, чи повертається цей ідентифікатор.
- "should add object correctly": Цей тест перевіряє, чи правильно додається об'єкт до модуля Vote. Він створює новий об'єкт і додає його до модуля Vote, а потім перевіряє, чи містить список Objects новий об'єкт.
- "should add heuristic correctly": Цей тест перевіряє, чи правильно додається евристика до модуля Vote. Він створює нову евристику і додає її до модуля Vote, а потім перевіряє, чи містить список Heuristics нову евристику.
- "should set assessments correctly": Цей тест перевіряє, чи правильно встановлюються оцінки (assessments) у модулі Vote. Він створює новий Map оцінок, встановлює його у модуль Vote і перевіряє, чи відповідає список Assessments встановленому Map оцінок.
- "should set object placements correctly": Цей тест перевіряє, чи правильно встановлюються розташування об'єктів (object placements) у модулі Vote. Він створює список голосів (votes) для об'єктів, встановлює цей список у модуль Vote і перевіряє, чи відповідає список Votes встановленому списку голосів.
- "should filter objects based on votes": Цей тест перевіряє, чи правильно фільтруються об'єкти на основі голосів у модулі Vote. Він створює кілька об'єктів і списків голосів для них, додає їх до модуля Vote, встановлює розташування об'єктів на основі голосів і перевіряє, чи відповідає список Objects відфільтрованим об'єктам.

- "should apply heuristics correctly": Цей тест перевіряє, чи правильно застосовуються евристики у модулі Vote. Він створює кілька об'єктів, списки голосів для них та евристики, додає їх до модуля Vote, встановлює розташування об'єктів на основі голосів, додає евристики і застосовує їх, а потім перевіряє, чи відповідає список Votes застосованим евристикам.

Ці тести перевіряють різні функціональності модуля Vote, такі як додавання експертів, об'єктів, евристик, встановлення оцінок, розташування об'єктів, фільтрація об'єктів та застосування евристик. Вони допомагають переконатись, що відповідні методи модуля Vote працюють правильно та відповідають очікуванням.

Також необхідним є тестування dataSlice компоненту, оскільки він відповідає за маніпулювання даними для відображення коректних даних у інтерфейсі користувача. Оскільки робити кожен раз запит на сервер та до бази даних це не є ефективним.

Нижче наведено описи тестів для модуля dataSlice:

- "should set votings correctly": Цей тест перевіряє, чи правильно встановлюються дані голосувань (votings). Він створює список голосувань, викликає дію setVotings редуктора dataSlice з цим списком і перевіряє, чи відповідає поле votings у стані наступному списку голосувань.
- "should set isLoading correctly": Цей тест перевіряє, чи правильно встановлюється прапорець isLoading. Він викликає дію setIsLoading редуктора dataSlice зі значенням true і перевіряє, чи відповідає поле isLoading у стані значенню true.
- "should add an expert correctly": Цей тест перевіряє, чи правильно додається експерт до голосування. Він створює об'єкт експерта, викликає дію addExpert редуктора dataSlice з цим експертом та ідентифікатором голосування і перевіряє, чи містить список експертів відповідного голосування нового експерта.

- "should add an object correctly": Цей тест перевіряє, чи правильно додається об'єкт до голосування. Він створює об'єкт об'єкта, викликає дію addObject редуктора dataSlice з цим об'єктом та ідентифікатором голосування і перевіряє, чи містить список об'єктів відповідного голосування новий об'єкт.
- "should remove an object correctly": Цей тест перевіряє, чи правильно видаляється об'єкт з голосування. Він викликає дію removeObject редуктора dataSlice з ідентифікатором об'єкта та ідентифікатором голосування і перевіряє, чи не містить список об'єктів відповідного голосування видаленого об'єкта.
- "should edit an object correctly": Цей тест перевіряє, чи правильно редагується об'єкт в голосуванні. Він створює оновлений об'єкт, викликає дію editObject редуктора dataSlice з цим оновленим об'єктом, ідентифікатором об'єкта та ідентифікатором голосування і перевіряє, чи містить список об'єктів відповідного голосування оновлений об'єкт.
- "should edit an expert correctly": Цей тест перевіряє, чи правильно редагується експерт в голосуванні. Він створює оновленого експерта, викликає дію editExpert редуктора dataSlice з цим оновленим експертом, ідентифікатором експерта та ідентифікатором голосування і перевіряє, чи містить список експертів відповідного голосування оновленого експерта.
- "should remove an expert correctly": Цей тест перевіряє, чи правильно видаляється експерт з голосування. Він викликає дію removeExpert редуктора dataSlice з ідентифікатором експерта та ідентифікатором голосування і перевіряє, чи не містить список експертів відповідного голосування видаленого експерта.
- "should add a voting correctly": Цей тест перевіряє, чи правильно додається голосування. Він створює об'єкт голосування, викликає дію addVoting редуктора dataSlice з цим голосуванням і перевіряє, чи містить список голосувань нове голосування.

- "should add a heuristic correctly": Цей тест перевіряє, чи правильно додається евристика до голосування. Він створює об'єкт евристики, викликає дію addHeuristic редуктора dataSlice з цією евристикою та ідентифікатором голосування і перевіряє, чи містить список евристик відповідного голосування нову евристику.
- "should edit a heuristic correctly": Цей тест перевіряє, чи правильно редагується евристика в голосуванні. Він створює оновлену евристику, викликає дію editHeuristic редуктора dataSlice з цією оновленою евристикою, ідентифікатором евристики та ідентифікатором голосування і перевіряє, чи містить список евристик відповідного голосування оновлену евристику.
- "should remove a heuristic correctly": Цей тест перевіряє, чи правильно видаляється евристика з голосування. Він викликає дію removeHeuristic редуктора dataSlice з ідентифікатором евристики та ідентифікатором голосування і перевіряє, чи не містить список евристик відповідного голосування видаленої евристики.
- "should set results correctly": Цей тест перевіряє, чи правильно встановлюються результати голосування. Він створює список результатів, викликає дію setResults редуктора dataSlice з цим списком і перевіряє, чи відповідає поле results у стані наступному списку результатів.
- "should clear results correctly": Цей тест перевіряє, чи правильно очищаються результати голосування. Він викликає дію clearResults редуктора dataSlice і перевіряє, чи поле results у стані є null.

Далі опишемо допоміжні функції модулю helpers.ts, дані тести перевіряють коректність розрахунків на основі тестових даних:

- transformAssessments - Тест перевіряє, чи правильно трансформується мапа оцінок assessments в матрицю оцінок transformedAssessments для об'єктів

objects. Перевіряється правильність відображення оцінок експертів на об'єкти.

- generateCombinations - Тест перевіряє, чи правильно генеруються всі можливі комбінації об'єктів зі списку objects. Перевіряється, що всі комбінації генеруються вірно.
- calculateMatrix - Тест перевіряє, чи правильно обчислюється матриця, отримана шляхом множення матриць matrixA і matrixB. Перевіряється, що результат обчислення матриці є правильним.
- findMinimaxValue - Тест перевіряє, чи правильно знаходиться мінімаксне значення у масиві array. Перевіряється, що мінімаксне значення і відповідний рядок масиву знаходяться правильно.
- findQuadraticCriterionValue - Тест перевіряє, чи правильно знаходиться значення квадратичного критерію у масиві array. Перевіряється, що значення квадратичного критерію і відповідний рядок масиву знаходяться правильно.
- findAdditiveCriterionValue - Тест перевіряє, чи правильно знаходиться значення адитивного критерію у масиві array. Перевіряється, що значення адитивного критерію і відповідний рядок масиву знаходяться правильно.

Ці тести допомагають переконатися, що функції модуля helpers.ts працюють правильно і повертають очікувані результати для заданих вхідних даних.

3.5 Висновки до третього розділу

В розділі "Програмна реалізація. Тестові приклади" виконано детальний опис реалізації програмного модуля підтримки голосування шляхом вибору упорядкованої підмножини переможців. Для створення цього модуля були використані сучасні технології розробки, такі як React, Typescript, Next.js, а також Nodemailer для надсилання електронних листів та MongoDB для зберігання даних.

Докладно розглянуто компоненти, класи та функції програмного модуля, а також їх взаємозв'язки і значення для загального функціонування додатку. В даному контексті, модуль розроблено з орієнтацією на користувача, таким чином, він взаємодіє з користувачем для забезпечення ефективного процесу голосування.

Згідно з мінімаксним, квадратичним та адитивним критеріями розв'язку, розрахунки проведені з максимальною точністю. Розроблений програмний модуль використовує алгебраїчний метод для обрахунку результатів голосування, що дозволяє гарантувати точність і надійність результатів. Тестування модуля було виконано за допомогою бібліотеки Jest, що дозволило перекрити всі критичні функції та класи, а також перевірити правильність розрахунків. Таким чином, було забезпечено відсутність помилок і неточностей у роботі програми. Застосування Jest допомогло провести комплексне тестування як бізнес-логіки, так і клієнтських компонентів.

У результаті проведеної роботи було створено надійний та ефективний програмний модуль підтримки процедури преференційного голосування, що відповідає поставленим задачам та вимогам.

ВИСНОВКИ

В ході виконання випускної кваліфікаційної роботи на тему "Програмний модуль підтримки процедури преференційного голосування" було досліджено проблему підтримки прийняття рішень на основі алгебраїчного методу обробки експертних даних, розглянуто сучасні програмні аналоги та проведено аналіз їх функціональності.

На основі проведеного аналізу було визначено необхідність розробки оригінального програмного модуля, орієнтованого на користувача, для підтримки прийняття рішень. За результатами роботи було розроблено структуру програмного модуля, діаграми Event-driven process chain, SwimLane та IDEF0 для визначення основних процесів предметного середовища. Було використано технології React, Typescript, Next.js для реалізації модуля, а також MongoDB для зберігання даних. У процесі розробки було приділено особливу увагу створенню користувацького інтерфейсу та реалізації функцій модуля для автоматизації процесу голосування з експертним оцінюванням. Реалізовано детальний опис реалізації програмного модуля, включаючи опис компонентів, класів та функцій, їх взаємозв'язків та значення для загального функціонування додатку. Розроблений програмний модуль використовує алгебраїчний метод для обрахунку результатів голосування, що гарантує точність і надійність результатів. Система тестування, реалізована за допомогою Jest, забезпечила перевірку всіх критичних функцій та класів, а також перевірку правильності розрахунків. Результати тестування демонструють високу стабільність та ефективність розробленого програмного модуля.

Новизна роботи полягає в інтеграції системи голосування на основі алгебраїчного методу обробки експертних даних з урахуванням критеріїв мінімаксного, квадратичного та адитивного розв'язку. Отриманий програмний модуль підтримки процедури преференційного голосування відповідає всім вимогам та задачам, поставленим на початку виконання роботи, є високоякісним та

готовим до впровадження. Робота має великий потенціал для використання в різних сферах, де необхідне прийняття важливих рішень на основі експертних оцінок.

У рамках роботи було здійснено детальне дослідження сучасних технологій розробки та їх можливого застосування для реалізації розробленого програмного модуля. Проведено аналіз поточного стану існуючих рішень, на основі якого було визначено ключові вимоги до модуля. Було показано, що розроблений модуль може бути використаний в широкому спектрі сфер, включаючи політичні вибори, корпоративне рішення, соціальні дослідження та інші галузі, де потрібно враховувати експертні оцінки при прийнятті рішень.

Технічно-економічна ефективність розробленого модуля є високою, враховуючи його низькі вимоги до ресурсів, швидкість обробки даних і високий рівень надійності. Розроблений модуль відповідає сучасним технологічним стандартам, має інтуїтивно зрозумілий користувацький інтерфейс і забезпечує максимальну точність обчислень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Hrygorii Hnatiienko, Snezhana Gamotska, Oksana Vlasenko, Ravhshanbek Zulunov - The problem of constructing the ranking of objects on the base of multiple comparisons for the cook metric.

[2] Vitaliy V.Tsyganok – Providing sufficient strict individual rankings' consistency level while group decision-makin with feedback.

[3] О.Ф. Волошин, С.О. Мащенко – Моделі та методи прийняття рішень. 2018р. – 72-94с.

[4] James N. Webb – "Game theory: Decisions, Interaction and Evolution" – 2003p.

[5] David E. Bell, Howard Raiffa, Amos Tversky – "Decision Making: Descriptive, Normative, and Prescriptive Interactions"

[6] Giarratano and Riley – "Expert Systems: Principles and Programming"

[7] Giovanni Parmigiani and Lurdes Inoue - "Decision Theory: Principles and Approaches"

[8] David A. Schilling, V. K. Chaturvedi, and Matthew J. Drake - "Decision Making: Process, Modeling and Support"

[9] Myra Spiliopoulou, Lars Schmidt-Thieme, and Ruth Janning - "Data Analysis, Machine Learning and Knowledge Discovery"

[10] Yacov Y. Haimes, Laszlo T. Gomory, and Thomas L. Saaty - "Multiobjective Decision Making: Policies and Methods"

[11] Mykel J. Kochenderfer, Tim A. Wheeler and Kyle H. Wray, "Algorithms for Decision Making

[12] React documentation[Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/learn

[13] Next documentation[Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/docs

[14] MongoDB documentation[Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/docs/

[15] React charts[Електронний ресурс] – Режим доступу до ресурсу: https://recharts.org/en-US/guide

[16] Nodemailer[Електронний ресурс] – Режим доступу до ресурсу: https://nodemailer.com/usage/

[17] Typescript documentation[Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/handbook/2/objects.html

[18] Node js[Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/docs

ДОДАТОК А. ПРОГРАМНИЙ КОД ТЕСТІВ ОСНОВНИХ МОДУЛІВ

Vote.test.ts

```
import { IVote } from '@/Types/Interfaces';
import { Expert, Heuristic, Vote, VotingObject } from '../Voting';

describe('Vote', () => {
  let vote: Vote;

  beforeEach(() => {
    vote = new Vote('title', 'criteria', 'method');
  });

  it('should add expert correctly', () => {
    const expert = new Expert('1', 'John Doe', 'john.doe@example.com', 'status');
    vote.addExpert(expert);
    expect(vote.Experts).toContain(expert);
  });

  it('should return id of vote correctly', () => {
    const expert = new Expert('1', 'John Doe', 'john.doe@example.com', 'status');

    expect(expert.id).toEqual('1');
  });

  it('should add object correctly', () => {
    const votingObject = new VotingObject('1', 'Object 1', 'Description 1');
    vote.addObject(votingObject);
    expect(vote.Objects).toContain(votingObject);
  });

  it('should add heuristic correctly', () => {
    const heuristic = new Heuristic('1', 10, 1);
    vote.addHeuristic(heuristic);
    expect(vote.Heuristics).toContain(heuristic);
  });

  it('should set assessments correctly', () => {
    const assessments = new Map();
    assessments.set('key1', ['assessment1', 'assessment2']);
    vote.setAssessments(assessments);
    expect(vote.Assessments).toEqual(assessments);
  });

  it('should set object placements correctly', () => {
    const votes: IVote[] = [
      { _id: '1', name: 'Object 1', scores: [1, 2, 3] },
      { _id: '2', name: 'Object 2', scores: [3, 2, 1] },
```

```
      ];
      vote.setObjectPlacements(votes);
      expect(vote.Votes).toEqual(votes);
    });

    it('should filter objects based on votes', () => {
      const votingObject1 = new VotingObject('1', 'Object 1', 'Description 1');
      const votingObject2 = new VotingObject('2', 'Object 2', 'Description 2');
      const votingObject3 = new VotingObject('3', 'Object 3', 'Description 3');
      const votes: IVote[] = [
        { _id: '1', name: 'Object 1', scores: [1, 2, 3] },
        { _id: '2', name: 'Object 2', scores: [3, 2, 1] },
      ];
      vote.addObject(votingObject1);
      vote.addObject(votingObject2);
      vote.addObject(votingObject3);
      vote.setObjectPlacements(votes);
      vote.filterObjects();
      expect(vote.Objects).toEqual([votingObject1, votingObject2]);
    });

    it('should apply heuristics correctly', () => {
      const votingObject1 = new VotingObject('1', 'Object 1', 'Description 1');
      const votingObject2 = new VotingObject('2', 'Object 2', 'Description 2');
      const votingObject3 = new VotingObject('3', 'Object 3', 'Description 3');
      const votes: IVote[] = [
        { _id: '1', name: 'Object 1', scores: [1, 2, 3] },
        { _id: '2', name: 'Object 2', scores: [0, 2, 0] },
        { _id: '3', name: 'Object 3', scores: [0, 0, 1] },
      ];
      const heuristic = new Heuristic('1', 1, 3);
      const heuristic2 = new Heuristic('2', 2, 2);
      vote.addObject(votingObject1);
      vote.addObject(votingObject2);
      vote.addObject(votingObject3);
      vote.setObjectPlacements(votes);
      vote.addHeuristic(heuristic);
      vote.addHeuristic(heuristic2);
      vote.applyHeuristics();
      expect(vote.Votes).toEqual([{ _id: '1', name: 'Object 1', scores: [1, 2, 3] }]);
    });
  });
```

Helpers.test.ts

```
import {
    transformAssessments,
    generateCombinations,
    calculateMatrix,
    findMinimaxValue,
    findQuadraticCriterionValue,
```

```
  findAdditiveCriterionValue,
} from '../helpers';
import { VotingObject } from '../Voting';

describe('Helpers', () => {
  describe('transformAssessments', () => {
    it('should transform assessments correctly', () => {
      const assessments = new Map();
      assessments.set('expert1', ['Title 1', 'Title 2', 'Title 4']);
      assessments.set('expert2', ['Title 3', 'Title 4', 'Title 2']);
      const objects = [
        new VotingObject('object1', 'Title 1', 'Description 1'),
        new VotingObject('object2', 'Title 2', 'Description 2'),
        new VotingObject('object3', 'Title 3', 'Description 3'),
        new VotingObject('object4', 'Title 4', 'Description 4'),
      ];

      const transformedAssessments = transformAssessments(assessments, objects);

      expect(transformedAssessments).toEqual([
        [1, 0],
        [2, 3],
        [0, 1],
        [3, 2],
      ]);
    });
  });

  describe('generateCombinations', () => {
    it('should generate combinations correctly', () => {
      const objects = [new VotingObject('1', 'Object 1', 'Description 1'), new VotingObject('2', 'Object
2', 'Description 2')];

      const combinations = generateCombinations(objects);

      expect(combinations).toEqual([[1, 2], [2, 1]]);
    });
  });

  describe('calculateMatrix', () => {
    it('should calculate matrix correctly', () => {
      const matrixA = [[1, 2], [3, 4], [5, 6]];
      const matrixB = [[1, 2], [3, 4]];
      const expectedMatrix = [[1, 3], [3, 1], [7, 5]];

      const resultMatrix = calculateMatrix(matrixA, matrixB);

      expect(resultMatrix).toEqual(expectedMatrix);
    });
```

```
    it('should return undefined if number of columns in A is not equal to the number of rows in B', () => {
      const matrixA = [[1, 2,4], [3, 4,5]];
      const matrixB = [[1, 2, 3], [4, 5, 6]];

      const resultMatrix = calculateMatrix(matrixA, matrixB);

      expect(resultMatrix).toBeUndefined();
    });
  });

  describe('findMinimaxValue', () => {
    it('should find minimax value correctly', () => {
      const array = [[5, 3, 7], [2, 9, 1], [4, 6, 8]];
      const expectedValue = {
        minMax: 7,
        minRowIndex: [[0, 15, 7]],
      };
      const result = findMinimaxValue(array);
      expect(result).toEqual(expectedValue);
    });
  });
  describe('findQuadraticCriterionValue', () => {
    it('should find quadratic criterion value correctly', () => {
      const array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
      const expectedValue = {
        minQuadraticSum: 14,
        minRowIndex: [[0, 6, 14]],
      };
      const result = findQuadraticCriterionValue(array);
      expect(result).toEqual(expectedValue);
    });
  });
  describe('findAdditiveCriterionValue', () => {
    it('should find additive criterion value correctly', () => {
      const array = [[5, 3, 7], [2, 9, 1], [4, 6, 8]];
      const expectedValue = {
        minSum: 12,
        minRowIndex: [[1, 9, 12]],
      };

      const result = findAdditiveCriterionValue(array);
      expect(result).toEqual(expectedValue);
    });
  });
});
```

ДОДАТОК Б. ПРОГРАМНИЙ КОД ОСНОВНИХ МОДУЛІВ

```
import {  IObject, IResult, IVote, IVoting } from "@/Types/Interfaces";
import { generateCombinations, transformAssessments, calculateMatrix, findMinimaxValue,
findAdditiveCriterionValue, findQuadraticCriterionValue } from "./helpers";
import { TEXT_CONSTANTS } from "@/Constants/constants";

export class Expert {
    constructor(
        private _id: string,
        private name: string,
        private email: string,
        private status: string,
    ) {}

    get id(): string {
        return this._id
    }

    get description(): string {
        return `${this.name} (${this.email}): ${this.status}`;
    }
}

export class VotingObject {
    constructor(
        private _id: string,
        public title: string,
        public description: string
    ) {}

    get name(): string {
        return `${this.title} (${this.description})`;
    }

    get id(): string {
        return this._id
    }
}

export class Heuristic {
    constructor(
        private _id: string,
        public sum: number,
        public placing: number
    ) {}

    get name(): string {
```

```
        return `Exclude Object with sum votes: ${this.sum} and with placement taken: ${this.placing}`;
    }

    get id(): string {
        return this._id
    }
}

export class Vote {
    private objects: VotingObject[] = [];
    private experts: Expert[] = [];
    private heuristics: Heuristic[] = [];
    private results: IResult[] = []; // Update with the right type
    private assessments: Map<string, string[]> = new Map();
    private votes: IVote[] = []; // Update with the right type

    constructor(private title: string, private criteria: string, private method: string) {}


    get Results(): IResult[] {
        return this.results
    }

    get Objects(): VotingObject[] {
        return this.objects
    }

    get Heuristics(): Heuristic[] {
        return this.heuristics
    }

    get Experts(): Expert[] {
        return this.experts
    }

    get Assessments(): Map<string, string[]> {
        return this.assessments
    }

    get Votes(): IVote[] {
        return this.votes
    }

    addExpert(expert:Expert) {
        this.experts.push(expert);
    }

    addObject(object:VotingObject) {
        this.objects.push(object);
```

```
}

addHeuristic(heuristic: Heuristic) {
   this.heuristics.push(heuristic);
}

setAssessments(assessments: Map<string, string[]>) {
   this.assessments = assessments;
}

setObjectPlacements(votes: IVote[]) {
   this.votes = votes;
}

applyHeuristics() {
   let filteredVotes = [...this.votes];
   for (const heuristic of this.heuristics) {
      filteredVotes = filteredVotes.filter((vote) => {
         let sum = vote.scores.reduce((a,b) => a+b,0);
         if (heuristic.placing !== undefined) {
            return !(sum === heuristic.sum && vote.scores[heuristic.placing-1] === heuristic.sum)
         }
         else {
            return !(sum === heuristic.sum)
         }
      })
   }
   this.votes = filteredVotes;
}

filterObjects() {
   const filteredIds = this.votes.map(vote => vote.name);
   this.objects = this.objects.filter(object => filteredIds.includes(object.title));
}

vote() {
   try {
      let sum:any
      this.applyHeuristics()
      this.filterObjects()
      const matrixA = generateCombinations(this.objects);
      const matrixB = transformAssessments(this.assessments, this.objects)
      const resultMatrix = calculateMatrix(matrixA, matrixB);
      if (resultMatrix) {
         switch(this.criteria) {
            case TEXT_CONSTANTS.MinMax:
               sum = findMinimaxValue(resultMatrix);
               break;
            case TEXT_CONSTANTS.ADDITIVE:
```

```
                sum = findAdditiveCriterionValue(resultMatrix)
                break;
            case TEXT_CONSTANTS.QUADRATIC:
                sum = findQuadraticCriterionValue(resultMatrix)
                break;
        }
      }
      sum.minRowIndex.sort((a:[number[], number, number],b:[number[], number, number]) => a[2]
- b[2]);
      let result:IResult[] = [];
      for (let i = 0; i < sum.minRowIndex.length; i++) {
        result.push({
          set: matrixA[sum.minRowIndex[i][0]].map((num:number) => this.objects[num - 1].title),
          criteria: sum.minRowIndex[i][2],
          minSum: sum.minRowIndex[i][1]
        })
      }
      this.results = result
      return result
    } catch(err) {
      console.log('Error in vote()', err);
      throw new Error('Couldn`t complete a vote')
    }

  }
}
```