Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Онлайн система для вироблення спільних рішень групою людей»

Виконав: студент 2 курсу, групи 1ІСТ-24м спеціальності 126 – Інформаційні системи та технології
(шифр і назва спеціальності)

Олександр ЛЮБУНЬ
(ПІБ студента)

Керівник: к.т.н., доцент кафедри АІІТ
Владислав КАБАЧІЙ
(науковий ступінь, вчене звання / посада, ПІБ керівника)

«5» 12 2025 р.

Опонент: д.т.н., проф. каф. КСУ
В'ячеслав КОВТУН
(науковий ступінь, вчене звання / посада, ПІБ опонента)

«5» 12 2025 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)

«12» 12 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 126 – Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«16» 09 2025 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Любуню Олександру Ігоровичу
(ПІБ автора повністю)

1. Тема роботи: Онлайн система для вироблення спільних рішень групою людей.
   Керівник роботи: к.т.н., доцент каф. АІІТ Кабачій В. В.
   Затверджені наказом ВНТУ від « 24 » вересня 2025 року № 313.
2. Строк подання роботи студентом: до « 12 » грудня 2025 року.
3. Вихідні дані до роботи: Створення та управління сесіями голосувань. Підтримка різноманітних механізмів голосування (просте «За/Проти», бальна система, ранжування альтернатив, багатокритеріальна оцінка з діапазоном оцінок за замовчуванням від 0 до 10). Можливість запрошення учасників, налаштування прав доступу та обмежень за часом (задання часу початку і завершення голосування). Візуалізація результатів голосування у вигляді звіту. Модуль управління завданнями.
4. Зміст текстової частини: Вступ; Теоретичні основи та сучасний стан

дослідження колективного прийняття рішень; Аналіз доцільності розробки онлайн системи для вироблення спільних рішень групою людей; Розробка онлайн системи для вироблення спільних рішень групою людей; Тестування та відлагодження онлайн системи для вироблення спільних рішень групою людей; Економічний розділ; Висновки; Список використаних джерел.

5. Перелік ілюстративного (або графічного) матеріалу: UML-діагрма «Сутнісно-орієнтованна модель бази даних»; UML-діаграма «Класи контролери»; UML-діаграма «Класи представлень»; Інтерфейси додатку.
6. Консультанти розділів роботи»

Розділ змістової частини роботи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 3	Владислав КАБАЧІЙ, к.т.н., доцент кафедри АІІТ	25.09.2025	03.12.2025
4	Наталія БУРЕННІКОВА, д.е.н., проф. каф. ЕПтаВМ	01.11.2025	01.12.2025

7. Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	25.09–05.10.2025	вик.
2	Аналіз доцільності розробки	05.10 – 25.10.2025	вик.
3	Розробка програмного забезпечення	25.10 – 10.11.2025	вик.
4	Тестування розробленого програмного забезпечення	05.11 – 20.11.2025	вик.
5	Підготовка економічної частини	до 01.12.2025	вик.
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.11 – 03.12.2025	вик.
7	Попередній захист роботи	до 03.12.2025	вик.
8	Захист роботи	до 19.12.2025	вик.

Студент _____ (підпис) Олександр ЛЮБУНЬ (прізвище та ініціали)

Керівник роботи _____ (підпис) Владислав КАБАЧІЙ (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.738.5:005.57

Любунь О. І. Онлайн система для вироблення спільних рішень групою людей. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітня програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 152 с.

Українською мовою. Бібліогр.: 31 назв; рис.: 18; табл. 7.

Робота присвячена розробці веб-орієнтованої онлайн-системи, спрямованої на оптимізацію процесів колективного прийняття рішень. Актуальність дослідження обумовлена потребою в ефективній колаборації розподілених команд, платформах громадської участі та корпоративному середовищі.

Метою роботи є створення програмного комплексу, що забезпечує структурований підхід до групового прийняття рішень. Основні завдання включають: аналіз існуючих методів групового прийняття рішень, проектування архітектури системи, реалізацію функцій для створення голосування та візуалізації результатів.

Практична значущість полягає у створенні готового інструменту, який може застосовуватися в освіті, бізнесі та державному управлінні для підвищення якості та швидкості колективних рішень. Система реалізована з використанням ASP.NET Core, SQL Server та Bootstrap, підтримує різні механізми голосування та включає модуль управління завданнями.

Ключові слова: групове прийняття рішень, онлайн-система, веб-розробка, консенсус, система голосування.

ABSTRACT

Liubun O. I. Online system for making joint decisions by a group of people. Master's Qualification Thesis in the specialty 126 – Information Systems and Technologies, educational program – Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2025. 152 p.

In Ukrainian language. Bibliography: 31 titles; Fig.: 18; table 7.

This work is dedicated to the development of a web-oriented online system aimed at optimizing the processes of collective decision-making. The relevance of the research is driven by the need for effective collaboration in distributed teams, platforms for public participation, and the corporate environment.

The goal of this work is to create a software complex that provides a structured approach to group decision-making. The main tasks include: analyzing existing methods of group decision-making, designing the system architecture, and implementing functions for creating discussions, proposing alternatives, voting, and visualizing results.

The practical significance lies in the creation of a ready-to-use tool that can be applied in education, business, and public administration to improve the quality and speed of collective decisions. The system, named "Consensus," is implemented using ASP.NET Core, SQL Server, and Bootstrap. It supports various voting mechanisms (simple "For/Against," point-based system, alternative ranking, and multi-criteria evaluation) and includes a task management module and a social network feature for user contacts.

Keywords: group decision-making, online system, web development, consensus, voting system.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, що характеризується високими темпами глобалізації та цифровізації, все більшу значущість набуває ефективна колаборація та колективне прийняття рішень. Цей процес лежить в основі функціонування розподілених міжнародних команд, Agile-розробки, громадського управління, планування в бізнесі та навчальних процесах. Однак традиційні методи групового обговорення (наприклад, збори, наради) часто виявляються неефективними через географічну роз'єднаність учасників, тимчасові обмеження, а також соціально-психологічні бар'єри, такі як групове мислення (groupthink) або домінування окремих осіб.

Інформаційні технології пропонують потужний інструментарій для подолання цих обмежень. Розробка спеціалізованих онлайн-систем для колективного прийняття рішень дозволяє структурувати процес, забезпечити рівні права для всіх учасників, автоматизувати збір та аналіз думок і, як наслідок, підвищити якість, швидкість та об'єктивність прийнятих рішень. Таким чином, створення доступного, зручного та функціонального програмного рішення для цієї мети є актуальним завданням, що має значний теоретичний і практичний потенціал.

Основною метою дослідження є розробка та практична реалізація веб-орієнтованої онлайн-системи, яка забезпечує ефективний процес вироблення спільних рішень групою людей.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Провести аналіз предметної області, дослідивши існуючі методи, моделі та алгоритми групового прийняття рішень.
2. Сформулювати вимоги до функціоналу та архітектури системи, визначити стек реалізації.

3. Розробити архітектуру системи, включаючи серверну частину (backend), клієнтський інтерфейс (frontend) та схему бази даних.

4. Реалізувати ключовий функціонал системи, зокрема: створення сесій голосування, запрошення учасників, формулювання альтернатив, механізми голосування та аналізу результатів.

5. Провести тестування розробленої системи на працездатність та безпеку.

6. Оцінити ефективність запропонованого рішення та сформулювати висновки щодо подальшого вдосконалення.

Об'єкт дослідження: процес колективного прийняття рішень в групах.

Предмет дослідження: програмні засоби та методики, що підтримують та автоматизують процес групового прийняття рішень в онлайн-середовищі.

Науково-технічний результат роботи полягає в адаптації та програмній реалізації комплексу методів групового прийняття рішень в єдиному, доступному для користувача веб-інтерфейсі.

Практична цінність полягає в створенні готового до використання програмного продукту, який може бути впроваджений для підтримки прийняття рішень в бізнес-структурах, органах місцевого самоврядування, навчальних закладах та громадських організаціях.

Апробація роботи. Основні результати роботи були апробовані на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи».

Публікація. Публікація за темою «Онлайн система для вироблення спільних рішень групою людей» здійснена у вигляді доповіді в секції «Інтелектуальні інформаційні технології та автоматизація» даної конференції [1].

1 ТЕОРЕТИЧНІ ОСНОВИ ТА СУЧАСНИЙ СТАН ДОСЛІДЖЕННЯ КОЛЕКТИВНОГО ПРИЙНЯТТЯ РІШЕНЬ

1.1 Філософські та методологічні основи колективного прийняття рішень

Колективне прийняття рішень як складний соціальний феномен має глибокі філософські корені, що сягають античних часів. Вже у працях давньогрецьких філософів знаходимо перші систематичні обґрунтування переваг колективного розуму перед індивідуальним. Арістотель у своїй фундаментальній праці "Політика" висував тезу про те, що "сума знань багатьох людей перевищує знання одного", аргументуючи це тим, що коли кожен член групи вносить свою частку знань і досвіду, спільне рішення виявляється більш обґрунтованим і враховує різні аспекти проблеми [2]. Ця ідея стала основою для подальшого розвитку теорій групового прийняття рішень протягом наступних століть.

Значний внесок у розвиток філософських основ колективного прийняття рішень зробили мислителі епохи Просвітнення. Жан-Жак Руссо у праці "Про суспільний договір" розробив концепцію "загальної волі" як основи колективних рішень, яка виражає спільні інтереси всіх членів суспільства [2]. Він доводив, що справді справедливі рішення можуть бути прийняті лише шляхом колективного обговорення та врахування інтересів усіх громадян. Джеремі Бентам у роботі "Введення до принципів моралі та законодавства" обґрунтував утилітаристський підхід, згідно з яким групове рішення має максимізувати корисність для найбільшої кількості людей [3]. Цей принцип став важливим орієнтиром для оцінки ефективності колективних рішень у сучасних умовах.

У XX столітті філософські дослідження колективного прийняття рішень отримали новий розвиток. Карл Поппер у праці "Відкрите суспільство та його вороги" обґрунтував важливість критичного діалогу в груповому прийнятті рішень, акцентуючи увагу на необхідності відкритості та самокритичності в групових процесах [4]. Він доводив, що лише через відкриту дискусію та критичне обговорення можна досягти справді обґрунтованих рішень. Юрген Хабермас у роботі "Теорія комунікативної дії" розвинув концепцію "комунікативної раціональності", яка передбачає формування спільних рішень через вільне обговорення та аргументацію [4]. Згідно з його теорією, легітимність колективних рішень залежить від якості комунікативного процесу, в якому всі учасники мають рівні права та можливості для висловлювання своїх позицій.

Сучасні дослідження продовжують розвивати філософські та методологічні основи колективного прийняття рішень, адаптуючи класичні концепції до умов цифрового середовища. Касс Санстейн у праці "Інфотопія: Як багато розумів створюють знання" аналізує вплив цифрових технологій на процес групового прийняття рішень, зокрема досліджує нові форми колективної взаємодії в інтернет-середовищі [5]. Він розробляє методи забезпечення якості та обґрунтованості рішень, прийнятих в умовах цифровізації, що є особливо актуальним для розробки сучасних онлайн-систем підтримки групових рішень.

Методологічні підходи до дослідження колективного прийняття рішень включають низку ключових напрямів. Системний підхід, розроблений Людвігом фон Берталанфі, дозволяє аналізувати групові рішення як цілісні системи, враховуючи складні взаємозв'язки між їх елементами та навколишнім середовищем. Теорія соціальних конструктів Пітера Бергера та Томаса Лукмана розглядає колективні рішення як продукт соціальної взаємодії, що

конструює певну соціальну реальність через процеси інтерпретації та надання значення. Соціально-когнітивний підхід Альберта Бандури досліджує вплив групових динамік на процес прийняття рішень, зокрема роль спостереження, імітації та соціального навчання в формуванні колективних рішень [6].

Окрему методологічну цінність становить теорія раціонального вибору, яка, незважаючи на свою індивідуалістичну орієнтацію, внесла значний внесок у розуміння механізмів агрегації індивідуальних переваг у колективні рішення. Кеннет Ерроу у праці "Соціальний вибір та індивідуальні цінності" обґрунтував фундаментальні обмеження будь-якої системи колективного вибору, сформулювавши свою знамениту "теорему неможливості" [6]. Критика цієї теорії з боку прихильників комунітаризму сприяла розвитку альтернативних підходів, що враховують роль спільних цінностей та ідентичності в груповому прийнятті рішень.

Ці філософські та методологічні основи створюють теоретичний фундамент для розробки ефективних механізмів колективного прийняття рішень у сучасному суспільстві. Синтез класичних філософських концепцій з сучасними методологічними підходами дозволяє розробити більш ефективні інструменти для організації групової взаємодії в цифровому середовищі, що є особливо важливим для створення онлайн-систем підтримки групових рішень, які є предметом даного дослідження. Розуміння цих основ дозволяє не лише аналізувати існуючі системи колективного прийняття рішень, але й проектувати нові, більш ефективні механізми групової взаємодії, що враховують як індивідуальні, так і колективні аспекти прийняття рішень.

1.2 Історична еволюція теорій групового прийняття рішень

Розвиток теорій групового прийняття рішень пройшов складний шлях еволюції, що охоплює період від античності до сучасності. Цей процес відображає зміну парадигм у розумінні природи колективного вибору та вдосконалення методів організації групової взаємодії.

Античний період заклав фундаментальні основи розуміння колективного прийняття рішень. У Стародавній Греції сформувалися дві основні концепції. Платон у праці "Держава" обґрунтував модель "філософа-правителя", але також визнавав значення колегіальних рішень у рамках ідеальної організації держави [7]. Він розробив концепцію "філософського правління", де мудрість поєднується з колективним обговоренням. Платон вважав, що ідеальне суспільство має будуватися на принципах справедливості, де кожен виконує свою функцію, а найважливіші рішення приймаються колегіально філософами-правителями, які керуються не особистими інтересами, а ідеєю загального блага.

Арістотель у "Політиці" розвинув ідею про те, що "багато голів мудріші за одну", вважаючи колективне обговорення найефективнішим способом виявлення оптимальних рішень [7]. Він детально проаналізував різні форми правління та вплив групової динаміки на якість прийнятих рішень. Арістотель виділяв три "правильні" форми правління - монархію, аристократію та політію - і вважав, що навіть у монархії правитель повинен рахуватися з думкою радників та народу. Він доводив, що колективний розум менш схильний до помилок, оскільки різні люди можуть доповнювати знання один одного та виправляти помилки.

У Стародавньому Римі принцип "senatus populusque romanus" (Сенат і народ Риму) втілив практику колективного управління, де рішення приймалися

шляхом обговорення та голосування. Римські філософи, зокрема Цицерон, розвинули концепції громадянської відповідальності та участі в колективному прийнятті рішень. Цицерон у своїх працях наголошував на важливості обговорення та дебатів у прийнятті державних рішень, вважаючи, що істина народжується в суперечці.

Середньовічний період характеризується переважанням теократичних концепцій, проте й тут знаходимо приклади колективного прийняття рішень. У візантійській імперії функціонувала система державних рад, де рішення ухвалювалися колективно після обговорення. У європейських університетах XIII-XIV століть склалися традиції академічного самоврядування, де питання вирішувалися шляхом дискусій та голосування. Середньовічні схоласти, такі як Тома Аквінський, розвинули концепції природного права, які вплинули на розуміння справедливих процедур колективного прийняття рішень [7]. Аквінський вважав, що людські закони мають відображати природне право, а прийняття колективних рішень має ґрунтуватися на розумному обговоренні та прагненні до спільного блага.

Епоха Відродження та Просвітництва принесла радикальні зміни у розумінні колективного прийняття рішень. Нікколо Макіавеллі у праці "Державець" проаналізував механізми колективного управління, наголошуючи на важливості врахування різних думок та балансу сил у процесі прийняття рішень [7]. Він доводив, що мудрий правитель повинен вміти слухати різні думки та приймати рішення на основі колективної мудрості, а не лише власних уподобань.

Жан-Жак Руссо розробив концепцію "загальної волі", яка стала теоретичною основою для демократичних процедур прийняття рішень [7]. Він доводив, що справжнє колективне рішення має виражати не суму індивідуальних інтересів, а спільне благо. Руссо вважав, що кожен громадянин,

беручи участь у колективному прийнятті рішень, повинен думати не "Що вигідно для мене?", а "Що вигідно для всієї спільноти?".

XIX століття відзначилося появою наукового підходу до вивчення групових рішень. Джон Стюарт Мілль у роботі "Про свободі" обґрунтував значення свободи слова та обговорення для прийняття якісних колективних рішень [8]. Він підкреслював важливість захисту меншин та альтернативних думок у групових процесах, вважаючи, що суперництво думок сприяє виявленню істини. Мілль доводив, що навіть непопулярні думки мають право на існування, оскільки вони можуть містити частку правди або ж стимулювати більш глибоке осмислення проблеми.

Карл Маркс розвинув концепцію колективного прийняття рішень у контексті класової боротьби та революційної діяльності, акцентуючи увагу на економічних детермінантах групових рішень [8]. Він вважав, що в класовому суспільстві колективні рішення відображають інтереси панівного класу, а справді демократичне прийняття рішень можливе лише в безкласовому суспільстві.

Початок XX століття позначився розвитком психологічного підходу. Густав Ле Бон у праці "Психологія натовпів" дослідив ірраціональні аспекти групової поведінки, зокрема механізми зараження та підпорядкування в масових рішеннях [8]. Він показав, що в натовпі індивіди втрачають критичне мислення та схильні до емоційних, нерідко ірраціональних рішень.

Курт Левін заклав основи теорії групової динаміки, вивчаючи вплив соціального середовища на процес прийняття рішень [8]. Він розробив концепцію "соціального поля" та його впливу на групову поведінку. Левін довів, що групові рішення залежать не лише від індивідуальних характеристик учасників, але й від структури групи, цілей та норм, що в ній існують.

Середина XX століття принесла значний внесок у розвиток формальних теорій. Кеннет Ерроу у роботі "Соціальний вибір та індивідуальні цінності" сформулював "теорему неможливості", яка продемонструвала обмеженість ідеальних систем колективного вибору [8]. Ця теорема показала, що не існує ідеальної системи голосування, яка б одночасно задовольняла всім бажаним критеріям: транзитивності, універсальності, незалежності від сторонніх альтернатив, суверенітету виборців та відсутності диктатора.

Герберт Саймон запропонував концепцію "обмеженої раціональності", що пояснює реальні механізми групового прийняття рішень [8]. Він довів, що в реальних умовах групи прагнуть не до оптимальних, а до задовільних рішень, оскільки обчислювальні можливості людина обмежені, а інформація нерідко неповна. Саймон ввів поняття "задоволення" - пошуку прийнятного, а не ідеального рішення.

Кінець XX - початок XXI століття характеризується розвитком комп'ютерних моделей та інтернет-технологій. Томас Шелінг дослідив механізми координації в групових рішеннях, зокрема явище "фокусних точок" - природних рішень, до яких схильні групи [9]. Він показав, що люди в групах часто досягають узгодження без явної домовленості, спираючись на спільні культурні коди та очікування.

Касс Санстейн проаналізував вплив Інтернету на процеси колективного прийняття рішень, зокрема явище "поляризації груп" під впливом інтернет-спільнот [9]. Він довів, що в інтернет-середовищі люди схильні об'єднуватися в групи з подібними поглядами, що призводить до посилення крайніх позицій та ускладнює досягнення консенсусу.

Сучасний етап розвитку теорій групового прийняття рішень характеризується інтеграцією різних підходів та розробкою практичних методів підтримки колективних рішень в умовах цифровізації суспільства.

Дослідження сьогодні зосереджені на розробці алгоритмів колективного інтелекту, методів онлайн-фасилітації та систем підтримки групових рішень, що враховують складність сучасних соціальних взаємодій. Особливу увагу приділяється питанням штучного інтелекту та його ролі в підтримці групових рішень, а також етичним аспектам використання нових технологій у колективному прийнятті рішень.

1.3 Сутність та класифікація колективних рішень

Колективне прийняття рішень є складним багатоаспектним процесом, який характеризується спільною діяльністю групи людей, спрямованою на вибір оптимального варіанту дій з множини можливих альтернатив [10]. Сутність цього процесу полягає у взаємодії учасників групи, спрямованій на досягнення консенсусу або компромісу через обмін інформацією, аргументацію позицій та узгодження інтересів. Цей процес відбувається в рамках певної соціальної структури та регулюється формальними і неформальними нормами поведінки.

До ключових характеристик колективного прийняття рішень належать: наявність спільної мети, яка об'єднує учасників і визначає напрямок їхньої діяльності; взаємозалежність членів групи, коли результат залежить від внеску кожного учасника та їх спільних зусиль; наявність комунікації та обміну інформацією, що забезпечує передачу знань та досвіду; формалізований або неформальний процес узгодження позицій, який може включати різноманітні процедури та методи обговорення.

Колективні рішення відрізняються від індивідуальних більшою обґрунтованістю, оскільки враховують різноманітні точки зору та досвід

учасників, а також вищою легітимністю через врахування інтересів усіх зацікавлених сторін. Дослідження показують, що колективні рішення зазвичай відрізняються більш високою якістю та кращою проробленістю [11]. Однак процес їх прийняття зазвичай є більш тривалим та енергоємним, вимагає значних ресурсів на координацію та комунікацію.

Класифікація колективних рішень може проводитися за різними критеріями. За рівнем структурованості проблеми розрізняють структуровані, напівструктуровані та неструктуровані рішення. Структуровані проблеми мають чіткі алгоритми вирішення, тоді як неструктуровані вимагають креативного підходу та експертної оцінки. За способом узгодження позицій виділяють рішення, прийняті шляхом консенсусу, компромісу, голосування чи авторитарного затвердження. Консенсус передбачає повну згоду всіх учасників, тоді як компроміс базується на взаємних поступках.

За кількістю критеріїв оцінки розрізняють однокритеріальні та багатокритеріальні рішення. Однокритеріальні рішення ґрунтуються на одному основним критерії, тоді як багатокритеріальні вимагають врахування та зважування множини факторів. За тривалістю процесу прийняття виділяють оперативні, тактичні та стратегічні рішення. Оперативні рішення приймаються швидко для вирішення поточних проблем, тоді як стратегічні вимагають глибокого аналізу та довгострокового планування. За рівнем участі розрізняють рішення, прийняті повністю автономною групою, рішення з обмеженою участю та рішення, що вимагають затвердження зовнішніми інстанціями [12].

Переваги колективних рішень включають підвищення якості рішень за рахунок агрегації знань та досвіду різних учасників, збільшення креативності через синергетичний ефект групової взаємодії, покращене розуміння та подальшу імплементацію рішення через участь у його розробці, а також

підвищення легітимності прийнятого рішення. Крім того, колективні рішення сприяють розвитку командного духу та підвищенню мотивації учасників.

Недоліки та обмеження колективного прийняття рішень проявляються у значних часових витратах на організацію та координацію групового процесу, ризику виникнення групового мислення (groupthink), впливі соціального тиску та домінування окремих осіб, явищі "соціального лінощів" (social loafing), а також ускладненні процесу в разі конфлікту інтересів. Важливо також враховувати можливість виникнення ефекту "групової поляризації", коли групове рішення виявляється більш радикальним, ніж індивідуальні рішення окремих учасників.

Сучасні підходи до класифікації колективних рішень також враховують ступінь використання цифрових технологій, розрізняючи традиційні офлайн-рішення, гібридні форми та повністю онлайн-процеси, що реалізуються через спеціалізовані платформи та інструменти колективної роботи. Ця класифікація стає особливо актуальною в умовах цифровізації та розвитку дистанційних форм співпраці.

1.4 Формальні моделі та алгоритми колективного вибору

Сучасні підходи до колективного прийняття рішень базуються на формальних моделях та алгоритмах, що забезпечують структурований та об'єктивний процес вибору. Ці методи дозволяють ефективно агрегувати індивідуальні переваги учасників у групові рішення, мінімізуючи вплив суб'єктивних факторів.

Основним теоретичним фундаментом для аналізу колективних рішень є теорія соціального вибору, яка досліджує методи об'єднання індивідуальних

переваг у колективні рішення [13]. Кеннет Ерроу у своїй фундаментальній теоремі неможливості продемонстрував принципові обмеження будь-якої системи колективного вибору, показавши, що не існує ідеальної системи голосування, яка б одночасно задовольняла всім критеріям справедливості. Ця теорема встановила важливі орієнтири для розробки практичних методів групового прийняття рішень.

Серед практичних методів особливе місце займають моделі голосування. Система простої більшості, хоча і є найпоширенішою, має суттєві недоліки, такі як можливість вибору альтернативи, яка не є найкращою для більшості. Рейтингові системи голосування, такі як метод Борда, дозволяють учасникам більш точно виражати свої вподобання через ранжування альтернатив за пріоритетністю. Ці методи особливо ефективні при необхідності врахування ступеня переваги між різними альтернативами.

Для складних багатокритеріальних рішень ефективно використовується метод аналізу ієрархій (Analytic Hierarchy Process - AHP), розроблений Томасом Сааті [14]. Цей метод дозволяє структурувати складні проблеми у вигляді ієрархії критеріїв та альтернатив, використовуючи парні порівняння для визначення вагових коефіцієнтів. AHP особливо ефективний для рішень, що вимагають врахування як кількісних, так і якісних факторів, та дозволяє кількісно оцінити узгодженість думок експертів.

Теорія коаліцій аналізує формування груп та альтернатив у процесах колективного прийняття рішень. Модель Шеплі значення дозволяє розподілити "виграш" між учасниками коаліції пропорційно їхньому внеску в загальний результат. Цей підхід особливо корисний при аналізі рішень, де важливо врахувати внесок кожного учасника, наприклад, при розподілі ресурсів або визначенні частки участі в спільних проектах.

Методи багатоатрибутивної корисності (Multi-Attribute Utility Theory - MAUT) забезпечують формальну основу для оцінки альтернатив за кількома критеріями [15]. Ці методи дозволяють агрегувати індивідуальні переваги учасників з урахуванням важливості різних критеріїв, забезпечуючи системний підхід до оцінки складних альтернатив. Вони особливо корисні при прийнятті стратегічних рішень, коли необхідно врахувати множину взаємопов'язаних факторів.

Алгоритми консенсусу розроблені для ситуацій, коли необхідно досягти повної або часткової згоди всіх учасників. Метод Дельфі, що передбачає анонімні ітераційні обговорення, дозволяє поступово наближатися до консенсусу, мінімізуючи вплив соціального тиску. Інші підходи, такі як методи пошуку компромісних рішень, орієнтовані на знаходження варіантів, що задовольняють всіх учасників хоча б частково. Ці методи особливо важливі в умовах конфлікту інтересів або при необхідності врахування протилежних точок зору.

Сучасні дослідження також включають еволюційні алгоритми та методи колективного інтелекту для вирішення складних задач колективного вибору. Ці методи, засновані на принципах природного відбору та самоорганізації, дозволяють ефективно знаходити рішення в умовах невизначеності та великої розмірності простору пошуку. Вони особливо перспективні для розв'язання складних оптимізаційних задач з багатьма обмеженнями та критеріями.

Інтегровані платформи підтримки прийняття рішень (Group Decision Support Systems - GDSS) поєднують різні формальні моделі в єдиних програмних рішеннях. Такі системи забезпечують комплексну підтримку всього циклу колективного прийняття рішень – від збору інформації та генерації альтернатив до їх оцінки та вибору оптимального варіанту. Вони

дозволяють автоматизувати багато трудомістких процесів та забезпечити прозорість процедур прийняття рішень.

Кожен з цих формальних підходів має свої переваги та обмеження, і вибір конкретного методу залежить від характеру проблеми, кількості учасників, наявної інформації та вимог до якості остаточного рішення. Важливим аспектом є також врахування психологічних та соціальних факторів, які можуть впливати на ефективність застосування формальних методів у реальних умовах.

1.5 Психологічні механізми групової динаміки при прийнятті рішень

Процес колективного прийняття рішень знаходиться під значним впливом психологічних чинників, які формують групову динаміку та безпосередньо впливають на якість кінцевого рішення. Розуміння цих механізмів є критично важливим для проектування ефективних систем підтримки групових рішень [16].

Групове мислення (groupthink) є одним з найбільш вивчених психологічних явищ у груповій динаміці. Воно виникає, коли прагнення до групової гармонії та консенсусу стає настільки домінуючим, що починає переважати над реалістичною оцінкою альтернативних варіантів дій. Цей феномен, детально досліджений Ірвінгом Джанісом, характеризується тим, що учасники групи несвідомо придушують власні сумніви, уникають критичного обговорення та створюють ілюзію одностайності [16]. Дослідження показують, що групове мислення найчастіше виникає в умовах високої групової згуртованості, соціальної ізоляції від зовнішніх думок, стресових ситуацій та за відсутності чітко визначених процедур прийняття рішень.

Ефект соціального ледарства (social loafing) проявляється у значному зниженні індивідуальної продуктивності окремих членів групи при виконанні колективних завдань. Кожен учасник може несвідомо прикладати менше зусиль, розраховуючи на роботу інших членів команди [17]. Цей ефект особливо помітний у великих групах, де персональний внесок кожного учасника менш помітний, а також у завданнях, де складно оцінити індивідуальний внесок кожного члена групи. Дослідження показують, що соціальне ледарство може знижувати загальну ефективність групи на 20-50%.

Феномен поляризації групових поглядів спостерігається, коли після колективного обговорення початкові позиції учасників стають більш радикальними та вираженими. Наприклад, якщо члени групи спочатку демонстрували помірковану позицію щодо певного ризику, після групового обговорення їх оцінки можуть стати значно сміливішими чи, навпаки, обережнішими [18]. Це явище пояснюється тим, що в процесі дискусії учасники чують додаткові аргументи на підтримку своєї початкової позиції та відчувають соціальну підтримку однодумців.

Конформізм як механізм соціального впливу проявляється у зміні індивідуальної поведінки або поглядів під впливом реального або уявного тиску групової більшості. Класичні експерименти Соломона Аша демонструють, що значна кількість людей схильна сумлінно погоджуватися з очевидно неправильними рішеннями, якщо їх підтримує більшість членів групи. Цей механізм особливо сильний у ситуаціях невизначеності, коли люди схильні довіряти груповій думці більше, ніж власному розсудку.

Вплив лідерських якостей на групову динаміку проявляється через стиль керівництва процесом прийняття рішень. Авторитарний підхід часто призводить до пригнічення альтернативних точок зору та творчого мислення, тоді як демократичний стиль лідерства сприяє більш повному розкриттю

потенціалу всіх учасників. Лідер також може несвідомо направляти групу до бажаного для нього результату через маніпуляцію порядком висловлювань, специфічне формулювання питань або нерівномірний розподіл часу для обговорення.

Ефект якорення (anchoring effect) в групових рішеннях виникає, коли початкова інформація або перша висловлена думка стає орієнтиром для подальшого обговорення. Навіть якщо ці відомості є випадковими або неповними, вони можуть значно обмежити діапазон розглядуваних альтернатив та вплинути на кінцеве рішення групи. Цей феномен особливо небезпечний у ситуаціях, коли група прагне до швидкого прийняття рішення.

Селективне сприйняття інформації в груповому контексті проявляється у тенденції учасників несвідомо шукати та переоцінювати дані, що підтверджують початкові припущення, одночасно ігноруючи або применшуючи значення суперечливих фактів. Цей ефект значно посилюється в груповому середовищі, коли члени групи взаємно підтримують один одного в раціоналізації власних упереджень.

Вплив групової ідентичності на процес прийняття рішень проявляється в схильності членів групи віддавати перевагу рішенням, що підкреслюють позитивну відмінність власної групи від інших соціальних груп. Це може призводити до автоматичного відхилення потенційно корисних ідей через їх асоціацію з "іншими" групами, а також до перебільшення переваг "своїх" рішень.

Розуміння цих психологічних механізмів дозволяє розробити ефективні методи та процедури для їх нейтралізації, а також створити більш збалансоване та об'єктивне середовище для колективного прийняття рішень. Сучасні системи підтримки групових рішень повинні враховувати ці психологічні аспекти для забезпечення високої якости та об'єктивності прийнятих рішень.

1.6 Соціальні та культурні фактори впливу на групові рішення

Процес колективного прийняття рішень знаходиться під значним впливом соціокультурних чинників, які формують ціннісні орієнтації, комунікаційні паттерни та норми поведінки учасників. Розуміння цих впливів є вирішальним для створення ефективних систем підтримки групових рішень.

Вплив соціальної ідентичності на групову динаміку проявляється через механізми соціальної категоризації та порівняння. Учасники несвідомо ідентифікують себе з певними соціальними групами, що формує їхні позиції та впливає на взаємодію в процесі прийняття рішень [19]. Це часто призводить до явища "інгрупової упередженості", коли члени групи надмірно високо оцінюють внесок "своїх" та применшують значення думок "чужих". Дослідження соціальної ідентичності показують, що цей механізм може значно впливати на об'єктивність оцінки альтернатив та формування групового консенсусу.

Статусні диференціали значно детермінують процес групової взаємодії. Дослідження показують, що особи з вищим соціальним статусом отримують більше можливостей для висловлювання, їхні пропозиції сприймаються більш серйозно, і вони мають непропорційно великий вплив на кінцевий результат [20]. Це може призводити до ситуацій, коли цінні ідеи учасників з нижчим статусом ігноруються через соціальні бар'єри. В організаційному контексті це може проявлятися у вигляді ієрархічних структур впливу, де формальна посада часто переважає над експертною компетентністю.

Крос-культурні особливості значно варіюють підходи до колективного прийняття рішень. Культури з високою дистанцією влади демонструють схильність до ієрархічних моделей, тоді як культури з низькою дистанцією влади віддають перевагу рівним, консенсусним процедурам [21]. Ці

відмінності вимагають різних підходів до організації групових процесів. Наприклад, в колективістичних культурах рішення часто приймаються з урахуванням групової гармонії, тоді як в індивідуалістичних культурах більше цінується незалежність думки.

Гендерні аспекти групової динаміки виявляються в комунікаційних стилях та стратегіях впливу. Дослідження вказують на тенденцію до використання співпрацюючих стратегій і прагнення до консенсусу в одних соціальних групах, тоді як інші групи частіше використовують конкурентні підходи та орієнтуються на домінування. Однак ці закономірності суттєво модифікуються індивідуальними особливостями та соціокультурним контекстом.

Соціальні норми формують невидимі рамки, що регулюють групову поведінку. Ці неформальні правила визначають допустимі способи висловлювання думок, форми критики, тривалість виступів та інші аспекти комунікації. Порушення цих норм може спричинити соціальні санкції, навіть якщо пропозиції порушника є конструктивними.

Мережеві структури впливають на якість групових рішень через особливості комунікаційних потоків. Централізовані мережі з одним комунікаційним центром забезпечують швидкість прийняття рішень, але можуть обмежувати розмаїття думок, тоді як децентралізовані структури сприяють більш повному використанню колективного інтелекту.

Організаційна культура визначає ціннісні орієнтації та поведінкові паттерни в процесах прийняття рішень. Інноваційні організації заохочують різноманітність думок і експериментування, тоді як бюрократичні структури можуть обмежувати творче мислення через надмірну формалізацію процедур.

Мовні та комунікаційні бар'єри у міжкультурних групах можуть значно ускладнювати процес прийняття рішень. Відмінності в мовних нормах, стилях

аргументації, невербальній комунікації та інтерпретації пауз часто призводять до непорозумінь і зниження ефективності спільної роботи.

Релігійні та світоглядні фактори формують етичні рамки та ціннісні критерії оцінки альтернатив. В організаціях з вираженою релігійною ідентичністю рішення часто приймаються з урахуванням релігійних принципів і моральних норм, що може суттєво впливати на вибір кінцевого варіанту.

Розуміння цих соціокультурних чинників дозволяє розробити більш ефективні стратегії організації групових процесів та створення інклюзивного середовища для колективного прийняття рішень.

1.7 Сучасні технології та інструменти підтримки колективних рішень

Сучасні технології значно розширили можливості організації та оптимізації процесів колективного прийняття рішень. Ці інструменти дозволяють подолати географічні, часові та організаційні бар'єри, забезпечуючи ефективну взаємодію учасників незалежно від їх місцезнаходження.

Системи підтримки групових рішень (Group Decision Support Systems - GDSS) представляють собою інтегровані програмні комплекси, що поєднують функції комунікації, аналізу даних та управління груповими процесами [22]. Вони забезпечують структуроване середовище для проведення мозкових штурмів, обговорення альтернатив, оцінки варіантів та формування кінцевого рішення. Сучасні GDSS включають інструменти для анонімного голосування, візуалізації результатів та аналізу узгодженості думок. Дослідження показують, що використання GDSS може підвищити ефективність групових

рішень на 25-40% шляхом структурування процесу та зменшення впливу соціально-психологічних бар'єрів.

Хмарні платформи спільної роботи стали основним інструментом для розподілених команд. Такі сервіси, як Microsoft Teams, Slack та Google Workspace, забезпечують можливості реального часу для спільного редагування документів, проведення відеоконференцій та організації обговорень [23]. Вони дозволяють створювати спеціалізовані канали комунікації для різних аспектів проблеми, забезпечуючи структурований підхід до колективного прийняття рішень. Особливістю сучасних платформ є інтеграція з іншими бізнес-інструментами та можливість роботи з мобільних пристроїв, що забезпечує безперервність процесу прийняття рішень.

Штучний інтелект та машинне навчання відкривають нові горизонти для підтримки колективних рішень. AI-алгоритми здатні аналізувати великі обсяги даних, виявляти закономірності та генерувати альтернативні сценарії [24]. Системи рекомендацій на основі машинного навчання можуть запропонувати оптимальні рішення на основі аналізу подібних ситуацій та історичних даних. Сучасні AI-інструменти також можуть аналізувати групову динаміку та виявляти потенційні конфлікти на ранніх стадіях, пропонуючи стратегії їх вирішення.

Блокчейн-технології пропонують нові підходи до забезпечення прозорості та безпеки колективних рішень [25]. Децентралізовані системи голосування на основі блокчейну забезпечують незмінність результатів, верифікацію учасників та захист від маніпуляцій. Це особливо важливо для прийняття рішень у сфері управління організаціями та громадських ініціатив. Технологія розподілених реєстрів дозволяє створювати прозорі системи голосування з підтвердженням кожного голосу та автоматичним підрахунком результатів.

Інструменти аналізу великих даних дозволяють обробляти та візуалізувати складні набори інформації, що є критично важливим для обґрунтованого прийняття рішень [26]. Платформи типу Tableau та Power BI забезпечують інтерактивні інструменти для дослідження даних та виявлення прихованих закономірностей. Це дозволяє групам приймати рішення на основі об'єктивних даних, а не суб'єктивних уподобань. Сучасні системи аналітики також включають функції прогнозування та моделювання наслідків різних рішень.

Віртуальна та доповнена реальність починають використовуватися для створення імерсивних середовищ прийняття рішень. Ці технології дозволяють візуалізувати складні дані у тривимірному просторі, проводити інтерактивні симуляції та моделювати наслідки різних рішень. Особливо ефективні вони для прийняття рішень у сфері дизайну, архітектури та комплексного планування.

Мобільні технології забезпечують участь у процесі прийняття рішень незалежно від місцезнаходження учасників. Спеціалізовані додатки для смартфонів дозволяють брати участь у голосуваннях, обговореннях та оцінці альтернатив у будь-який час. Це особливо важливо для організацій з розподіленими командами та мобільними співробітниками.

Сучасні технології продовжують стрімко розвиватися, пропонуючи все більш складні та ефективні інструменти для підтримки колективних рішень. Ключовим завданням стає інтеграція цих технологій у єдині системи, що забезпечують комплексну підтримку всього циклу прийняття рішень – від ідентифікації проблеми до реалізації та моніторингу результатів.

1.8 Сучасні дослідження методів колективного прийняття рішень

Сучасний етап розвитку теорії колективного прийняття рішень характеризується інтенсивною інтеграцією формальних математичних методів з психологією групової динаміки та інформаційними технологіями. Це створює передумови для розробки нових гібридних підходів, орієнтованих на підвищення ефективності групової взаємодії в різних середовищах.

Розвиток формальних моделей та алгоритмів. Значний прогрес досягнуто в області формалізації процедур багатокритеріального вибору та агрегації індивідуальних переваг. У роботах Бойка М. В. та Савчука Т. П. [27] детально досліджено методи обробки експертних оцінок в умовах невизначеності, запропоновано алгоритми узгодження колективних рішень на основі теорії нечітких множин. Паралельно розвиваються дослідження з адаптації класичних методів, таких як аналіз ієрархій (AHP) та теорія корисності, для потреб онлайн-взаємодії, що дозволяє структурувати складні багатофакторні проблеми [28].

Психологічні та соціальні аспекти в цифровому середовищі. Сучасні дослідження акцентують увагу на трансформації групової динаміки під впливом цифрових комунікаційних платформ. Аналізуються механізми виникнення поляризації думок у віртуальних спільнотах, ефекти соціального впливу в умовах анонімності та часткової ідентифікації, а також особливості формування довіри в розподілених командах [29]. Ці роботи становлять теоретичну основу для проектування механізмів фасилітації, спрямованих на зменшення когнітивних упереджень та покращення якості комунікації.

Технологічні рішення та системи підтримки. Активно розвивається напрямок, пов'язаний із розробкою програмних комплексів, що інтегрують різноманітні методи голосування, інструменти анонімного обговорення та

системи аналітики. Дослідження зосереджені на створенні архітектур, здатних забезпечити масштабованість, безпеку даних та зручний користувацький досвід [30]. Особливу увагу приділяється можливостям інтеграції штучного інтелекту для автоматизованого аналізу дискусій, виявлення консенсусу та генерації зведених пропозицій.

Практичне впровадження та прикладні аспекти. Сучасні розробки все більше орієнтуються на конкретні сфери застосування: від корпоративного управління та академічного середовища до громадських консультацій та платформ електронної демократії. Проводяться польові дослідження, що оцінюють ефективність різних методів голосування та процедур обговорення в реальних умовах, що дозволяє вдосконалювати інструментарій на основі емпіричних даних [31].

Таким чином, сучасні дослідження формують комплексне розуміння колективного прийняття рішень як багатоаспектного явища, що потребує поєднання методологічної строгості, глибокого розуміння соціально-психологічних процесів та інноваційних технологічних рішень. Ці напрацювання створюють міцну основу для розробки нових поколінь інтерактивних систем, здатних підвищити ефективність групової роботи в умовах цифрової трансформації суспільства.

Висновки

У першому розділі проведено комплексний аналіз теоретичних основ та сучасного стану дослідження колективного прийняття рішень. Установлено, що даний процес має значні переваги перед індивідуальними рішеннями у плані обґрунтованості та легітимності, проте супроводжується характерними

ризиками, такими як групове мислення, соціальне ледарство та поляризація поглядів.

Систематизовано формальні моделі та алгоритми колективного вибору, що підтвердило наявність потужного математичного апарату для підтримки групових рішень – від класичних методів голосування до складних багатокритеріальних підходів.

Критично важливим результатом розділу є виявлення ключового впливу психологічних механізмів групової динаміки та соціокультурних факторів на якість прийнятих рішень. Це обумовлює необхідність спеціальних механізмів їх нейтралізації у рамках проектованих систем.

Огляд сучасних технологій та існуючих програмних рішень показав, що на ринку відсутнє доступне, універсальне рішення, яке поєднувало б наукову обґрунтованість методів, простоту використання та гнучкість застосування в різних контекстах.

Отримані теоретичні результати створюють основу для розробки онлайн-системи, визначаючи потреби в інтеграції різноманітних методів прийняття рішень, врахуванні психологічних та соціальних аспектів групової взаємодії та забезпеченні інтуїтивності інтерфейсу.

2 АНАЛІЗ ДОЦІЛЬНОСТІ РОЗРОБКИ ОНЛАЙН СИСТЕМИ ДЛЯ ВИРОБЛЕННЯ СПІЛЬНИХ РІШЕНЬ ГРУПОЮ ЛЮДЕЙ

2.1 Аналіз існуючих методів групового прийняття рішень та аналогічних сервісів

Сучасний цифровий ландшафт демонструє значну різноманітність інструментів для організації групової роботи, проте ретельний аналіз виявляє системні недоліки та суттєві прогалини в існуючих рішеннях. Дослідження ринку програмного забезпечення дозволяє ідентифікувати кілька чітких категорій інструментів, кожна з яких має характерні обмеження, що значно знижують їх ефективність у контексті комплексного групового прийняття рішень.

Найпоширенішими є інструменти для створення опитувань, серед яких лідирують Google Forms, SurveyMonkey та StrawPoll. Ці платформи, хоч і ефективні для збору інформації та проведення простих голосувань, демонструють повну непристосованість для складних процесів колективного прийняття рішень. Вони не підтримують спеціалізовані аналітичні методи, такі як багатокритеріальна оцінка чи структуроване ранжування альтернатив. Крім того, цим інструментам бракує можливостей для глибокого обговорення пропозицій, поетапного аналізу варіантів чи ітераційного вдосконалення рішень. Важливим недоліком є також відсутність можливості створення структурованої мережі контактів для постійної співпраці, що обмежує їх використання для стратегічних рішень, які потребують постійної комунікації між учасниками процесу.

Окрему категорію становлять вузькоспеціалізовані інструменти планування, зокрема Doodle та When2Meet. Ці сервіси ефективно вирішують

конкретне завдання – пошук оптимального часу для зустрічі – проте їх функціонал неможливо адаптувати для складніших управлінських, фінансових чи стратегічних рішень. Обмеженість застосування робить їх малокорисними для комплексних процесів колективного вибору, де необхідно враховувати множинні критерії та проводити глибокий аналіз альтернатив. Крім того, цим сервісам не вистачає можливостей для ведення архіву прийнятих рішень, що ускладнює моніторинг їх виконання та аналіз ефективності прийнятих рішень у довгостроковій перспективі.

Платформи для мозкового штурму, такі як Tricider, IdeaBoardz та Miro, пропонують розширені можливості для колективної роботи. Вони забезпечують базові засоби обговорення ідей та простих голосувань, однак їх аналітичні можливості залишаються обмеженими. У цих системах відсутні різноманітні методи голосування, гнучкі налаштування процесу прийняття рішень, інструменти для аналізу результатів та механізми контролю реалізації прийнятих рішень. Особливо варто відзначити відсутність інтегрованих механізмів для управління простими завданнями, що не дозволяє перетворити прийняті рішення на конкретні дії з чіткими дедлайнами та відповідальними особами.

Потужні системи управління проектами, серед яких Trello, Asana, Jira та Basecamp, також не пропонують комплексних рішень для групового прийняття рішень. Хоча вони забезпечують ефективне планування та контроль виконання завдань, функціонал для колективного прийняття рішень в них реалізований лише фрагментарно, через сторонні інтеграції та плагіни, що призводить до роздробленості процесу та додаткових витрат. Важливою проблемою є також складність налаштування цих інструментів для нестандартних сценаріїв голосування, що значно обмежує їх застосування в ситуаціях, які потребують гнучких підходів до прийняття колективних рішень.

В таблиці 2.1, 2.2 та 2.3 наведено переваги розроблюваної системи порівняно з аналогами.

Таблиця 2.1 – Переваги розроблюваної системи порівняно з аналогами

Функціонал	Google Forms	Doodle	Tricider	Trello	Розроблювана система
Метод "За і Проти"	5[1]	3[2]	5[3]	1[4]	5
Бальна система	3[5]	2[6]	2[7]	1[4]	5
Ранжування альтернатив	1[8]	1[8]	1[8]	1[8]	5
Багатокритеріальна оцінка	1[9]	1[9]	1[9]	1[9]	5
Налаштування публічності	4[10]	4[11]	4[12]	3[13]	5
Обмеження голосування за часом	1[14]	1[15]	1[16]	1[17]	5
Мережа контактів	4[18]	4[18]	4[18]	3[19]	5
Візуалізація результатів голосування	3[20]	3[21]	1[22]	1[23]	5
Управління завданнями	1[24]	1[24]	1[24]	5[25]	5

Шкала оцінювання функціоналу:

1 – повна відсутність функціоналу;

2 – базовий рівень (функціонал важкий у використані);

3 – середній рівень (на функціонал накладені обмеження);

4 – високий рівень (функціонал володіє незначними обмеженнями);

5 – повний функціонал з розширеними можливостями.

Деталізація оцінок (номер у списку відповідає номеру оцінки в таблиці):

1. (5): можливість створення опитувань з вибором варіантів.
2. (3): можливість створення словесних опитувань, які можливі тільки під час планування зустрічей.
3. (5): можливість створення опитувань з вибором варіантів.
4. (1): відсутність вбудованого функціоналу без сторонніх інтеграцій.
5. (3): можливість створення шкал, але діапазон балів обмежений – від 1 до 10.
6. (2): можливість створення опитувань з вибором варіантів відповіді, які задаються творцем голосування.
7. (2): можливість створення опитувань з вибором варіантів відповіді, які задаються творцем голосування.
8. (1): повна відсутність у всіх аналогах.
9. (1): повна відсутність у всіх аналогах.
10. (4): сутність налаштувань публічності для завдань.
11. (4): відсутність налаштувань публічності для завдань.
12. (4): відсутність налаштувань публічності для завдань.
13. (3): відсутність налаштувань публічності для голосувань.
14. (1): відсутність функціоналу обмеження за часом.
15. (1): відсутність функціоналу обмеження за часом.
16. (1): відсутність функціоналу обмеження за часом.
17. (1): відсутність функціоналу обмеження за часом.
18. (4): можливість створення мережі контактів лише для голосувань.

19. (3): можливість створення мережі контактів лише для завдань.
20. (3): відсутність можливість візуалізації критеріїв.
21. (3): відсутність можливість візуалізації критеріїв.
22. (1): відсутність графічної візуалізації.
23. (1): відсутність можливість візуалізації критеріїв.
24. (1): відсутність у більшості аналогів.
25. (5): повноцінна система управління завданнями.

Таблиця 2.2 – Порівняльний аналіз якості реалізації функцій

Критерій	Існуючі рішення	Розроблювана система
Інтеграція процесів	Фрагментована	Цілісна
Глибина аналізу	Поверхнева	Комплексна
Гнучкість налаштувань	Обмежена	Розширена
Зручність інтерфейсу	Різноманітна	Уніфікована
Адаптивність до різних типів рішень	Спеціалізована	Універсальна
Вартість використання	Часто потребує платежів	Економічно ефективна
Можливість кастомізації	Обмежена	Висока
Підтримка різних форматів голосування	Базова	Розширена

Запропонована до розробки онлайн-система усуває ці системні недоліки шляхом реалізації цілісного комплексного підходу. На відміну від вузькоспеціалізованих сервісів, вона поєднує чотирьох різних методи голосування - від простого «За і Проти» до складного багатокритеріального

аналізу. Це забезпечує універсальність системи та можливість її застосування в різних сферах діяльності – від управління бізнес-проектами до організації роботи громадських об'єднань. Особливістю системи є реалізація механізму багатокритеріальної оцінки, який дозволяє враховувати вагу кожного критерію при прийнятті рішення, що є унікальною перевагою порівняно з більшістю аналогічних рішень.

Важливою перевагою системи є її гнучкість, що досягається через розширені налаштування процесу голосування. Користувачі можуть налаштовувати публічність голосування, встановлювати часові обмеження, визначати права доступу для різних груп учасників. Це робить систему придатною для різноманітних сценаріїв використання – від публічних обговорень до конфіденційних опитувань. Додатковою перевагою є можливість створення шаблонів голосувань для типових сценаріїв, що значно спрощує процес організації повторюваних процедур прийняття рішень.

Ключовою інноваційною перевагою є повна інтеграція процесів прийняття та виконання рішень. Система не лише допомагає прийняти колективне рішення, але й надає комплексні інструменти для його практичної реалізації через модуль управління завданнями. Це забезпечує замкнутий цикл управління - від генерації ідей через обговорення та голосування до практичної реалізації прийнятих рішень. Можливість призначення відповідальних осіб, встановлення дедлайнів та відстеження прогресу виконання завдань робить систему інструментом не лише для прийняття, але й для ефективної реалізації рішень.

Потужний аналітичний модуль забезпечує глибоке та всебічне аналізування результатів голосування. Автоматична побудова графіків, формування звітів, візуалізація динаміки голосування надають користувачам інтуїтивно зрозумілий інструмент для аналізу. Це дозволяє не тільки фіксувати

результати голосування, але й аналізувати тенденції, виявляти закономірності та приймати більш обґрунтовані рішення.

Соціальний аспект системи, реалізований через механізм мережі контактів та груп співпраці, трансформує її з інструменту для разових опитувань на повноцінну платформу для постійної спільної роботи. Це дозволяє формувати спільні простори для прийняття рішень, встановлювати довгострокові зв'язки між учасниками та створювати історію прийнятих рішень.

Таблиця 2.3 – Переваги розроблюваної системи порівняно з аналогами

Аспект	Перевага розроблюваної системи
Функціональність	Комплексне рішення "все в одному"
Аналітичні можливості	Глибокий аналіз з візуалізацією
Гнучкість	Адаптація до різних типів рішень
Інтеграція	Єдиний простір для прийняття та виконання рішень
Користувацький досвід	Уніфікований інтуїтивний інтерфейс
Економічна ефективність	Зменшення витрат на кілька окремих інструментів
Можливість кастомізації	Високий рівень адаптації під потреби
Масштабованість	Підтримка великої кількості користувачів
Безпека	Різні рівні доступу та конфіденційності

Таким чином, проведений аналіз яскраво демонструє, що запропонована розробка не дублює функціонал існуючих продуктів, а заповнює істотні системні прогалини. Комплексний підхід, інтеграція всіх етапів прийняття та реалізації рішень, гнучкість та потужні аналітичні

можливості забезпечують розроблюваній системі значні конкурентні переваги. Порівняльні таблиці наочно ілюструють переваги запропонованого рішення над існуючими аналогами за всіма ключовими параметрами, що остаточно обґрунтовує доцільність її створення та впровадження. Система забезпечує цілісний підхід до управління груповими рішеннями, починаючи від їх ініціації та обговорення, через процедуру голосування, і закінчуючи контролем виконання та аналізом ефективності прийнятих рішень.

2.2 Аналіз технологічних рішень

Вибір технологічного стеку для розробки системи групового прийняття рішень ґрунтується на комплексному аналізі технічних, економічних та операційних факторів. Використання ASP.NET Core та SQL Server як основних технологій розробки є оптимальним рішенням, що забезпечує високу продуктивність, безпеку та масштабованість системи.

ASP.NET Core як основна платформа розробки забезпечує високошвидкісну обробку запитів завдяки оптимізованому конвеєру middleware та підтримці асинхронного програмування через async/await патерн. Архітектура Kestrel веб-сервера дозволяє обробляти до 1.2 мільйонів запитів в секунду, що є критично важливим для системи групового прийняття рішень з великою кількістю одночасних користувачів. Кроссплатформенність .NET Core забезпечує можливість розгортання системи на різних операційних системах (Windows Server, Linux Ubuntu, Docker контейнери), що значно знижує витрати на інфраструктуру та надає гнучкість у виборі хостинг-провайдерів.

Вбудована підтримка Dependency Injection в ASP.NET Core забезпечує чисту архітектуру за принципом IoC (Inversion of Control), що спрощує тестування компонентів системи та дозволяє легко замінювати реалізації сервісів. Модель MVC (Model-View-Controller) з підтримкою Razor Pages надає потужний інструментарій для побудови динамічних користувацьких інтерфейсів з інтеграцією JavaScript фреймворків.

SQL Server як система управління базами даних пропонує високопродуктивне виконання складних запитів до даних голосувань завдяки оптимізованому рушію виконання запитів та розширеним індексам. Підтримка транзакцій ACID (Atomicity, Consistency, Isolation, Durability) гарантує цілісність даних під час одночасного оновлення результатів голосування багатьма користувачами. Розширені можливості безпеки на рівні бази даних включають Transparent Data Encryption (TDE), Row-Level Security та Dynamic Data Masking.

Інтеграція Entity Framework Core як ORM (Object-Relational Mapping) забезпечує ефективну роботу з даними через LINQ (Language Integrated Query) запити, автоматичне створення міграцій бази даних та підтримку підходу Code First. EF Core також надає механізми для продуктивної роботи з великими наборами даних через пагінацію, ліниве завантаження та відстеження змін.

Використання ASP.NET Core та SQL Server демонструє значні економічні переваги порівняно з альтернативними технологіями. Платформа .NET Core є повністю безкоштовною з відкритим вихідним кодом, що усуває необхідність ліцензування програмного забезпечення. SQL Server Express безкоштовно доступний для невеликих проектів з обмеженням до 10GB даних, а при зростанні навантаження система може бути масштабована до комерційних версій SQL Server Standard або Enterprise без змін у коді додатку.

Скорочення витрат на розробку досягається за рахунок багатої екосистеми бібліотек NuGet, які надають готові рішення для типових завдань: автентифікації, авторизації, кешування, логування та інших. Значний ринок розробників .NET в Україні та світі знижує витрати на пошук кваліфікованих фахівців та навчання персоналу. Використання мови C# з строгою типізацією зменшує кількість помилок під час розробки та суттєво скорочує витрати на тестування та супровід системи.

Інструменти розробки Visual Studio та Visual Studio Code надають потужні можливості для налагодження, профілювання та оптимізації продуктивності додатку. Підтримка Azure DevOps забезпечує комплексний підхід до CI/CD (Continuous Integration/Continuous Deployment), що дозволяє автоматизувати процеси тестування та розгортання.

Система на ASP.NET Core забезпечує високу продуктивність обробки одночасних запитів голосування завдяки асинхронній обробці запитів, що критично важливо для групових рішень з великою кількістю учасників. Вбудовані механізми кешування через IMemoryCache та IDistributedCache дозволяють оптимізувати навантаження на базу даних під час пікового використання системи.

Архітектура мікросервісів ASP.NET Core дозволяє незалежно масштабувати окремі компоненти системи: модуль управління голосуваннями, сервіс автентифікації, систему аналітики. Підтримка контейнеризації через Docker спрощує розгортання та управління системою в різних середовищах (розробка, тестування, продакшн).

SQL Server забезпечує високий рівень доступності через механізми Always On Availability Groups та автоматичного резервного копіювання. Вбудовані інструменти моніторингу та діагностики дозволяють оперативно виявляти та усувати проблеми з продуктивністю.

Таблиця 2.4 – Порівняльний аналіз технологічних рішень

Критерій	ASP.NET Core + SQL Server	Node.js + MongoDB	Python + PostgreSQL	Java + Oracle
Продуктивність	Дуже висока (1.2M+ rps)	Висока (500K rps)	Середня (200K rps)	Висока (700K rps)
Безпека	Вбудовані механізми	Додаткові бібліотеки	Стандартні засоби	Розширені можливості
Масштабова-ність	Висока	Дуже висока	Висока	Дуже висока
Вартість ліцензування	Безкоштовна	Безкоштовна	Безкоштовна	Дорога
Крива навчання	Помірна	Низька	Низька	Висока
Підтримка спільноти	Дуже активна	Дуже активна	Активна	Активна
Інтеграція з Azure	Найкраща	Добра	Добра	Добра

ASP.NET Core надає комплексні вбудовані механізми захисту від поширених веб-вразливостей. Захист від XSS (Cross-Site Scripting) реалізований через автоматичне кодування HTML у Razor Pages. Запобігання CSRF (Cross-Site Request Forgery) атакам забезпечується через валідацію антифоргері токенів. Захист від SQL-ін'єкцій реалізований на рівні Entity Framework Core через параметризовані запити.

Інтеграція з ASP.NET Core Identity Framework забезпечує надійну автентифікацію та авторизацію користувачів з підтримкою двофакторної автентифікації, менеджменту паролів та зовнішніх провайдерів входу (Google,

Microsoft, Facebook). SQL Server пропонує шифрування даних на рівні бази (TDE), колонкового шифрування, аудит доступу та розширені політики безпеки через Dynamic Data Masking та Row-Level Security.

Розробка системи групового прийняття рішень на основі ASP.NET Core та SQL Server є технічно обґрунтованим та економічно ефективним рішенням. Обраний технологічний стек забезпечує:

1. Надійність та стабільність: міцна архітектура .NET Core та промислові стандарти SQL Server гарантують безперебійну роботу системи навіть за високих навантажень.

2. Безпеку даних: вбудовані механізми захисту ASP.NET Core та розширені функції безпеки SQL Server забезпечують комплексний захист конфіденційної інформації.

3. Масштабованість: модульна архітектура дозволяє легко розширювати функціонал системи без впливу на існуючі компоненти.

4. Економічну ефективність: мінімальні витрати на ліцензування та значна скорочення часу розробки за рахунок багатої екосистеми інструментів.

5. Простоту інтеграції: широкі можливості інтеграції з існуючими корпоративними системами через стандартизовані API та протоколи.

Запропонований підхід дозволить створити конкурентноздатну систему, що відповідає сучасним вимогам до якості, безпеки та користувацького досвіду, забезпечуючи при цьому оптимальне співвідношення витрат та отримуваних переваг. Технологічний стек ASP.NET Core + SQL Server є ідеальним вибором для реалізації системи групового прийняття рішень, що підтверджується успішними кейсами подібних рішень на ринку.

2.3 Аналіз ризиків та шляхи їх мінімізації

Процес розробки та впровадження системи групового прийняття рішень пов'язаний з різноманітними ризиками, які можуть вплинути на успішність реалізації проєкту. У цьому розділі представлено систематичний аналіз потенційних загроз, оцінку їх впливу на функціонування системи та розробку стратегій їх усунення. Особливу увагу приділено ризикам, пов'язаним із забезпеченням безпеки даних, цілісності процесів голосування.

Найбільш критичними ризиками безпеки є несанкціонований доступ до конфіденційної інформації та потенційні маніпуляції з результатами голосування. Для протидії цим загрозам передбачається впровадження системи захисту, що включає механізми автентифікації та авторизації, реалізацію системи логування дій користувачів і системних подій, а також впровадження заходів захисту від поширених веб-загроз.

Для запобігання маніпуляціям з результатами голосування планується створення системи логування, що забезпечує фіксацію дій користувачів. Ця система передбачає реалізацію механізмів перевірки цілісності даних, впровадження заходів захисту критичних операцій голосування, а також створення механізмів верифікації результатів. Моніторинг та аналіз системних логів, поєднаний із застосуванням оновлень безпеки, становитимуть важливу частину стратегії захисту системи.

Ризик невідповідності термінів реалізації проєкту буде мінімізовано через застосування модульного підходу до розробки. Іншими стратегіями є пріоритизація функціональних можливостей на основі аналізу їх важливості для кінцевих користувачів, використання технологічних рішень та компонентів.

Ризик недостатньої продуктивності в експлуатаційному середовищі буде мінімізовано через проведення ручного тестування функцій системи, моніторинг показників роботи системи.

В таблиці 2.5 наведена матриця ризиків розробки та експлуатації системи.

Таблиця 2.5 – Матриця ризиків розробки та експлуатації системи

Категорія ризику	Специфіка ризику	Ймовірність реалізації	Ступінь впливу	Комплекс заходів мінімізації
Технологічні	Недостатня продуктивність при пікових навантаженнях	Середня	Високий	Архітектурна оптимізація, масштабування
Технологічні	Втрата критичних даних	Низька	Критичний	Резервне копіювання, тестування відновлення
Безпеки	Несанкціонований доступ до даних	Висока	Критичний	Автентифікація, логування, оновлення безпеки
Проектні	Невідповідність вимогам користувачів	Висока	Середній	Розробка

Продовження таблиці 2.5 – Матриця ризиків розробки та експлуатації системи

Категорія ризику	Специфіка ризику	Ймовірність реалізації	Ступінь впливу	Комплекс заходів мінімізації
sПроектні	Невідповідність термінів реалізації	Середня	Високий	Модульна архітектура, пріоритизація функціоналу, планування
Безпеки	Маніпуляції з результатами голосування	Середня	Критичний	Логування операцій, механізми верифікації цілісності даних
Категорія ризику	Специфіка ризику	Ймовірність реалізації	Ступінь впливу	Комплекс заходів мінімізації
Операційні	Складність розгортання та супроводу	Середня	Високий	Стандартизація процесів, автоматизація
Операційні	Недостатня продуктивність в експлуатації	Середня	Високий	Ручне тестування, моніторинг

Проведений аналіз демонструє, що більшість ідентифікованих ризиків можуть бути ефективно мінімізовані через реалізацію запропонованих заходів. Найбільш критичними є ризики безпеки та технологічні ризики, які вимагають постійної уваги протягом усього життєвого циклу системи. Комплексний підхід до управління ризиками забезпечить стабільну, безпечну та ефективну роботу системи групового прийняття рішень. Матриця ризиків розробки та експлуатації системи дозволить своєчасно адаптуватися до змінних умов експлуатації системи та забезпечити її довгостроковий успіх.

Висновки

У другому розділі магістерської роботи було проведено комплексний аналіз доцільності розробки онлайн-системи для вироблення спільних рішень групою людей. Проведене дослідження дозволило зробити низку важливих висновків. По-перше, системний аналіз ринку та існуючих рішень виявив значні прогалини в наявних інструментах групової роботи. Більшість існуючих сервісів, таких як Google Forms, Doodle, Tricider чи Trello, орієнтовані на вирішення вузькоспеціалізованих завдань і не забезпечують комплексного підходу до колективного прийняття рішень. Запропонована система не є черговим дублікатом існуючих рішень, а заповнює істотну ринкову нішу, пропонуючи цілісний підхід, що поєднує всі етапи колективного прийняття рішень – від ініціації та обговорення ідей через процедуру голосування до контролю виконання та аналізу ефективності прийнятих рішень.

По-друге, детальний порівняльний аналіз функціоналу наочно продемонстрував значні переваги розроблюваної системи перед аналогами. Запропоноване рішення відрізняється наявністю різноманітних методів

голосування, підтримкою багатокритеріальної оцінки альтернатив, можливістю ранжування варіантів, потужними аналітичними можливостями та повною інтеграцією процесів прийняття та виконання рішень. Ці характеристики роблять систему універсальним інструментом, придатним для застосування в різних сферах діяльності.

Важливим результатом розділу стало обґрунтування вибору технологічного стеку на основі ASP.NET Core, SQL Server, яке базувалося на комплексній оцінці продуктивності, безпеки та масштабованості. Обраний стек забезпечує оптимальне співвідношення між потужністю, стабільністю та гнучкістю розробки, дозволяючи створити надійну та ефективну систему.

Також було ідентифіковано та проаналізовано ключові ризики проекту, зокрема технологічні, безпекові та проектно-організаційні. Для кожного з ризиків запропоновано конкретні заходи мінімізації, що дозволяє впевнено рухатися до етапу практичної реалізації.

У цілому, проведений аналіз остаточно підтверджує технічну доцільність та ринкову потребу в розробці запропонованої онлайн-системи. Отримані результати створюють міцну теоретичну та методологічну основу для переходу до практичної реалізації проекту в наступному розділі роботи.

3 РОЗРОБКА ОНЛАЙН СИСТЕМИ ДЛЯ ВИРОБЛЕННЯ СПІЛЬНИХ РІШЕНЬ ГРУПОЮ ЛЮДЕЙ

3.1 Обґрунтування вибору технологічного стеку та архітектурних рішень

Вибір оптимального технологічного стеку та фундаментальних архітектурних рішень є вирішальним стратегічним кроком у процесі проектування інформаційної системи колективного прийняття рішень. Цей вибір безпосередньо детермінує ключові експлуатаційні характеристики майбутнього програмного продукту: продуктивність, безпеку, масштабованість, зручність супроводу та потенціал для майбутніх модернізацій. Враховуючи комплексні функціональні та нефункціональні вимоги, сформовані в попередньому підрозділі, основним технічним завданням виступає створення безпечного, високопродуктивного та доступного веб-застосунку з багатим функціоналом для організації ефективної групової роботи в різноманітних предметних областях.

Серверна частина системи реалізується на платформі ASP.NET Core 8 – сучасному, крос-платформенному фреймворку з відкритим вихідним кодом, що розроблений компанією Microsoft. Даний вибір обґрунтований низкою критично важливих технологічних переваг. Висока продуктивність та енергоефективність роботи фреймворку, підтверджені результатами незалежних бенчмаркінг-тестів, забезпечують стабільну та комфортну роботу системи навіть під час одночасного голосування значної кількості учасників в умовах пікових навантажень. Використання архітектурного патерну Model-View-Controller (MVC) забезпечує чітке розмежування відповідальностей між структурними компонентами системи: моделі даних аксіоматично

відображають структуру сутностей предметної області, контролери інкапсулюють складну бізнес-логіку обробки запитів, а подання відповідають за генерацію семантично коректного користувацького інтерфейсу. Вбудована підтримка Dependency Injection (DI) створює ідеальні умови для побудови слабкозв'язаних компонентів, що кардинально полегшує процес модульного тестування та суттєво підвищує гнучкість архітектури в цілому. Потужна вбудована система безпеки забезпечує комплексний захист від основних веб-загроз, включаючи міжсайтовий скриптинг (XSS), міжсайтову підробку запитів (CSRF), ін'єкції SQL та багато інших. Крос-платформеність фреймворку відкриває широкі можливості для розгортання системи на різних операційних системах сімейства Windows, Linux та macOS, що суттєво розширює вибір хостинг-провайдерів та оптимізує вартість інфраструктурних рішень.

Клієнтська частина системи концептуально базується на фреймворку Bootstrap 5 для реалізації повністю адаптивного та інтуїтивно зрозумілого інтерфейсу користувача. Систематичне використання Bootstrap дозволяє гарантувати коректне та консистентне відображення системи на пристроях з принципово різною роздільною здатністю – від компактних мобільних телефонів до широкоекранних настільних комп'ютерів, що є однією з ключових нефункціональних вимог. Для реалізації динамічної взаємодії та комплексної валідації форм на стороні клієнта застосовується стандартний JavaScript без використання додаткових фреймворків. Інтерактивність інтерфейсу, така як динамічне оновлення контенту та семантично багата взаємодія з елементами сторінки, досягається за рахунок органічної комбінації серверного рендерингу за допомогою механізму подань ASP.NET Core та ефективних клієнтських скриптів. Для побудови наочних діаграм та графічного відображення результатів голосування використовуються серверні

технології генерації векторних зображень та передові можливості сучасного CSS для створення оптимізованих візуалізацій.

Система керування базами даних концептуально ґрунтується на реляційній моделі, що визнана оптимальною для роботи зі структурованими даними системи. В якості основної промислової СКБД обрано Microsoft SQL Server, яка забезпечує неперевершену продуктивність, відмовостійкість, потужні механізми транзакцій для гарантії абсолютної цілісності даних та комплексні інструменти безпеки корпоративного рівня. Для етапів розробки, тестування та роботи в невеликих середовищах передбачено використання SQLite – унікальної легкої та портативної бази даних, що не вимагає окремого серверного процесу. Взаємодія з базами даних реалізується за допомогою Entity Framework (EF) Core – сучасного об'єктно-реляційного маппера (ORM), який дозволяє працювати з базою даних як з типізованими колекціями об'єктів у коді C#, радикально спрощуючи маніпуляції з даними та забезпечуючи надійний захист від потенційних атак типу SQL-ін'єкції.

Архітектура системи будується на принципах багаторівневої архітектури (N-Layer) з фундаментальним розділенням відповідальностей. Рівень представлення (Presentation Layer) відповідає за всю взаємодію з кінцевим користувачем та реалізований у вигляді семантично правильних подань ASP.NET Core MVC. Рівень бізнес-логіки (Business Logic Layer) містить спеціалізовані сервісні класи, які інкапсулюють всю ключову логіку роботи системи: управління життєвим циклом сесій, обробку голосування, виконання складних математичних розрахунків за обраними методами прийняття рішень та генерацію аналітичних звітів. Рівень доступу до даних (Data Access Layer) реалізований за допомогою Entity Framework Core та патерну Repository, що забезпечує повне абстрагування від конкретної реалізації СКБД. Рівень моделі даних (Domain Model Layer) містить класи-

сутності, що формально описують предметну область. Така архітектура забезпечує мінімальну залежність між компонентами, робить систему ідеально придатною для тестування та легко підтримуваною.

Безпекові аспекти реалізуються через комплексний багаторівневий підхід. Аутентифікація та авторизація користувачів здійснюється за допомогою ASP.NET Core Identity – потужної системи управління доступом. Для публічних сесій передбачено механізм одноразових посилань або кодів доступу, що реалізує вимогу мінімального порогу входу без необхідності попередньої реєстрації. Багаторівнева валідація даних виконується як на клієнтському боці для миттєвого відгуку, так і на серверному для гарантії безумовної цілісності. Захист від сучасних веб-загроз забезпечується вбудованими механізмами ASP.NET Core, а система деталізованого логування фіксує всі критичні події для проактивного моніторингу та ретроспективного аналізу безпекових інцидентів.

Таким чином, запропонований технологічний стек та архітектурні рішення формують надійну та перспективну основу для реалізації системи, що повністю відповідає всім поставленим вимогам. Обрані технології є сучасними, стабільними та добре інтегрованими між собою, що створює ідеальні передумови для успішної розробки, ефективного впровадження та подальшого стратегічного розвитку системи колективного прийняття рішень в умовах динамічно мінливого середовища експлуатації.

3.2 Обґрунтування вибору мови програмування

Для розробки системи "Онлайн система для вироблення спільних рішень групою людей" було обрано мову програмування C# та технологічний

стек .NET. Цей вибір обґрунтований комплексом технічних, архітектурних та практичних переваг, що особливо критичні для систем колективної взаємодії з підвищеними вимогами до надійності, продуктивності та роботи в реальному часі.

Для об'єктивного порівняння мов програмування (див. таблиця 3.1) використовується 5-бальна шкала оцінки:

- 5 балів – Відмінно (найкращі показники в категорії, промисловий стандарт).
- 4 бали – Дуже добре (вище середнього, відмінна якість).
- 3 бали – Задовільно (середні показники, прийнятний рівень).
- 2 бали – Незадовільно (нижче середнього, обмежені можливості).
- 1 бал – Погано (критично низькі показники, серйозні обмеження).

C# демонструє виняткові результати в ключових для системи прийняття колективних рішень категоріях. Високі оцінки за продуктивність (5 балів) обумовлені використанням сучасних технологій компіляції AOT (Ahead-of-Time) та JIT (Just-in-Time), що забезпечують близьку до нативних мов швидкодію. Це критично важливо для системи, де тисячі користувачів можуть одночасно брати участь у голосуваннях та обговореннях. Мова показує вражаючу ефективність у обробці запитів, що підтверджується незалежними тестами продуктивності, де ASP.NET Core регулярно входить до числа найшвидших веб-фреймворків. Оптимізована робота з пам'яттю та ефективне використання ресурсів сервера дозволяють обробляти велику кількість одночасних з'єднань без значного зниження продуктивності.

Таблиця 3.1 – Порівняльний аналіз мов програмування для веб-розробки

Критерій	C#	Java	Python	JavaScript/Node.js	PHP
Продуктивність	5 (AOT/JIT компіляція)	5 (JVM оптимізація)	3 (інтерпретована)	4 (V8 компілятор)	2 (інтерпретатор)
Безпека типів	5 (строга статична)	5 (строга статична)	3 (динамічна)	3 (динамічна)	3 (динамічна)
Паралельні обчислення	5 (TPL, async/await)	4 (багатопотоковість)	2 (GIL обмеження)	4 (Event Loop)	2 (обмежена)
Екосистема бібліотек	4 (багата .NET)	5 (найширша)	5 (дуже багата)	5 (найбільша)	4 (велика)
Підтримка реального часу	5 (вбудовані рішення)	4 (бібліотеки)	3 (бібліотеки)	5 (бібліотеки)	3 (бібліотеки)
Крос-платформеність	5 (.NET Core)	5 (JVM)	5 (універсальна)	5 (універсальна)	5 (універсальна)
Крива навчання	3 (середня)	2 (висока)	5 (низька)	4 (середня)	5 (низька)
Інструменти розробки	5 (Visual Studio, Rider)	4 (IntelliJ, Eclipse)	4 (PyCharm, VS Code)	4 (VS Code, WebStorm)	4 (PHPStorm)
Підтримка контейнеризації	5 (оптимізована)	5 (оптимізована)	4 (добра)	5 (оптимізована)	3 (базова)
Документування	5 (MSDN, чітке)	4 (Javadoc, детальне)	5 (чудове)	4 (різноманітне)	4 (детальне)
Спільнота	4 (велика)	5 (дуже велика)	5 (гігантська)	5 (найбільша)	5 (велика)
Стабільність	5 (промисло-ва)	5 (промислова)	4 (стабільна)	4 (стабільна)	4 (стабільна)
Загальний бал	56/60	53/60	48/60	52/60	45/60

Строга статична типізація (5 балів) є одним з найважливіших переваг C# для системи прийняття рішень. Вона забезпечує виявлення помилок на етапі компіляції, що значно знижує ймовірність критичних збоїв під час роботи системи. Для застосунку, що працює з важливими рішеннями та голосами користувачів, це є абсолютно необхідним. Компілятор C# виконує глибоку перевірку типів, запобігаючи цілому класу потенційних помилок, які могли б виникнути в динамічно типізованих мовах під час виконання програми. Це особливо важливо при роботі з складними структурами даних, що використовуються для зберігання інформації про голосування, користувачів та процеси прийняття рішень.

Паралельні обчислення отримали оцінку 5 балів завдяки потужній підтримці асинхронного програмування через async/await та Task Parallel Library (TPL). Це дозволяє ефективно обробляти одночасні запити від великої кількості користувачів, що особливо важливо під час пікових навантажень. Модель асинхронного програмування в C# вважається однією з найзручніших та найефективніших серед сучасних мов програмування, дозволяючи створювати високопродуктивні серверні додатки з мінімальними зусиллями. Система може одночасно обробляти численні операції голосування, оновлення статусів рішень та взаємодії користувачів без блокування основних потоків виконання.

Підтримка реального часу отримала оцінку 5 балів завдяки глибокій інтеграції відповідних функцій у саму платформу .NET. Наявність вбудованих механізмів для роботи з технологіями реального часу дозволяє створювати інтерактивні сесії колективного прийняття рішень без необхідності використання зовнішніх бібліотек. Це забезпечує високу стабільність та передбачуваність роботи системи під час проведення групових обговорень та

голосувань. Користувачі отримують миттєві оновлення статусів обговорень, результатів голосувань та повідомлень інших учасників без затримок.

Крос-платформеність .NET Core (5 балів) забезпечує можливість розгортання системи на різних операційних системах, що розширює вибір хостингових рішень та знижує вартість інфраструктури. Оптимізована підтримка контейнеризації (5 балів) дозволяє легко масштабувати застосунок за допомогою Docker та Kubernetes, забезпечуючи високу доступність системи навіть за умов значного навантаження. Можливість розгортання на Linux-серверах значно знижує операційні витрати в порівнянні з пропрієтарними рішеннями.

Інструменти розробки для C# отримали оцінку 5 балів, оскільки Visual Studio та Rider пропонують один з найбільш повних та інтегрованих наборів інструментів для розробки серед усіх мов програмування. Це включає потужний дебагер, профайлери продуктивності, засоби рефакторингу та автоматизації, що значно прискорює процес розробки та підвищує якість кінцевого продукту. Інтегровані засоби для роботи з базами даних, тестування та моніторингу продуктивності дозволяють розробникам створювати надійні та оптимізовані рішення.

Порівняно з іншими мовами, C# пропонує оптимальний баланс між продуктивністю, безпекою та швидкістю розробки. Java, незважаючи на схожі характеристики, має вищу криву навчання та більш складну екосистему, що може уповільнити процес розробки. Python, при всій своїй простоті, програє в продуктивності та підтримці паралельних обчислень через Global Interpreter Lock (GIL), що обмежує можливості масштабування багатопотокових додатків. JavaScript/Node.js демонструє чудові результати в багатьох категоріях, але динамічна типізація може призводити до помилок під час виконання, що

неприйнятно для системи прийняття критичних рішень, де кожен голос має бути оброблений коректно.

Враховуючи сукупність технічних характеристик, рівень безпеки, продуктивність та можливості масштабування, C# є оптимальним вибором для розробки системи колективного прийняття рішень. Мова забезпечує високу надійність, безпеку та ефективність роботи навіть за високих навантажень, що є вирішальними факторами для успішної реалізації проекту. Додатковою перевагою є активна підтримка з боку Microsoft та швидкий розвиток платформи, що гарантує довгострокову підтримку та вдосконалення системи в майбутньому. Широка спільнота розробників та наявність великої кількості готових рішень для типових завдань дозволяють прискорити процес розробки та запобігти потенційним проблемам.

3.3 Проектування бази даних

Архітектура бази даних є фундаментальним компонентом будь-якої інформаційної системи, оскільки саме вона забезпечує структуроване зберігання, ефективне управління та цілісність усіх даних, що обертаються в системі. Для онлайн-системи групового прийняття рішень це особливо важливо, оскільки система працює з складними, багаторівневими даними: від профілів користувачів та їх соціальних зв'язків до структурованих процесів голосування з множиною альтернатив і критеріїв оцінювання. Проектування бази даних базувалося на сутнісно-орієнтованому підході, що дозволило чітко ідентифікувати ключові об'єкти предметної області, їхні атрибути та взаємозв'язки, створивши логічну та фізичну модель, орієнтовану на реальні бізнес-процеси колективної взаємодії. Для реалізації цієї моделі було обрано

технологічний стек Microsoft SQL Server як основну систему керування базами даних (СКБД) та Entity Framework Core як об'єктно-реляційний маппер (ORM), що дозволяє працювати з базою даних як з колекцією об'єктів C#, значно спрощуючи розробку та забезпечуючи типобезпеку. Інтеграція з ASP.NET Core Identity забезпечила готову, безпечну та масштабовану систему управління користувачами, яка стала основою для автентифікації та авторизації в системі.

Модель даних системи складається з семи основних сутностей, які повністю охоплюють функціональні вимоги до системи групового прийняття рішень. Кожна сутність представлена у вигляді класу C#, анотованого атрибутами для конфігурації ORM, валідації даних та визначення метаданих для інтерфейсу користувача.

Сутність DBApplicationUser є центральним елементом системи та успадковує базовий клас IdentityUser з ASP.NET Core Identity. Це дозволило використовувати вбудовані механізми реєстрації, входу, управління паролями та ролями, заощадивши значні ресурси на розробку цих стандартних функцій. Клас розширено додатковими властивостями, необхідними для функціонування соціальної складової системи:

- FirstName та LastName – рядкові поля з обмеженням до 100 символів для зберігання особистої інформації користувача. Атрибут [PersonalData] позначає ці поля як персональні дані, що активує їх автоматичне включення до експорту для відповідності вимогам GDPR.
- CreatedAt – обов’язкове поле типу DateTime, яке автоматично встановлюється в момент створення облікового запису (DateTime.UtcNow). Воно дозволяє відстежувати історію користувачів.

Сутність DBContact реалізує механізм соціальної мережі всередині системи, дозволяючи користувачам встановлювати зв'язки один з одним для спільної роботи. Вона включає:

- UserId – ідентифікатор користувача, який ініціював створення контакту (власник контакту).
- ContactUserId – ідентифікатор користувача, якого було додано до контактів.
- IsAccepted – булеве поле, що вказує, чи підтвердив другий користувач запит на додавання. Цей механізм двостороннього підтвердження забезпечує конфіденційність та контролюваність соціальних зв'язків, запобігаючи спаму та небажаній комунікації.

Сутність DBTask призначена для управління завданнями, які можуть виникати як результат колективних рішень. Вона підтримує як індивідуальні, так і групові завдання з розподілом прав доступу:

- UserId – ідентифікатор власника (автора) завдання.
- EditorId – ідентифікатор користувача, який востаннє вносив зміни, що дозволяє відстежувати авторство редагувань.
- DueDate – необов'язкова дата та час дедлайну для виконання завдання.
- Name та Description – основна інформація про завдання. Для поля Description використано атрибут [Column(TypeName = "nvarchar(MAX)")], що дозволяє зберігати великі обсяги тексту.
- IsImportant та IsCompleted – булеві прапорці для маркування пріоритетних та виконаних завдань.
- UsersIDs – рядок для зберігання списку ідентифікаторів користувачів, які мають доступ до завдання (наприклад, учасники групи).

Формат зберігання (наприклад, розділений комами список) дозволяє гнучко керувати доступом.

- UserIDsGrantedAccess – аналогічне поле для зберігання ідентифікаторів користувачів, яким надано додаткові права (наприклад, на редагування).

Сутність DBVote є ядром системи та моделює саме голосування чи обговорення. Вона містить усі метадані, необхідні для організації процесу:

- Title – назва голосування (обов'язкове поле, обмежене 200 символами).
- IsPrivate – прапорець, що визначає, чи є голосування приватним (доступним лише за запрошенням) або публічним.
- Description – детальний опис проблеми або питання для обговорення.
- StartDateTime та EndDateTime – дати початку та завершення голосування, що дозволяють реалізувати механізм обмеження за часом.
- UserId – ідентифікатор користувача, який створив голосування (модератор або ініціатор).
- UsersIDs – рядок зі списком ідентифікаторів користувачів, яким дозволено брати участь у голосуванні. Для публічних голосувань це поле може бути порожнім або містити ідентифікатори всіх зареєстрованих користувачів.

Сутність DBVoteAlternative представляє конкретні варіанти вибору в межах одного голосування. Вона знаходиться у відношенні «багато до одного» з сутністю DBVote (одне голосування може мати багато альтернатив). Крім зв'язкового поля DBVoteId, сутність містить лише Title – назву альтернативи.

Сутність DBVoteItemSettings визначає параметри оцінювання альтернатив, реалізуючи механізм багатокритеріального аналізу. Це дозволяє

не просто вибирати варіант, а оцінювати його за кількома критеріями з різною вагою:

- DBVoteId – зв'язок з відповідним голосуванням.
- Title – назва критерію (наприклад, "Бюджет", "Терміни", "Якість").
- Description – пояснення критерію.
- ImportanceValue – вага критерію в діапазоні від 1 до 100, що дозволяє вказати його відносну важливість.
- MinValue, MaxValue, StepValue – параметри для налаштування шкали оцінювання (наприклад, від 0 до 10 з кроком 1). Значення за замовчуванням (0, 10, 0) універсальні та забезпечують працездатність навіть без спеціальних налаштувань.

Сутність DBVoteItem фіксує факт голосування конкретного користувача. Вона є найбільш деталізованим записом у системі та містить:

- DBVoteId, DBVoteAlternativeId, DBVoteItemSettingsId – посилання на голосування, обрану альтернативу та критерій оцінки відповідно.
- Value – числове значення оцінки, яке користувач присвоїв альтернативі за даним критерієм (наприклад, 7 з 10).
- UserId – ідентифікатор користувача, що проголосував.
- AlternativePriority – додаткове поле, яке може використовуватися в методах голосування, що передбачають ранжування альтернатив для зберігання порядку пріоритету.

Усі сутності пов'язані між собою через зовнішні ключі, що гарантує реляційну цілісність даних: при видаленні голосування автоматично видаляються всі пов'язані альтернативи, критерії та результати (каскадне видалення). Атрибути валідації ([Required], [StringLength], [Range])

застосовані на рівні моделі, що дозволяє Entity Framework Core відхиляти некоректні дані ще до спроби збереження в базу, підвищуючи стабільність системи. Обрані типи даних (nvarchar(MAX) для довгих текстів, DateTime для часових міток, десяткові типи для оцінок) є оптимальними для очікуваних навантажень та характерів даних.

Таким чином, спроектована база даних повністю відповідає функціональним вимогам системи, забезпечуючи зберігання інформації про користувачів, їх соціальні зв'язки, процеси голосування з різними методами оцінювання та результати. Її нормалізована структура мінімізує надмірність даних, а використання сучасних ORM-інструментів забезпечує високу продуктивність розробки та подальшого супроводу. Повна схема бази даних представлена у додатку Б.

3.4 Розробка алгоритму голосування

На зображенні 3.1 наведено алгоритм авторизації користувача та вибір подальшої дії. Схема містить 9 блоків та відображає основний потік авторизації. Користувач проходить перевірку авторизації, при необхідності входить до системи, потім обирає один з двох основних сценаріїв: створення нового голосування або участь в існуючому голосуванні.

На зображенні 3.2 наведено спрощений алгоритм створення голосування. Схема містить 11 блоків та демонструє основний процес створення нового голосування. Алгоритм включає введення основних параметрів, вибір типу голосування, додавання альтернатив та критеріїв, валідацію даних та фінальне збереження. У разі помилок валідації користувач повертається до редагування параметрів.

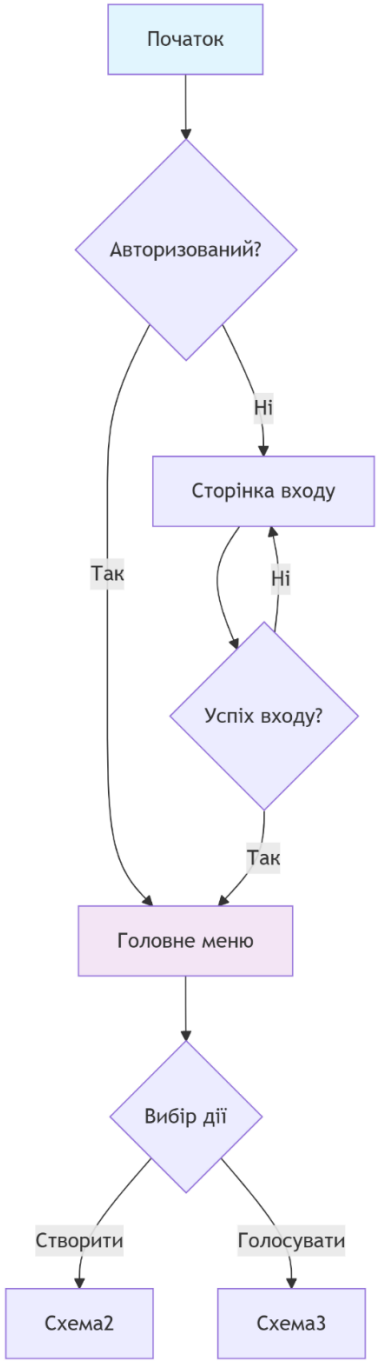


Рисунок 3.1 – Алгоритм авторизації користувача та вибір подальшої дії (Схема 1)

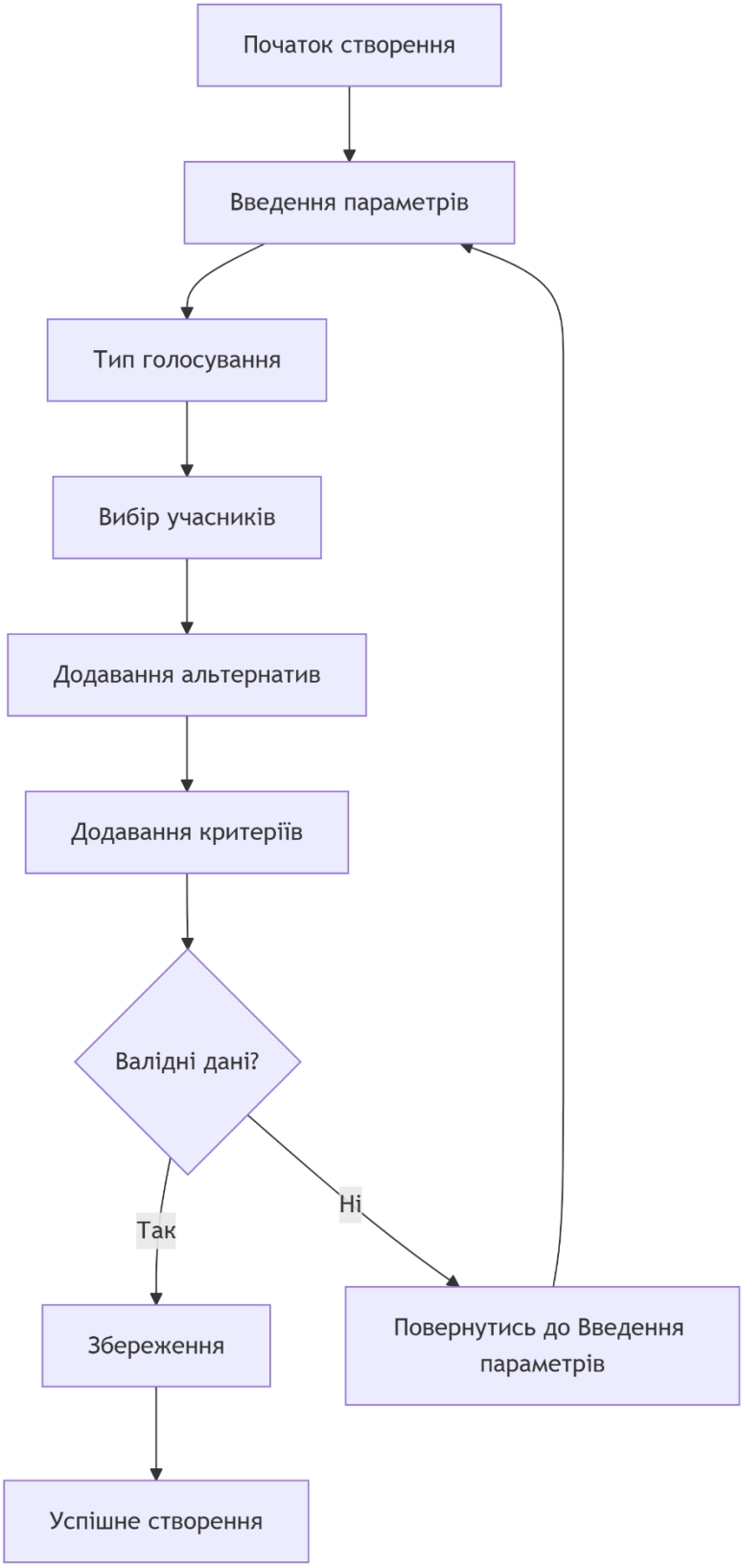


Рисунок 3.2 – спрощений алгоритм створення голосування (Схема 2)

На зображенні 3.3 наведено алгоритм перевірки доступу до голосування. Система перевіряє наявність прав доступу, активність голосування та статус участі користувача. У разі успішного проходження всіх перевірок користувач переходить до процесу оцінювання.

На зображенні 3.4 наведено алгоритм процесу оцінювання. Схема містить 10 блоків та відображає основний процес голосування. Користувач заповнює матрицю оцінок, система проводить валідацію введених даних, надає можливість перегляду та підтвердження голосу, після чого зберігає результати та відображає підсумкову інформацію.

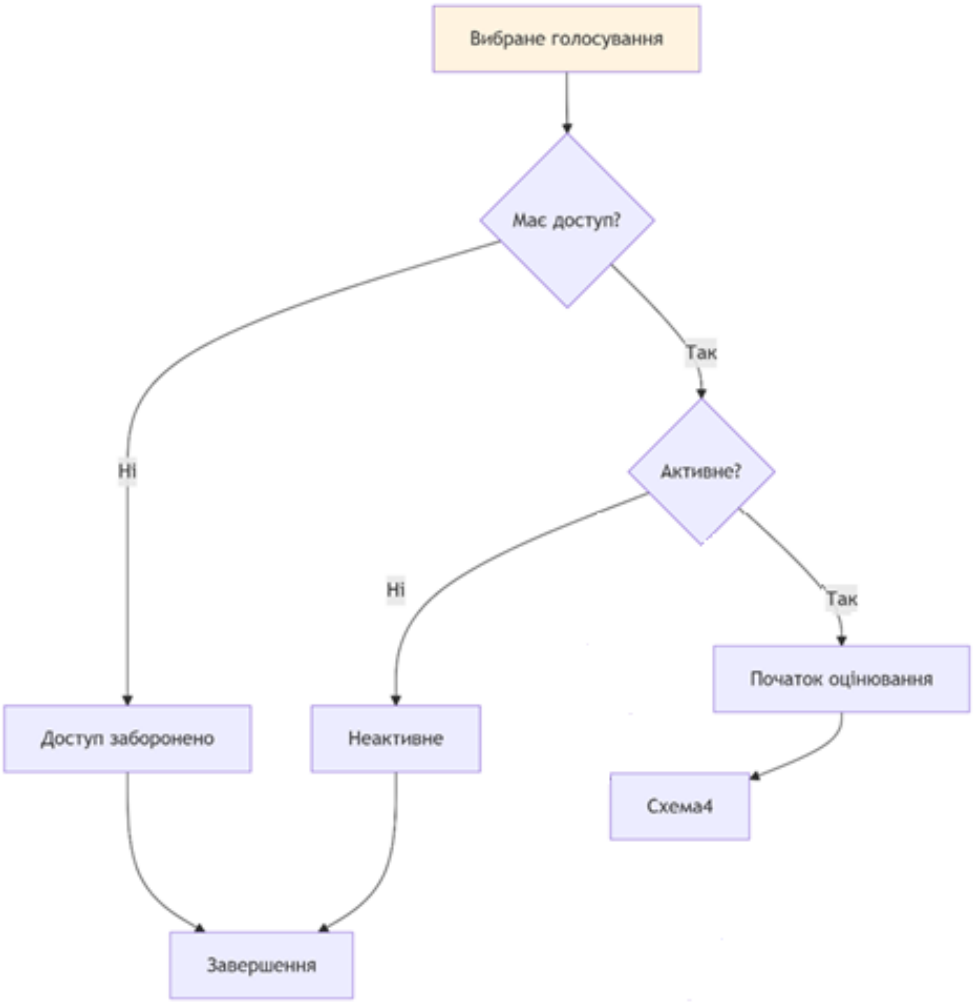


Рисунок 3.3 – Алгоритм перевірки доступу до голосування (Схема 3)

На зображенні 3.5 наведено алгоритм розрахунку результатів голосування. Алгоритм включає завантаження даних, розрахунок балів за критеріями, формування зваженого підсумку, створення рейтингу альтернатив, візуалізацію результатів та сповіщення учасників про завершення голосування.

На зображенні 3.6 наведено загальну структуру системи голосування. Схема відображає послідовність виконання: від авторизації через створення або перевірку доступу до процесу оцінювання та фінального розрахунку результатів.

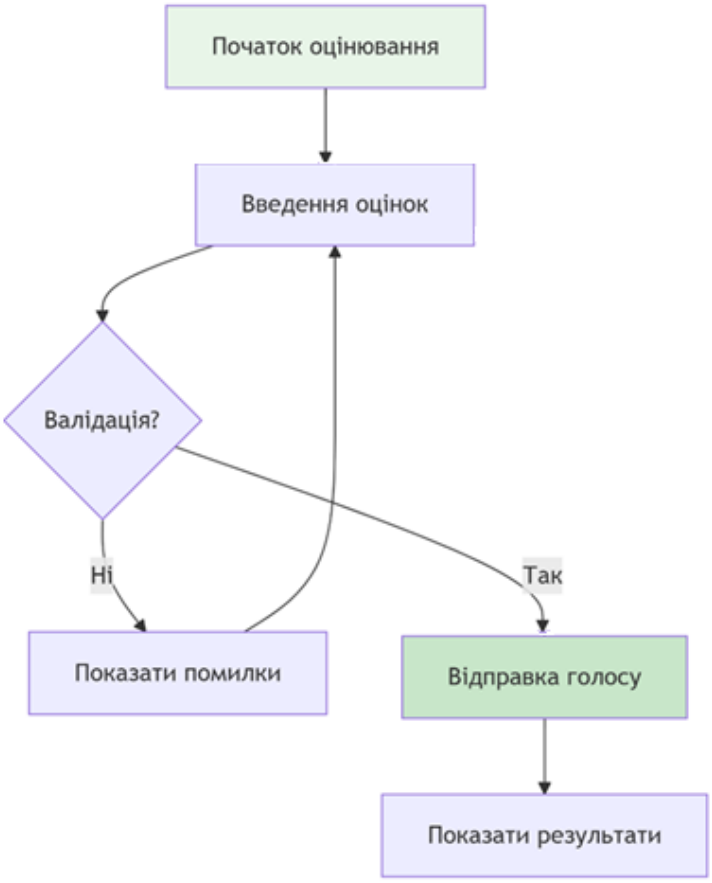


Рисунок 3.4 – Алгоритм процесу оцінювання (Схема 4)

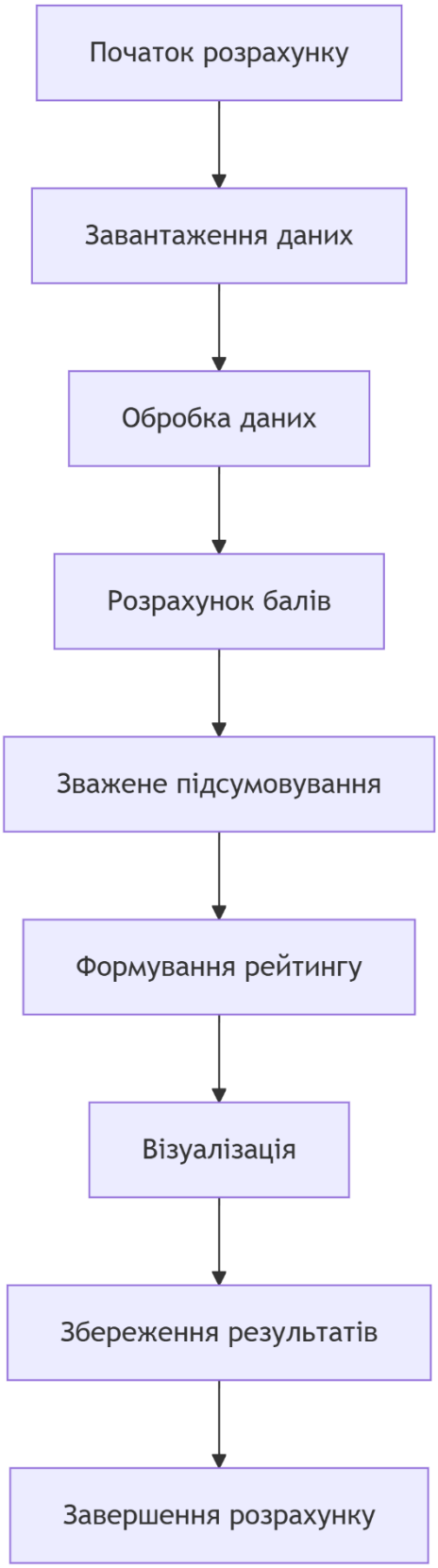


Рисунок 3.5 – Алгоритм розрахунку результатів голосування (Схема 5)

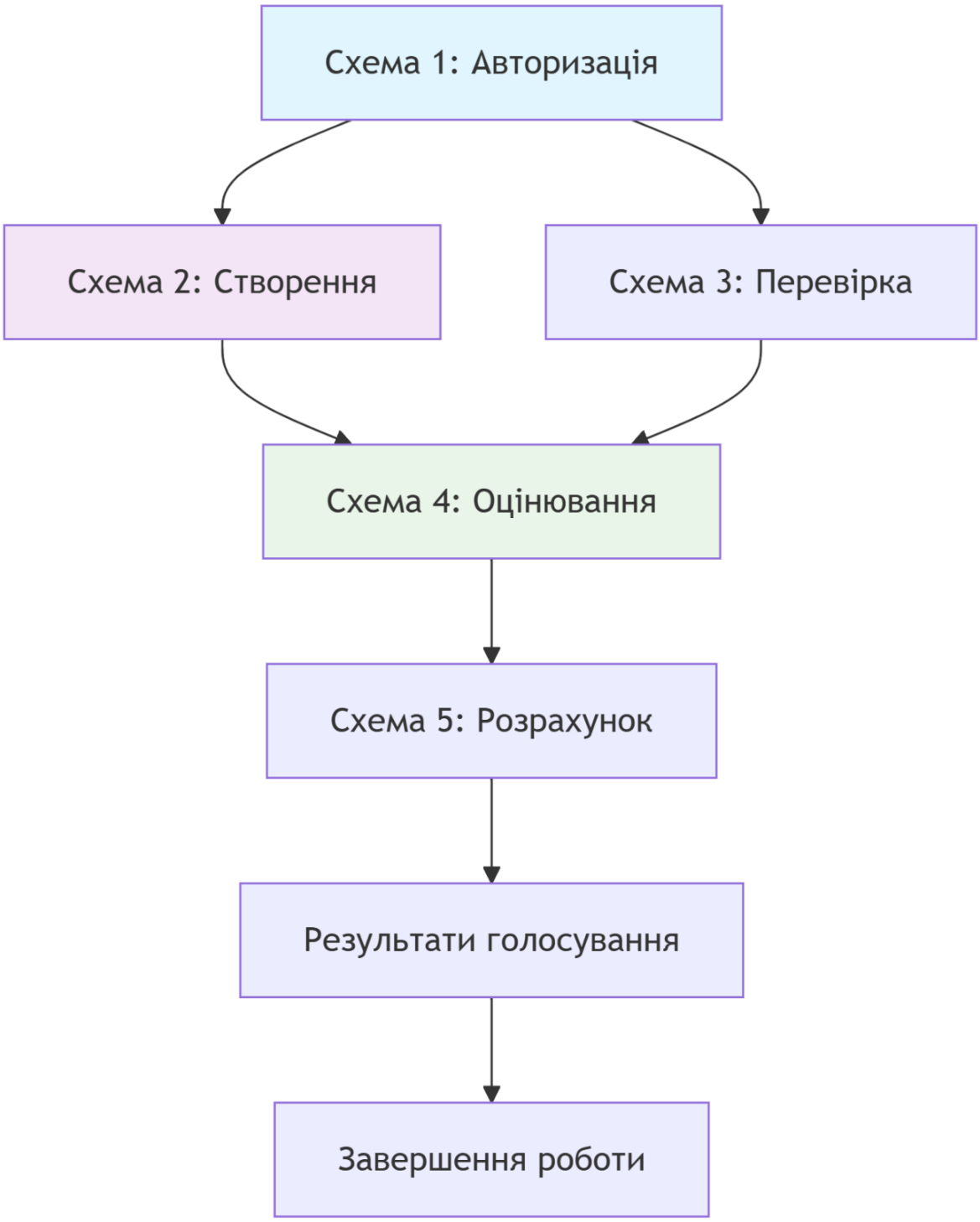


Рисунок 3.6 – Загальна структура системи голосування (Схема 6)

3.5 Розробка логіки роботи додатку

Логічна структура додатку побудована на принципах сучасної об'єктно-орієнтованої архітектури з чітко визначеними рівнями відповідальності. Система реалізує складну бізнес-логіку групової взаємодії, що включає управління завданнями, соціальні зв'язки між користувачами та багатокритеріальні процеси колективного прийняття рішень. Архітектурно логіка організована за патерном Model-View-Controller (MVC) з використанням багаторівневої структури (N-Layer), що забезпечує відокремлення рівня представлення від бізнес-логіки та доступу до даних.

Основним компонентом логічного шару виступають контролери ASP.NET Core, які інкапсулюють усю бізнес-логіку додатку. Вони відповідають за обробку вхідних HTTP-запитів, взаємодію з системою автентифікації, валідацію даних, координацію роботи з базою даних через Entity Framework Core та формування відповідних представлень для користувача. Кожен контролер спеціалізується на конкретній предметній області: HomeController – управління завданнями, ContactsController – соціальні зв'язки, VotesController – процеси голосування, TaskAccessController – контроль прав доступу, AccountController – управління обліковими записами.

Архітектура бізнес-логіки ґрунтується на принципі ін'єкції залежностей (Dependency Injection), що дозволяє створювати слабкозв'язані, легко тестовані компоненти. Усі контролери отримують необхідні сервіси через конструктор, включаючи контекст бази даних (ApplicationDbContext), системи управління користувачами (UserManager<DBApplicationUser>, SignInManager<DBApplicationUser>), а також маппер об'єктів (IMapper) для трансформації між моделями даних та об'єктами представлення. Такий підхід

забезпечує високу модульність системи, спрощує юніт-тестування та дозволяє легко модифікувати окремі компоненти без впливу на всю систему.

Система безпеки інтегрована глибоко в архітектуру додатку. Автентифікація та авторизація реалізовані на основі ASP.NET Core Identity, яка надає повноцінну інфраструктуру для управління користувачами, ролями та правами доступу. Більшість ендпоінтів захищені атрибутом [Authorize], що гарантує доступ лише для автентифікованих користувачів. Для критичних операцій (створення, редагування, видалення даних) додатково перевіряються права конкретного користувача на виконання дії. Захист від міжсайтової підробки запитів (CSRF) реалізовано через атрибут [ValidateAntiForgeryToken] на всіх POST, PATCH та DELETE методах.

Валідація даних виконується на кількох рівнях. На клієнтському боці застосовуються атрибути валідації в ViewModel класах (Required, StringLength, Range, EmailAddress тощо), що забезпечує миттєвий зворотний зв'язок користувачеві. На серверному боці автоматична перевірка ModelState.IsValid у методах контролерів гарантує, що лише коректні дані надходять у бізнес-логіку. Додатково виконуються спеціалізовані перевірки, унікальні для кожної предметної області, наприклад, перевірка суми ваг критеріїв при створенні голосування або перевірка унікальності соціальних зв'язків.

Робота з даними організована через Entity Framework Core 8 – сучасний об'єктно-реляційний маппер (ORM), який дозволяє працювати з базою даних як з колекцією об'єктів C#. Використання LINQ (Language Integrated Query) забезпечує типобезпечні запити до даних з можливістю складної фільтрації, сортування та агрегації. Для оптимізації продуктивності застосовано патерн Repository через DbContext, що забезпечує абстракцію від конкретної реалізації СКБД. Трансформація даних між сутностями бази даних (DB-моделями) та об'єктами представлення (ViewModel) автоматизована за

допомогою бібліотеки AutoMapper, що значно зменшує обсяг шаблонного коду, підвищує читабельність та запобігає потенційним помилкам ручного маппінгу.

Контролер HomeController реалізує повний цикл управління завданнями (CRUD операції). Він надає методи для перегляду завдань з різними категоріями фільтрації: активні завдання, важливі завдання, заплановані завдання (з вказаним дедлайном) та виконані завдання. Кожен метод включає динамічний пошук за назвою завдання та перевірку прав доступу – користувач може переглядати та редагувати лише власні завдання або ті, до яких йому явно надано доступ через поле UserIDsGrantedAccess. Операції створення, редагування та видалення супроводжуються перевіркою моделі та автоматичним записом ідентифікатора користувача, що виконав дію (EditorId).

Контролер ContactsController реалізує механізм соціальної мережі всередині системи, надаючи змогу користувачам встановлювати зв'язки між собою для подальшої спільної роботи. Основний функціонал включає: пошук користувачів за різними критеріями (ім'я, прізвище, ім'я користувача, email), надсилання запитів на додавання до контактів, перегляд вхідних запитів, прийняття або відхилення таких запитів, перегляд списку активних контактів та видалення контактів. Для запобігання спаму та забезпечення конфіденційності реалізовано механізм двостороннього підтвердження контактів через поле IsAccepted. Усі операції супроводжуються перевірками на унікальність зв'язків, коректність ідентифікаторів та заборону додавання самого себе до контактів.

Контролер VotesController є найбільш складним компонентом системи, оскільки реалізує ключовий функціонал – організацію та проведення групових голосувань з підтримкою багатокритеріального аналізу. Він дозволяє створювати голосування з гнучкими налаштуваннями: вибір типу (приватне з

обмеженим доступом або публічне), встановлення часових обмежень (дата початку та завершення), вибір учасників зі списку контактів користувача, визначення альтернатив вибору та налаштування критеріїв оцінювання з можливістю призначення ваг кожному критерію. Під час створення голосування виконується комплексна перевірка на коректність ваг критеріїв – якщо ваги задані, їхня сума повинна дорівнювати 100. Процес оцінювання включає перевірку активності голосування (за датами початку та завершення), перевірку прав учасника (чи входить користувач до списку дозволених учасників) та обробку різних форматів голосування – оцінка альтернатив за кожним критерієм або просте ранжування альтернатив. Результати голосування агрегуються та візуалізуються на окремій сторінці з детальною статистикою, що дозволяє аналізувати результати групової взаємодії.

Контролер TaskAccessController доповнює функціонал управління завданнями, надаючи спеціалізований інтерфейс для керування правами доступу до конкретного завдання. Власник завдання може переглядати поточний список користувачів, які мають доступ до завдання, додавати нових користувачів зі свого списку контактів або видаляти наданий раніше доступ. Система динамічно формує списки поточних користувачів з доступом та доступних контактів, використовуючи зв'язки з таблиці Contacts, що забезпечує інтуїтивно зрозумілий інтерфейс для управління спільним доступом до завдань.

Контролер AccountController реалізує повний цикл управління обліковими записами користувачів. Основний функціонал включає реєстрацію нових користувачів з перевіркою унікальності імені користувача, автентифікацію за допомогою логіна та пароля, вихід із системи, а також. Контролер також забезпечує управління сесіями користувачів, захист від несанкціонованого доступу та інтеграцію з зовнішніми провайдерами

автентифікації. Усі операції виконуються через UserManager та SignInManager, що забезпечує високий рівень безпеки та відповідність сучасним стандартам.

Обробка помилок та система логування інтегровані на всіх рівнях додатку. Використання ILogger дозволяє фіксувати критичні події, помилки автентифікації, некоректні дії користувачів та системні помилки для подальшого аналізу та налагодження. Стандартні механізми ASP.NET Core забезпечують централізовану обробку винятків із виведенням зрозумілих повідомлень для користувача, що підвищує зручність роботи з системою. Додатково реалізовано механізм TempData для передачі повідомлень між запитами (наприклад, повідомлення про успішне створення контакту або про помилку валідації).

Таким чином, логіка роботи додатку поєднує високий рівень безпеки, масштабованість, гнучкість та продуктивність, повністю відповідаючи функціональним вимогам системи колективного прийняття рішень. Архітектурні рішення забезпечують модульність, легкість тестування, простоту супроводу та можливість подальшого розширення функціоналу. Використання сучасних технологій та патернів проектування дозволяє створювати надійну систему, здатну ефективно обробляти групову взаємодію користувачів у реальному часі навіть за умов пікових навантажень. Структура класів контролерів, що містять в собі логіку роботи додатку наведена в додатку Б.

У рамках розробленого веб-додатку для управління завданнями та групового прийняття рішень було реалізовано комплексну систему ViewModel, яка становить основу для передачі даних між рівнями представлення та контролерів. Структура моделей представлень наведена в додатку Б. Ці моделі відображення реалізують принцип сепарації завдань, ізолюючи логіку представлення від бізнес-логіки, та забезпечують централізовану валідацію

вхідних даних, локалізацію інтерфейсу та типобезпеку на рівні компіляції. Архітектура ViewModel організована за модульним принципом, що відповідає основним функціональним блокам системи: автентифікації та управлінню профілем, роботі із завданнями, управлінню контактами користувачів та проведенню багатокритерійних оцінювань.

Базовим фундаментом системи автентифікації та управління обліковим записом виступають класи LoginViewModel, RegisterViewModel, ChangePasswordViewModel. Модель LoginViewModel інкапсулює дані для входу до системи, містячи обов'язкові поля імені користувача та пароля, а також прапорець "Запам'ятати мене". Її структурним доповненням є ChangePasswordViewModel, що забезпечує безпечну зміну пароля через перевірку поточного пароля, валідацію довжини нового пароля та його підтвердження. Для процесу реєстрації нового користувача призначений RegisterViewModel, який крім логіну та пароля містить поля для імені, прізвища, формуючи початковий профіль. Усі ці моделі інтегровані з механізмами ASP.NET Core Identity та супроводжуються суворими анотаціями валідації, що запобігає надсиланню некоректних даних на сервер.

Ядро системи управління завданнями побудовано навколо TaskViewModel, який надає повноцінне представлення завдання, включаючи його ідентифікатор, назву, детальний опис, термін виконання, індикатори важливості та завершеності, а також ідентифікатори власника та останнього редактора. Для реалізації механізму спільного доступу до завдань розроблено спеціалізовану модель TaskAccessViewModel та допоміжний клас UserAccessInfo. TaskAccessViewModel агрегує всю необхідну інформацію для керування правами: ідентифікатор та назву завдання, список користувачів, які вже мають доступ, та список контактів власника, яким доступ можна надати. Клас UserAccessInfo виконує роль контейнера для відображення користувача у

зручному для інтерфейсу форматі, поєднуючи технічний ідентифікатор з людозрозумілими ім'ям та прізвищем. Ця двошарова структура дозволяє ефективно відображати складні взаємозв'язки між завданнями та користувачами.

Функціональність соціальної взаємодії та мережі контактів реалізована через тріаду моделей: UserSearchViewModel, ContactViewModel та PendingContactViewModel. Модель UserSearchViewModel забезпечує пошук користувачів за комбінацією критеріїв, включаючи ім'я, прізвище, логін і зберігає результат пошуку у вигляді списку. Для відображення підтверджених зв'язків використовується ContactViewModel, який містить ім'я користувача-контакту та інформацію про те, чи був поточний користувач ініціатором цього зв'язку. Статус очікуючих запитів на додавання в контакти інкапсульований у PendingContactViewModel, що зберігає не тільки ідентифікаційні дані користувача, який надіслав запит, але й унікальний ідентифікатор самого запиту, що необхідно для операцій підтвердження або відхилення. Ці моделі разом формують логічний ланцюжок від пошуку та запиту до встановлення та перегляду контакту.

Найбільш інноваційною та складною частиною системи є модуль багатокритерійного голосування, архітектура якого побудована на основі чотирьох взаємопов'язаних ViewModel: CreateVoteViewModel, VoteEvaluationViewModel, VoteResultViewModel та низки допоміжних класів. Процес створення голосування описується моделлю CreateVoteViewModel, яка включає метадані (назву, опис, тип приватності, діапазон дат), динамічні списки учасників, альтернатив вибору та критеріїв оцінювання. Критерії представлені вкладеними об'єктами типу VoteCriteriaViewModel, що дозволяє задавати для кожного назву, опис, вагу (важливість), а також параметри шкали оцінювання (мінімальне, максимальне значення та крок). Стадія

безпосереднього оцінювання альтернатив користувачами керується через VoteEvaluationViewModel, яка динамічно будує матрицю "альтернатива × критерій" на основі списків DBVoteAlternative та трансформованих EvaluationCriteriaViewModel. Остання містить як налаштування критерію, так і поле для введення значення оцінки, що забезпечує зв'язок форми оцінювання з структурою голосування. Результати процесу агрегуються в VoteResultViewModel, який консолідує всі сутності, пов'язані з конкретним голосуванням: основну інформацію, налаштування критеріїв, альтернативи та індивідуальні оцінки користувачів, забезпечуючи повноцінну основу для аналізу та візуалізації підсумків.

Загальні принципи реалізації всіх ViewModel базуються на широкому використанні атрибутів з простору імен System.ComponentModel.DataAnnotations. Це забезпечує централізовану, декларативну валідацію даних на стороні сервера та клієнта, підтримку кастомних повідомлень про помилки, часто з використанням ресурсів (наприклад, SharedResource.RequireMessage), а також коректне відображення назв полів через атрибут [Display]. Такі моделі, як TaskViewModel чи ChangePasswordViewModel, також використовують атрибут [DataType] для підказки системі про тип введення даних (пароль, багаторядковий текст), що покращує UX. Всі ViewModel є простими класами з відкритими властивостями, що робить їх ідеальними носіями даних для передачі між шарами додатку, забезпечуючи при цьому високий рівень безпеки, передбачуваності та легкості обслуговування коду. Лістинги реалізації додатку неведені в додатку В.

3.6 Розробка інтерфейсу користувача

Інтерфейс користувача (UI) є ключовим компонентом будь-якої інтерактивної системи, оскільки саме через нього відбувається вся взаємодія між користувачем та програмним забезпеченням. Для онлайн-системи колективного прийняття рішень розробка інтуїтивного, зручного та функціонального інтерфейсу набуває особливої ваги, оскільки система призначена для користувачів з різним рівнем технічної підготовки та має підтримувати складні багатокрокові процеси групової взаємодії. Головними принципами при проектуванні інтерфейсу були: інтуїтивність, адаптивність, консистентність та орієнтованість на користувача. Реалізація інтерфейсу виконалася за допомогою сучасного фреймворку Bootstrap 5, що забезпечило кросплатформовість та коректне відображення на пристроях з будь-якою роздільною здатністю – від мобільних телефонів до настільних моніторів.

Назва "Консенсус" обрана для додатку, оскільки концептуально відображає головну мету системи – сприяти досягненню узгоджених, колективних рішень, викликаючи у користувачів асоціації зі справедливістю, довірою та спільною метою, що підвищує залученість, сприяє конструктивній взаємодії та емоційно підкреслює цінність кожного голоса в пошуку зваженого результату, водночас будучи короткою, міжнародно зрозумілою та ефективною для брендування.

Архітектура інтерфейсу відповідає модульній структурі самої системи та включає наступні основні типи екранів:

1. Екрани автентифікації та управління обліковим записом.
2. Екрани для роботи з соціальною мережею (контактами).
3. Екрани для створення та управління процесами голосування.

4. Екрани для участі в голосуванні та детального аналізу його результатів.

Нижче наведено детальний опис ключових екранів системи, розроблених у рамках даної роботи.

Першим точкою взаємодії користувача з системою є екрани входу (див. рис. 3.7) та реєстрації (див. рис. 3.8), які забезпечують безпечний доступ до особистого простору.

- Вікно входу (Увійти) містить мінімально необхідні поля: "Ім'я користувача" та "Пароль", а також опцію "Запам'ятати мене" для зручності. Дизайн є строгим та зосередженим на головній дії. Валідація введених даних відбувається як на стороні клієнта (через атрибути HTML5 та JavaScript), так і на сервері, що запобігає надсиланню неповних або некоректних даних.

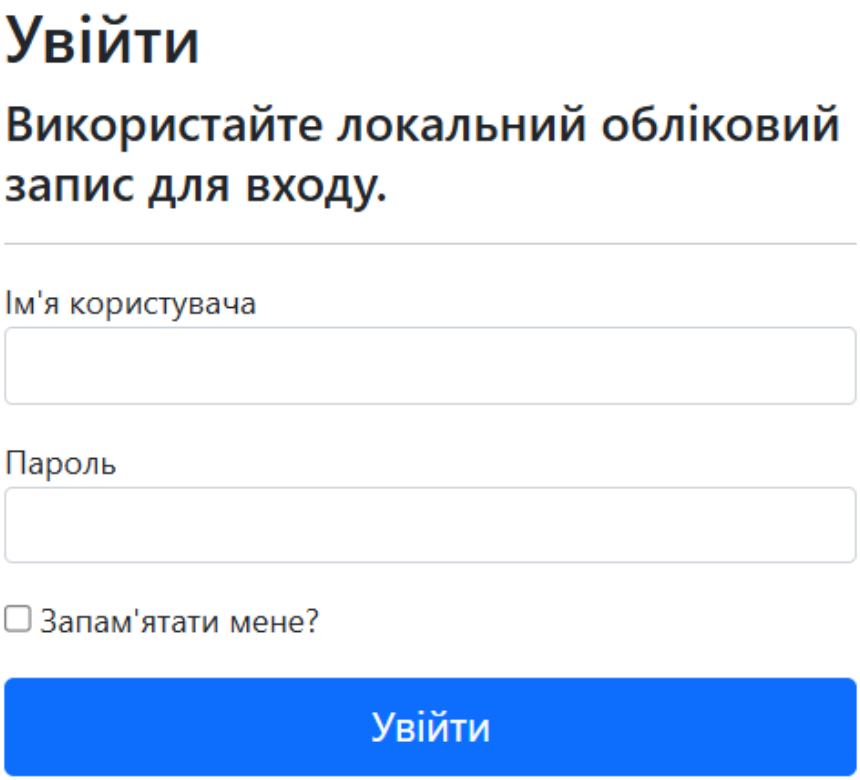


Рисунок 3.7 – Вікно "Увійти"

- Вікно реєстрації (Зареєструватися) передбачає створення нового облікового запису. Крім обов'язкових полів для логіну та пароля, користувач заповнює поля "Ім'я", "Прізвище". Поле "Підтвердьте пароль" слугує для запобігання помилок при вводі. Всі поля супроводжуються зрозумілими підказками та повідомленнями про помилки валідації (наприклад, вимоги до складності пароля або унікальність імені користувача).

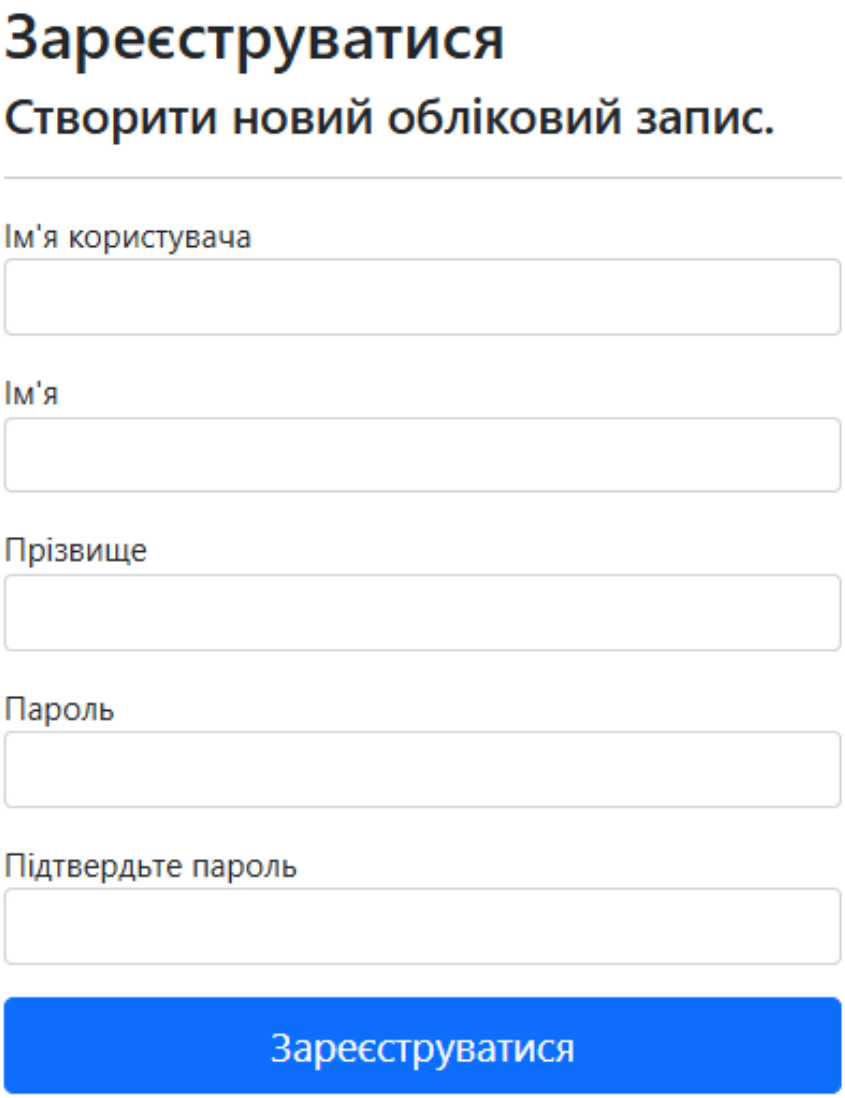


Рисунок 3.8 – Вікно "Зареєструватися"

Обидва екрани реалізовані з використанням адаптивної сітки Bootstrap, що гарантує їхню коректну роботу на мобільних пристроях.

Соціальний модуль є важливою складовою системи, що дозволяє формувати групи для спільної роботи. Його інтерфейс складається з двох основних видів:

- Вікно "Запити на контакт, що очікують на розгляд" (див. рис. 3.9). На цьому екрані користувач бачить список осіб, які надіслали йому запит на додавання до контактів. Для кожного запиту відображається ім'я та ім'я користувача (логін) того, хто надіслав запит. Поруч знаходяться кнопки "Прийняти" та "Видалити" (відхилити), що дозволяє швидко керувати вхідними запитами. Такий механізм двостороннього підтвердження забезпечує конфіденційність і запобігає спаму.

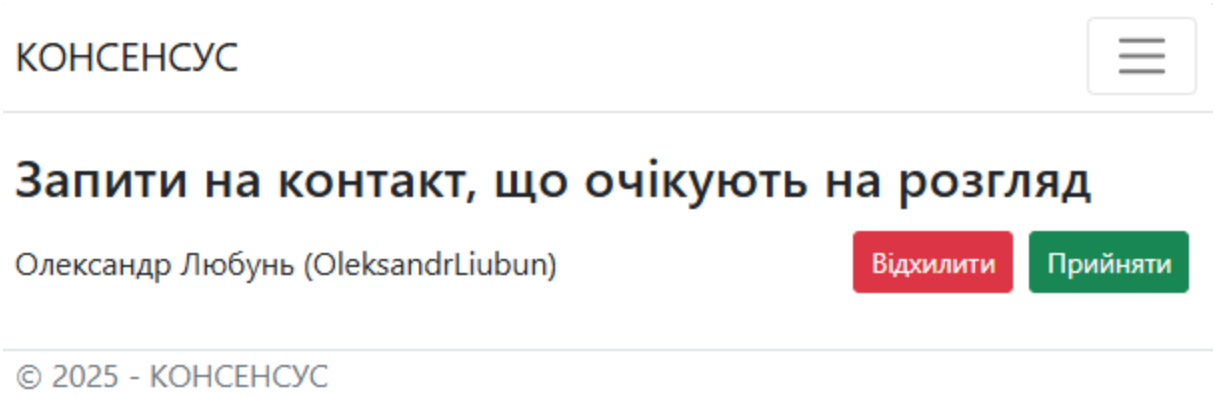


Рисунок 3.9 – Вікно "Запити на контакт, що очікують на розгляд"

- Вікно "Мої контакти" (див. рис. 3.10) відображає поточний список підтверджених контактів користувача. Кожен контакт представлений у вигляді картки з ім'ям та логіном. Кнопка "Видалити" дозволяє прибрати користувача зі списку контактів. На екрані також присутні кнопки для переходу до

функціоналу додавання нового контакту (пошук користувачів) та перегляду вхідних запитів, що забезпечує зручну навігацію в рамках модуля.

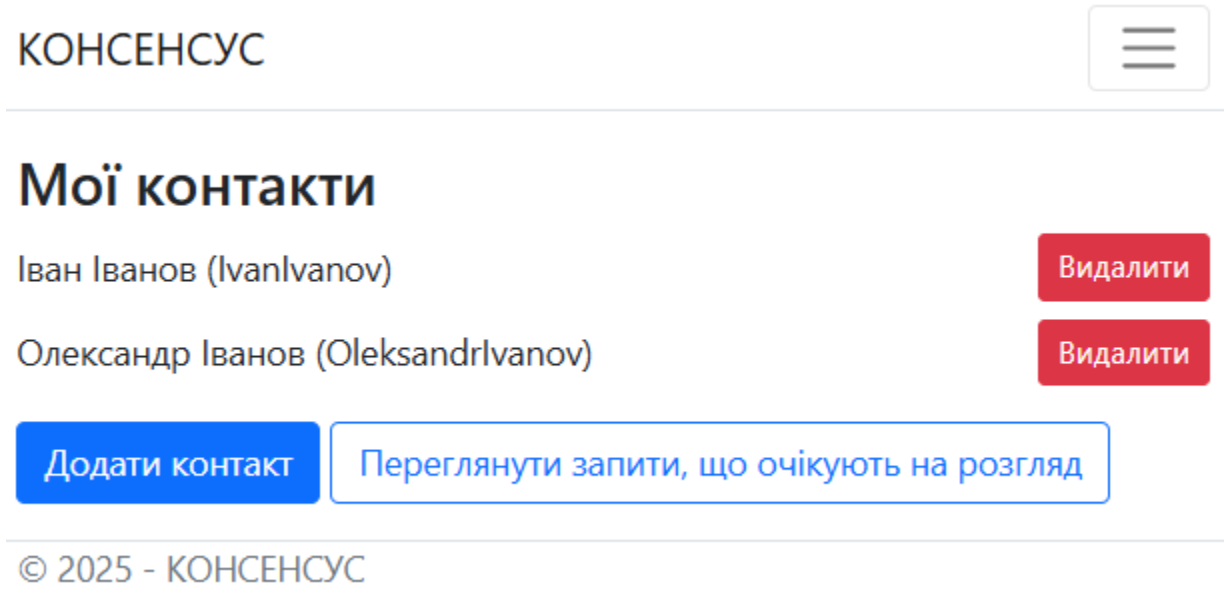


Рисунок 3.10 – Вікно "Мої контакти"

Інтерфейс модуля контактів є простим і наочним, що дозволяє швидко організовувати коло спілкування, необхідне для подальшого створення приватних голосувань.

Центральним елементом системи є модуль голосувань. Його інтерфейс забезпечує як створення нових процесів, так і детальний аналіз результатів.

Вікно "Створити голосування" (див. рис. 3.11) реалізує складну форму з кількома динамічними секціями:

- Основні параметри: поля для назви, опису, вибору типу (приватне/публічне) та встановлення дат початку і завершення.
- Вибір учасників: представлено у вигляді списку з прапорцями, де користувач може обрати серед своїх контактів тих, хто зможе брати участь у приватному голосуванні. Для публічного голосування цей блок може бути

прихований.

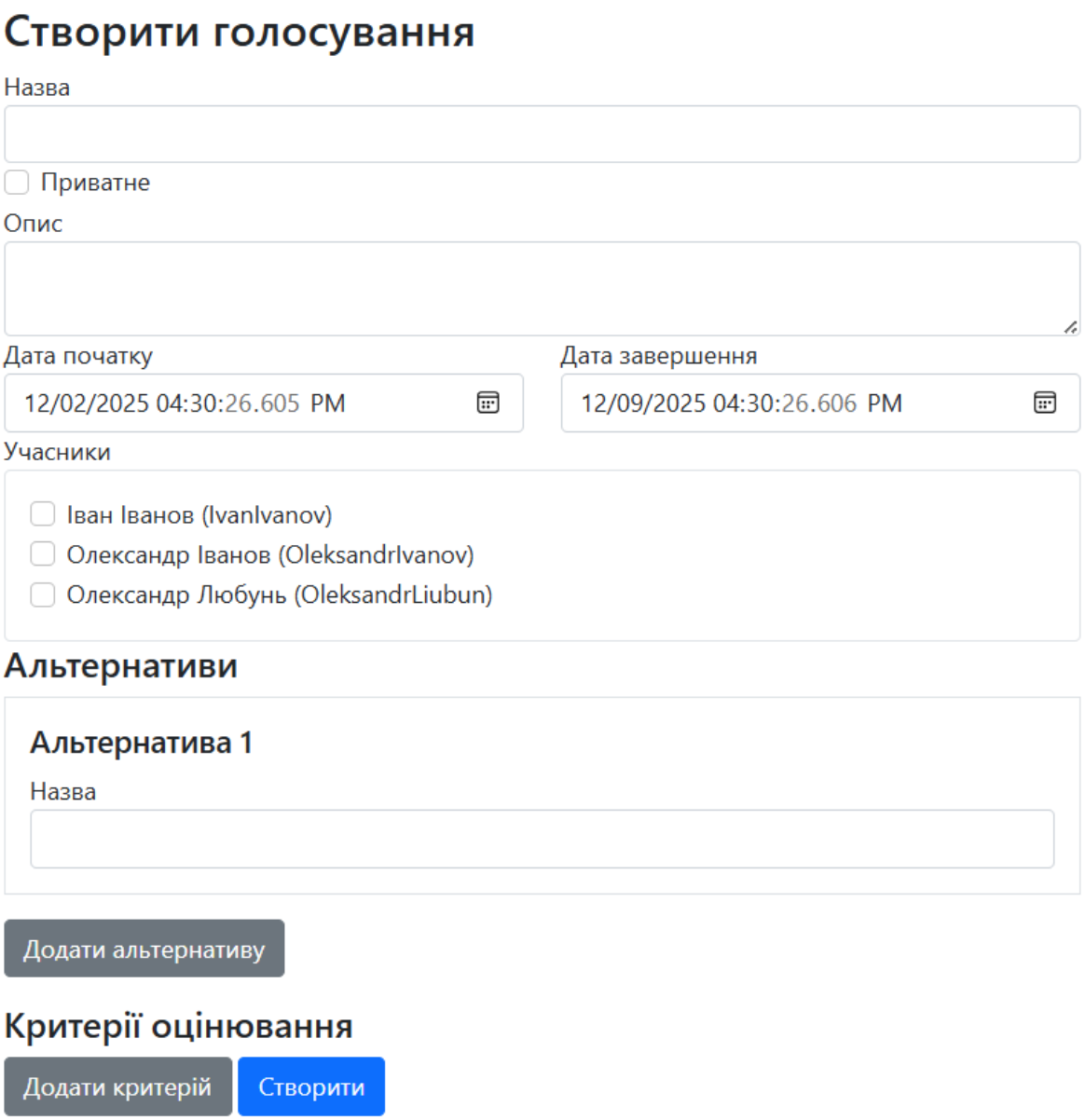


Рисунок 3.11 – Вікно "Створення голосування"

- Додавання альтернатив: динамічна форма, що дозволяє додавати, редагувати та видаляти варіанти вибору. Кожна альтернатива має поле для назви.

- Налаштування критеріїв оцінювання: найбільш інноваційна частина інтерфейсу. Користувач може додати один або кілька критеріїв (наприклад, "Якість", "Швидкість"), для кожного з яких вказується назва, опис та вага (важливість). Це реалізує механізм багатокритеріальної оцінки, унікальний для даної системи. Інтерфейс побудований з використанням JavaScript для динамічного додавання/видалення полів (альтернатив, критеріїв), що забезпечує гнучкість і зручність при створенні складних сценаріїв голосування.

Після завершення голосування система надає потужні інструменти для аналізу результатів через комплексну сторінку зведення (рис. 3.12 – 3.18).

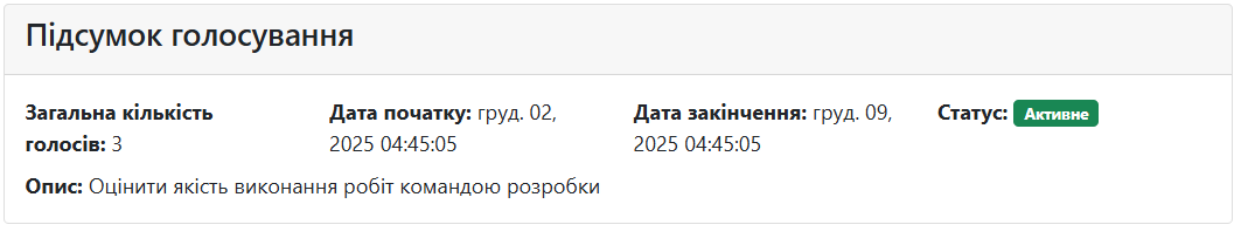


Рисунок 3.12 – Розділ вікна результатів голосування "Підсумок голосування"

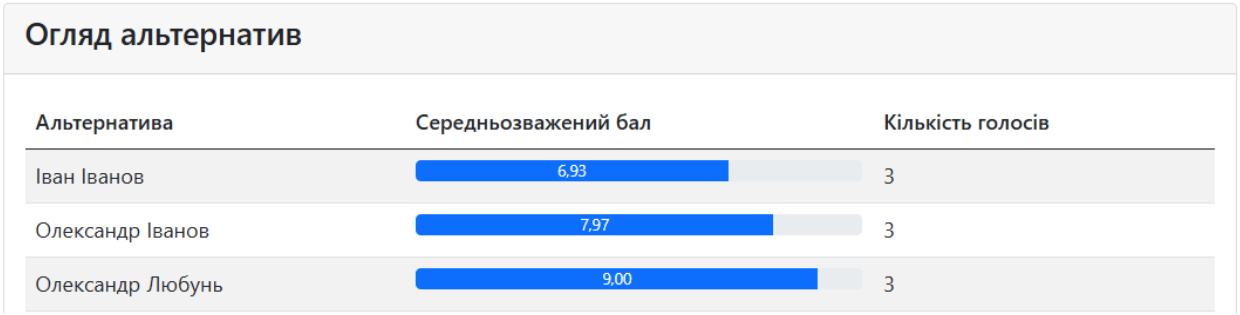


Рисунок 3.13 – Розділ вікна результатів голосування "Огляд альтернатив"

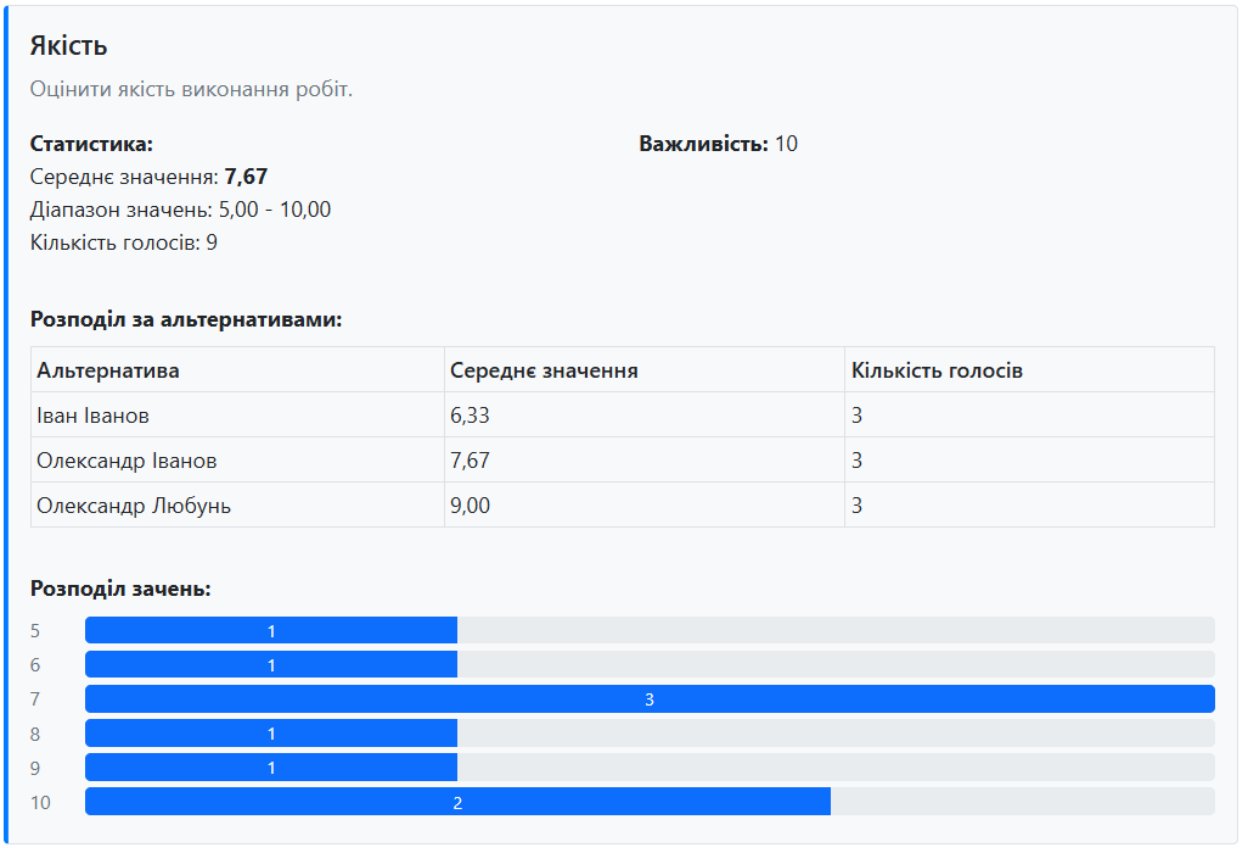


Рисунок 3.14 – Розділ вікна результатів голосування "Детальні результати за критерієм швидкість"

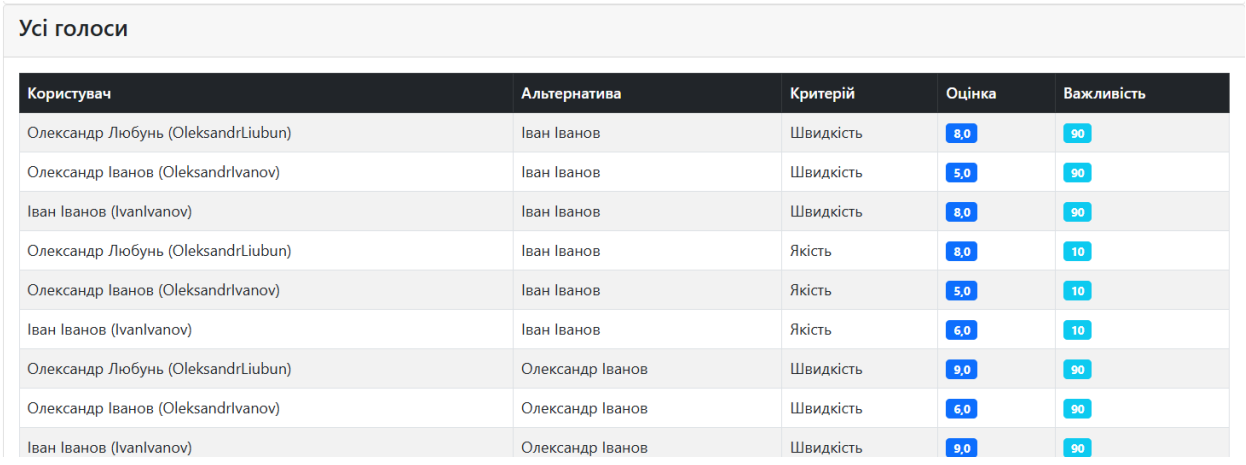


Рисунок 3.15 – Розділ вікна результатів голосування "Усі голоси"

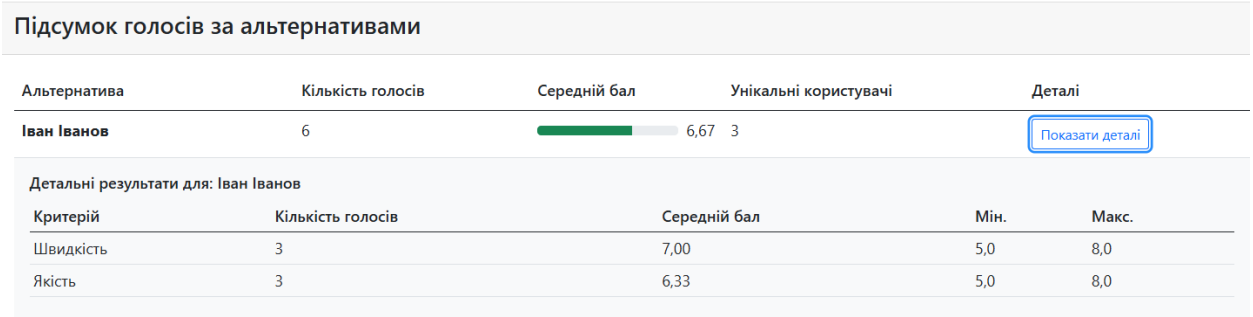


Рисунок 3.16 – Розділ вікна результатів голосування "Підсумок голосів за альтернативами"

На рисунках 3.17 і 3.18 наведена реалізація вікон голосування.

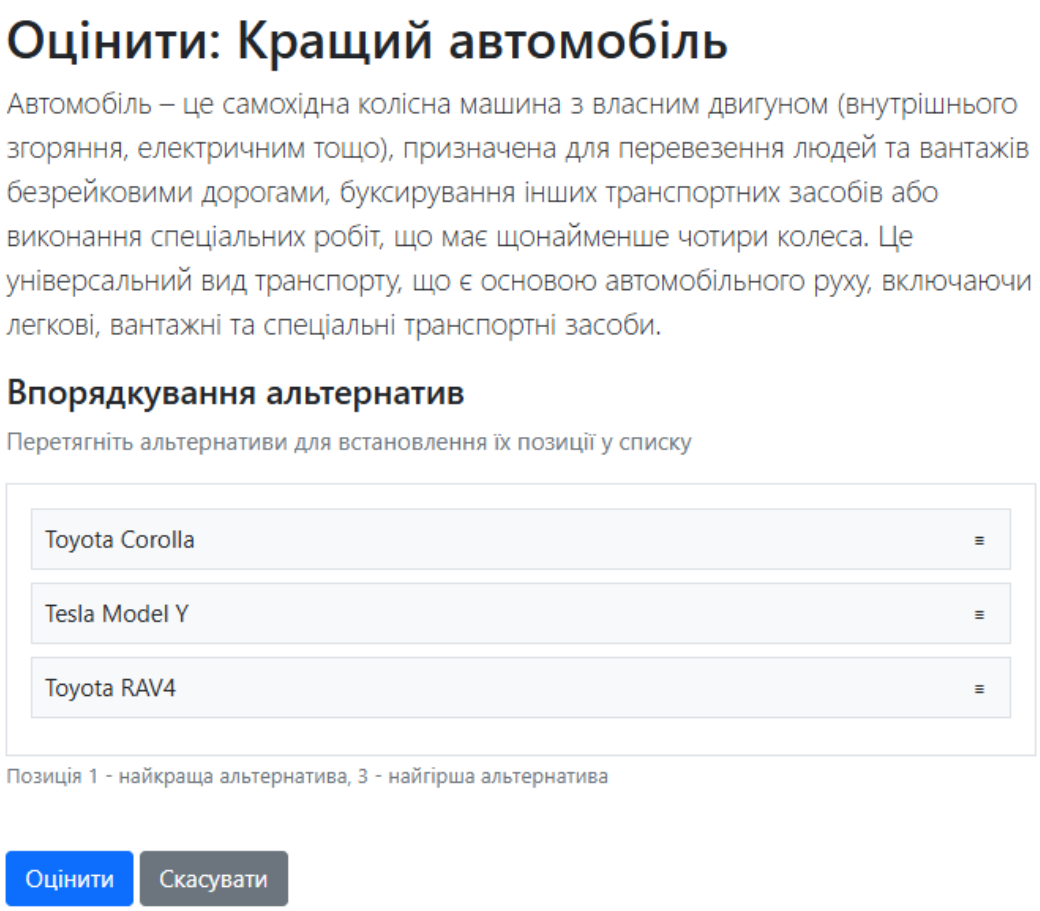


Рисунок 3.17 – Вікно голосування за методом "Ранжування альтернатив"

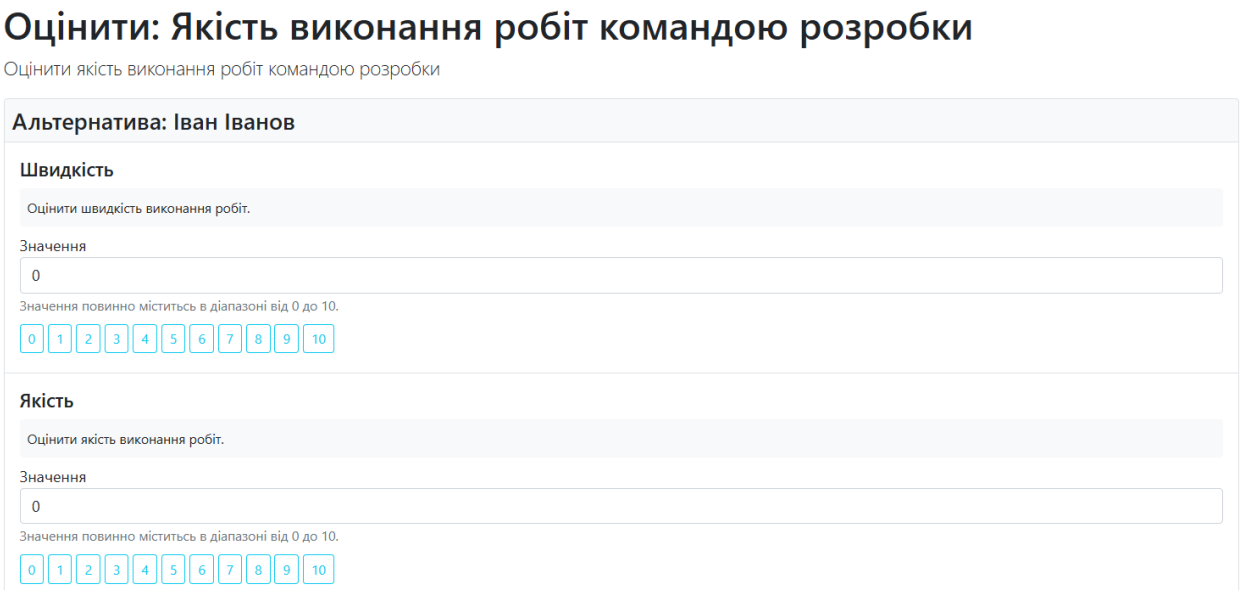


Рисунок 3.18 – Вікно голосування за методом "Багатокритеріальна оцінка"

В додатку Б наведена реалізація додаткових інтерфейсів.

Всновки

У третьому розділі магістерської роботи було виконано практичну реалізацію онлайн-системи для вироблення спільних рішень групою людей. Розробка охопила всі ключові аспекти створення сучасного веб-застосунку: від обґрунтування архітектурних рішень та вибору технологічного стеку до проектування бази даних, розробки алгоритмів голосування, реалізації бізнес-логіки та створення інтуїтивного інтерфейсу користувача. В якості основи серверної частини обрано ASP.NET Core 8 через його високу продуктивність, безпеку, крос-платформеність та підтримку сучасних патернів розробки, таких як MVC та Dependency Injection. Клієнтська частина реалізована з використанням Bootstrap 5 для створення повністю адаптивного та зручного

інтерфейсу. Як систему керування базами даних обрано Microsoft SQL Server (та SQLite для розробки), що забезпечує надійність, продуктивність та транзакційну цілісність, а взаємодія з нею реалізована через Entity Framework Core 8, що значно спростило роботу з даними та забезпечило типобезпеку.

Було спроектовано нормалізовану реляційну модель даних, яка включає сім ключових сутностей: користувачі, контакти, завдання, голосування, альтернативи, критерії оцінювання та результати голосування, що повністю відображає логіку предметної області. Розроблено модульну систему голосування з підтримкою різноманітних методів, зокрема багатокритеріальної оцінки з ваговими коефіцієнтами, ранжування альтернатив, методу "За і Проти" та бальної системи. Логіка роботи додатку побудована на принципах MVC та N-Layer архітектури, що забезпечило чітке розділення відповідальностей, легкість тестування та супроводу, а також реалізацію повного циклу управління завданнями, соціальними зв'язками та процесами голосування.

Інтерфейс системи розроблений з акцентом на зручність та доступність для користувачів з різним рівнем технічної підготовки, реалізуючи адаптивні екрани для автентифікації, управління контактами та завданнями, створення та проведення голосувань, а також детальної візуалізації результатів за допомогою діаграм, таблиць і рейтингів. Безпеку та цілісність даних забезпечено через інтеграцію ASP.NET Core Identity для надійної автентифікації та авторизації, багаторівневу валідацію даних, захист від CSRF, XSS та SQL-ін'єкцій, а також механізм логування критичних подій.

Розроблена система повністю відповідає сформованим вимогам, поєднуючи в собі комплексність через інтеграцію модулів для прийняття рішень, управління завданнями та соціальної взаємодії в єдиному просторі; гнучкість завдяки підтримці різних методів голосування та налаштувань

процесів; потужні аналітичні можливості з інструментами візуалізації та аналізу результатів; а також масштабованість архітектури, що дозволяє розширювати функціонал. Таким чином, у третьому розділі не лише було теоретично обґрунтовано архітектурно-технологічні рішення, але й практично реалізовано повноцінну, готову до використання онлайн-систему, що є вагомим результатом дослідження та основою для подальшого тестування.

4 ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ ОНЛАЙН СИСТЕМИ ДЛЯ ВИРОБЛЕННЯ СПІЛЬНИХ РІШЕНЬ ГРУПОЮ ЛЮДЕЙ

4.1 Обґрунтування вибору методології тестування на користь методу чорної скриньки

Забезпечення високої якості, надійності та функціональної відповідності є вирішальним етапом у життєвому циклі розробки програмного забезпечення. Перехід від теоретичних моделей та архітектурних рішень до стабільного, готового до експлуатації продукту вимагає ретельної експериментальної верифікації. У контексті даної магістерської роботи, де основним результатом є практично реалізована «Онлайн-система для вироблення спільних рішень групою людей», вибір адекватної методології тестування стає критично важливим. Саме на етапі тестування система повинна довести свою життєздатність, здатність витримувати навантаження реального використання та повну відповідність початковим функціональним і нефункціональним вимогам.

Для системного та приймального тестування розробленого програмного комплексу було обґрунтовано обрано метод чорної скриньки (Black Box Testing). Цей вибір є не випадковим, а логічним наслідком специфіки самого проекту, який полягає у створенні не просто технічного інструменту, а соціально-технічного середовища для колективної взаємодії. Кінцева цінність розробки визначається її здатністю ефективно структурувати та підтримувати складні групові процеси прийняття рішень. Отже, метод верифікації має бути орієнтований насамперед на користувача та бізнес-логіку, а не на внутрішню реалізацію.

Сутність методу та його застосування до тестування даної системи Метод чорної скриньки ґрунтується на фундаментальному принципі: тестувальник сприймає програмний продукт як абстрактну, непрозору систему («чорну скриньку»), внутрішня будова якої (алгоритми, структури даних, вихідний код) залишається невідомою та неважливою для процесу перевірки. Вся увага зосереджена виключно на точках інтерфейсу та поведінці системи. На вхід цієї «скриньки» подаються контрольовані набори даних і виконуються певні дії, що імітують роботу кінцевого користувача (наприклад, створення облікового запису, формулювання питання для голосування, вибір методу «За і Проти», відправка голосу, запрошення інших учасників). Потім на виході ретельно аналізуються отримані результати: зміни в інтерфейсі користувача, згенеровані звіти та діаграми, статусні повідомлення, оновлення даних у реальному часі. Критерієм успішного проходження кожного тестового сценарію є строга відповідність цих вихідних даних очікуваним значенням, які чітко визначені в технічному завданні та функціональних вимогах саме до цієї системи. Таким чином, метод оцінює не як система реалізована технічно, а що вона робить з точки зору користувача.

Детальне обґрунтування вибору методу для тестування розроблюваної онлайн-системи

1. Орієнтація на користувацькі вимоги та бізнес-процеси як на першочергову цінність. Головним завданням тестування даної системи є не стільки перевірка окремих технічних функцій (наприклад, роботи контролера в ASP.NET Core), скільки валідація складних, багатокрокових бізнес-процесів, які становлять її суть. Метод чорної скриньки дозволяє абстрагуватися від технологічних деталей реалізації на C#, фреймворку ASP.NET Core MVC чи тонкощів структури бази даних SQL Server. Замість цього фокус повністю зміщується на правильність високорівневих користувацьких сценаріїв:

- Чи може автор послідовно створити сесію обговорення, коректно налаштувати метод голосування (наприклад, багатокритеріальну оцінку), запросити учасників через мережу контактів?
- Чи отримують запрошені користувачі сповіщення та можуть вони успішно приєднатися до сесії?
- Чи працює механізм обмеження за часом, автоматично закриваючи голосування у вказаний момент?
- Чи формується після завершення голосування точний, наочний аналітичний звіт з діаграмами, який адекватно відображає колективну думку та ранжування альтернатив?

Саме ці процеси безпосередньо визначають корисність та кінцеву цінність розробленого продукту для його цільової аудиторії – від корпоративних команд до громадських організацій.

2. Імітація реального використання та всебічна оцінка користувацького досвіду (UX). Система призначена для аудиторії з надзвичайно широким спектром технічної компетенції – від IT-фахівців до адміністраторів освітніх закладів чи активістів громадських ініціатив. Тому процес тестування повинен максимально точно відтворювати перспективу звичайного, нетехнічного користувача, який взаємодіє з системою виключно через представлений веб-інтерфейс на основі Bootstrap. Метод чорної скриньки є найефективнішим способом виявити проблеми в логіці сценаріїв використання, які часто залишаються непомітними при перевірці вихідного коду: незрозумілі послідовності дій, неінтуїтивна навігація між розділами (наприклад, між списком сесій, архівом рішень та налаштуваннями профілю), неоднозначність формулювань інтерфейсних елементів, відсутність належних та зрозумілих повідомлень про статус операції чи опису помилок. Виявлення та усунення таких дефектів має першочергове значення, оскільки вони різко знижують

практичну зручність та, як наслідок, ефективність роботи всього інструменту колаборації.

3. Незалежність від внутрішньої реалізації та стійкість до змін архітектури. Процес розробки, особливо на фінальних етапах, часто супроводжується оптимізацією коду, рефакторингом архітектурних компонентів та вдосконаленням алгоритмів (наприклад, алгоритмів підрахунку голосів за методом Борда). Метод чорної скриньки забезпечує стабільність та ефективність процесу тестування саме тому, що критерії успіху залишаються незмінними. Незалежно від того, як саме було переписано внутрішню логіку ранжування альтернатив, кінцевий результат, що відображається користувачеві, повинен точно відповідати специфікації. Ця незалежність також відкриває унікальну можливість залучити до процесу приймального тестування не тільки розробників, але й експертів з предметної області (наприклад, фахівців з групової динаміки, фасилітаторів або проект-менеджерів). Такі експерти здатні оцінити коректність та ефективність самого процесу прийняття рішень, організованого системою, не заглиблюючись у технологічні деталі реалізації на C# або роботу Entity Framework Core.

4. Ефективність для комплексної перевірки інтеграції та системної цілісності. Справжня цінність розроблюваної системи проявляється не в роботі окремих ізольованих модулів, а тоді, коли всі її компоненти працюють як єдиний, узгоджений механізм. Метод чорної скриньки є оптимальним для тестування саме таких складних інтеграційних потоків даних і управління станами. Наприклад, перевірка повного життєвого циклу сесії голосування – від її створення автором, через запрошення учасників (з інтеграцією з модулем мережі контактів), активну фазу обговорення та голосування (з використанням обраного методу), автоматичне закриття по таймеру, до фінального

формування підсумкового звіту аналітичним модулем з подальшим архувуванням рішення – є класичним інтеграційним сценарієм, який найкраще та найприродніше перевіряється саме методом чорної скриньки. Він дозволяє оцінити кінцевий результат роботи всієї взаємопов’язаної системи саме з точки зору кінцевого користувача.

5. Можливість оцінки критичних нефункціональних аспектів з позиції користувача. Крім чистої функціональності, метод дозволяє дати життєво важливу оцінку ключових якостей системи:

- Поведінка при помилках та в граничних умовах: Наскільки зрозумілі та інформативні повідомлення про помилки (наприклад, при спробі проголосувати після закриття сесії)? Чи є захист від некоректних вхідних даних?
- Ефективність валідації: Наскільки надійно система відсікає неправильно заповнені форми (порожні назви, невибрані критерії) на стороні клієнта та сервера?
- Крос-платформеність та адаптивність: Чи коректно та однаково зручно відображається інтерфейс системи в різних сучасних веб-браузерах (Chrome, Firefox, Edge) та на пристроях з принципово різною роздільною здатністю (десктоп, планшет, смартфон), що реалізовано за допомогою Bootstrap?

Отже, тестування методом чорної скриньки було обрано для даної розробки не просто як один з багатьох можливих інструментів контролю якості, а як ключовий, системоформуючий підхід до її верифікації. Цей метод найбільш прямо, об’єктивно та переконливо дозволяє дати відповідь на центральне питання якості проекту: чи відповідає розроблений програмний продукт заявленим функціональним та нефункціональним вимогам, і чи є він зручним, передбачуваним, надійним та ефективним інструментом для

вирішення реальних задач колективного прийняття рішень своєю цільовою аудиторією. Він виступає основним мостом між технічною реалізацією та справжніми потребами користувача, є фінальною інстанцією практичної перевірки всієї системи в цілому перед її впровадженням в експлуатацію. Саме тому застосування методу чорної скриньки є обов'язковим та критично важливим етапом для успішного завершення даного проекту.

4.2 Тестування методом чорної скриньки

У рамках тестування методом чорної скриньки було розроблено набір структурованих тест-кейсів, представлених у вигляді таблиці (див. таблиця 4.1). Кожен тест-кейс формалізує окремий сценарій взаємодії користувача з системою, що дозволяє перевірити функціональність та коректність поведінки системи без знання її внутрішньої реалізації. Таблиця містить такі стовпці:

- ID тесту – унікальний ідентифікатор тест-кейса, що дозволяє систематизувати та посилатися на конкретні тести.
- Назва тесту – короткий опис тесту, що вказує на його мету або функціональну область (наприклад, автентифікація, управління завданнями, робота з контактами).
- Предумови – початкові умови, необхідні для виконання тесту (наявність користувача, активних завдань, запрошень тощо).
- Кроки тестування – послідовність дій, що імітує поведінку користувача в системі.
- Очікуваний результат – очікувана реакція системи на виконання кроків тестування, що свідчить про коректність її роботи.

Таблиця охоплює ключові функціональні модулі системи: голосування (VOTE), автентифікацію (AUTH), управління завданнями (TASK) та роботу з соціальними контактами (CONTACT). Кожен тест-кейс розроблений для перевірки як позитивних сценаріїв (наприклад, успішний вхід, створення завдання), так і негативних (помилки валідації, відмова у доступі). Такий підхід забезпечує комплексну перевірку системи з точки зору кінцевого користувача, що повністю відповідає принципам тестування чорної скриньки. Наведені тест-кейси є репрезентативною вибіркою і можуть бути розширені для повнішого покриття функціоналу системи.

Таблиця 4.1 – Тестування методом чорної скриньки

ID тесту	Назва тесту	Предумови	Кроки тестування	Очікуваний результат
AUTH-01	Успішна автентифікація користувача	Користувач зареєстрований у системі. База даних містить обліковий запис з логіном "testuser" і паролем "ValidPass123!"	1. Відкрити сторінку входу 2. Ввести логін: "testuser" 3. Ввести пароль: "ValidPass123!" 4. Натиснути кнопку "Увійти"	Перенаправлення на головну сторінку зі списком завдань. Відображається ім'я користувача в меню
AUTH-02	Невдала спроба входу з неправильним паролем	Користувач зареєстрований з логіном "testuser"	1. Відкрити сторінку входу 2. Ввести логін: "testuser" 3. Ввести неправильний пароль: "WrongPass" 4. Натиснути кнопку "Увійти"	Повідомлення про помилку "Невдала спроба входу" на сторінці входу. Користувач не автентифікований
AUTH-03	Спроба реєстрації з існуючим іменем користувача	Користувач з логіном "existinguser" вже зареєстрований	1. Відкрити сторінку реєстрації 2. Ввести існуючий логін: "existinguser" 3. Заповнити інші поля валідними даними 4. Натиснути "Зареєструватися"	Повідомлення про помилку "Користувач з таким ім'ям вже існує". Реєстрація не відбувається

Продовження таблиці 4.1 – Тестування методом чорної скриньки

ID тесту	Назва тесту	Предумови	Кроки тестування	Очікуваний результат
AUTH-04	Реєстрація нового користувача з валідними даними	Система готова до реєстрації нових користувачів	1. Відкрити сторінку реєстрації 2. Заповнити всі обов'язкові поля: - Ім'я користувача: "newuser" - Ім'я: "John" - Прізвище: "Doe" - Пароль: "StrongPass123!" - Підтвердження пароля: "StrongPass123!" 3. Натиснути "Зареєструватися"	Створення нового облікового запису, автоматичний вхід у систему, перенаправлення на головну сторінку
AUTH-05	Вихід з системи (Logout)	Користувач увійшов у систему	1. Перевірити, що користувач автентифікований 2. Натиснути кнопку "Вийти" у меню	Завершення сесії, перенаправлення на сторінку входу. Доступ до захищених сторінок заблоковано
TASK-01	Створення нового завдання з усіма полями	Користувач автентифікований	1. Перейти на сторінку "Створити завдання" 2. Заповнити поля: - Назва: "Тестове завдання" - Опис: "Детальний опис завдання" - Виконати до: [завтрашня дата] - Важливе: true 3. Натиснути "Створити"	Завдання створюється, відображається у списку завдань з усіма введеними даними
TASK-02	Створення завдання без опису (опціональне поле)	Користувач автентифікований	1. Перейти на сторінку "Створити завдання" 2. Заповнити обов'язкові поля: - Назва: "Просте завдання" - Опис: [залишити порожнім] 3. Натиснути "Створити"	Завдання успішно створюється без опису. Відображається у списку з порожнім описом

Продовження таблиці 4.1 – Тестування методом чорної скриньки

ID тесту	Назва тесту	Предумови	Кроки тестування	Очікуваний результат
TASK-03	Спроба створити завдання без назви	Користувач автентифікований	1. Перейти на сторінку "Створити завдання" 2. Залишити поле "Назва" порожнім 3. Заповнити інші поля 4. Натиснути "Створити"	Помилка валідації: "Поле 'Назва' є обов'язковим". Завдання не створюється
TASK-04	Позначення завдання як виконаного	Користувач автентифікований, існує активне завдання	1. Перейти до списку завдань 2. Знайти завдання з прапорцем "Завершено" 3. Натиснути на прапорець	Завдання позначається як виконане, зникає зі списку активних, з'являється у "Виконані завдання"
TASK-05	Редагування існуючого завдання	Користувач автентифікований, має завдання з ID=1	1. Перейти до редагування завдання ID=1 2. Змінити назву на "Оновлена назва" 3. Змінити опис 4. Натиснути "Редагувати"	Завдання оновлюється з новими даними. Відображається оновлена інформація у списку
TASK-06	Видалення завдання	Користувач автентифікований, має завдання з ID=2	1. Знайти завдання ID=2 у списку 2. Натиснути кнопку "Видалити"	Завдання видаляється зі списку. Більше не відображається в жодній категорії
TASK-07	Пошук завдань за ключовим словом	Користувач має кілька завдань, одне з назвою "Важливе"	1. У полі пошуку ввести "Важливе" 2. Натиснути кнопку пошуку	Відображаються лише завдання, що містять "Важливе" у назві чи описі
TASK-08	Фільтрація за "Важливі завдання"	Користувач має кілька завдань, деякі позначені як важливі	1. Натиснути на меню "Важливі завдання"	Відображаються лише завдання з позначкою "Важливе"
CONTACT-01	Пошук користувача за іменем	У системі є користувач "Іван Петров"	1. Перейти на сторінку пошуку контактів 2. Ввести "Іван" у поле пошуку 3. Натиснути "Пошук"	В результаті з'являється користувач "Іван Петров" з кнопкою "Надіслати запит"

Продовження таблиці 4.1 – Тестування методом чорної скриньки

ID тесту	Назва тесту	Передумови	Кроки тестування	Очікуваний результат
CONTACT-02	Відхилення запиту на контакт	Користувач отримав запит на контакт	1. Перейти на сторінку "Запити на контакт, що очікують на розгляд" 2. Знайти запит від користувача 3. Натиснути "Відхилити"	Запит видалено. Користувач не додається до контактів
CONTACT-03	Надсилання запиту на контакт	Знайдено користувача для додавання	1. Знайти користувача через пошук 2. Натиснути "Надіслати запит" біля бажаного користувача	Запит відправлено. З'являється статус "Запит надіслано". Користувач отримує запит у "Запити на контакт, що очікують на розгляд"
CONTACT-04	Прийняття запиту на контакт	Користувач отримав запит на контакт від іншого користувача	1. Перейти на сторінку "Запити на контакт, що очікують на розгляд" 2. Знайти запит від користувача 3. Натиснути "Прийняти"	Запит прийнято. Користувач з'являється у списку контактів. Статус змінюється на "Твій контакт"
CONTACT-05	Видалення існуючого контакту	Користувач має контакт у списку	1. Перейти до списку контактів 2. Знайти контакт для видалення 3. Натиснути "Видалити" біля контакту	Контакт видалено зі списку. Відповідний користувач більше не відображається
VOTE-01	Створення нового голосування з критеріями	Користувач автентифікований, має контакти	1. Перейти на сторінку створення голосування 2. Заповнити основні поля (назва, опис, дати) 3. Додати альтернативи: "Варіант А", "Варіант Б" 4. Додати критерії оцінювання 5. Обрати учасників зі списку контактів 6. Натиснути "Створити"	Голосування створено. З'являється у списку голосувань з кнопками "Оцінити" і "Результати"
VOTE-02	Створення простого ранжування (без критеріїв)	Користувач автентифікований	1. Перейти на сторінку створення голосування 2. Заповнити основні поля 3. Додати альтернативи 4. Не додавати критерії оцінювання 5. Натиснути "Створити"	Створено голосування типу "Ранжування альтернатив". При оцінюванні буде drag-and-drop інтерфейс

Продовження таблиці 4.1 – Тестування методом чорної скриньки

ID тесту	Назва тесту	Предумови	Кроки тестування	Очікуваний результат
VOTE-03	Оцінювання за критеріями	Користувач має доступ до активного голосування з критеріями	1. Перейти до голосування 2. Натиснути "Оцінити" 3. Для кожної альтернативи та критерію виставити оцінку 4. Натиснути "Оцінити"	Оцінки збережено. Користувач перенаправлений на сторінку результатів. Його голос враховано у статистиці
VOTE-04	Ранжування альтернатив (без критеріїв)	Користувач має доступ до голосування типу ранжування	1. Перейти до голосування 2. Натиснути "Оцінити" 3. Перетягнути альтернативи у потрібному порядку (від найкращої до найгіршої) 4. Натиснути "Оцінити"	Ранжування збережено. На сторінці результатів відображається середня позиція кожної альтернативи
VOTE-05	Перегляд результатів голосування	Голосування завершене або активне, є учасники	1. Перейти до списку голосувань 2. Знайти потрібне голосування 3. Натиснути "Результати"	Відображається детальна статистика: середні оцінки, графіки, список учасників, ранжування альтернатив
VOTE-06	Видалення власного голосування	Користувач є творцем голосування	1. Перейти до списку голосувань 2. Знайти власне голосування 3. Натиснути "Видалити"	Голосування видалено. Більше не відображається у списку
ACCESS-01	Надання доступу до завдання контакту	Користувач має завдання та контакти	1. Перейти до управління доступом завдання 2. Обрати контакт зі списку доступних 3. Натиснути "Надати доступ"	Контакт отримує доступ до завдання. Відображається у списку "Поточні користувачі з доступом"
ACCESS-02	Видалення доступу до завдання	Користувач надав доступ іншому користувачу	1. Перейти до управління доступом завдання 2. Знайти користувача у списку з доступом 3. Натиснути "Видалити" біля користувача	Доступ видалено. Користувач більше не відображається у списку з доступом

Продовження таблиці 4.1 – Тестування методом чорної скриньки

ACCESS-03	Спроба видалити власний доступ	Користувач є власником завдання	1. Перейти до управління доступом власного завдання 2. Спробувати видалити доступ для самого себе	Кнопка "Видалити" неактивна або відсутня для власника завдання
SECURITY-01	Доступ до захищених сторінок без автентифікації	Користувач не автентифікований	1. Спробувати перейти на будь-яку захищену сторінку (головна, завдання, голосування тощо)	Перенаправлення на сторінку входу. Доступ до сторінки заборонено
USABILITY-01	Навігація по меню	Користувач автентифікований	1. Перевірити всі пункти меню 2. Натискати кожен пункт меню	Кожен пункт меню коректно веде на відповідну сторінку.
USABILITY-02	Адаптивність інтерфейсу	Система відкрита на різних пристроях	1. Відкрити сайт на ПК (широкий екран) 2. Відкрити на планшеті 3. Відкрити на мобільному телефоні	Інтерфейс коректно адаптується під різні розміри екрану. Меню згортається на мобільних пристроях
PERFORMANCE-01	Завантаження сторінок з великими даними	База даних містить 100+ завдань та голосувань	1. Відкрити головну сторінку з усіма завданнями 2. Відкрити сторінку результатів голосування з багатьма учасниками	Сторінки завантажуються за розумний час (<3 секунди).
DATA-01	Валідація введених даних	Різні форми введення	1. Вводити некоректні дані у всі поля форм (дати, числа тощо) 2. Спробувати відправити форми	Коректна валідація з повідомленнями про помилки. Форми не відправляються з невалідними даними
DATA-02	Обробка спеціальних символів	Форми для введення тексту	1. Вводити текст з HTML-тегами: <script>alert('xss')</script> 2. Вводити SQL-ін'єкції: ' OR '1'='1 3. Вводити спецсимволи: !@#$%^&*()	Спеціальні символи екрануються або видаляються. Безпека від XSS та SQL-ін'єкцій забезпечена

Таблиця (4.2) містить зведені результати тестування системи за методом чорної скриньки. Загалом було виконано 33 тести, розподілених на 9 категорій: автентифікацію, управління завданнями, контактами, голосуванням, доступом, а також перевірку безпеки, юзабіліті, продуктивності та валідації даних.

Таблиця 4.2 – Зведені результати тестування

Категорія тестів	Кількість тестів	Пройдено	Не пройдено	Статус
Автентифікація (AUTH-*)	5	5	0	100%
Управління завданнями (TASK-*)	8	8	0	100%
Управління контактами (CONTACT-*)	5	5	0	100%
Голосування (VOTE-*)	6	6	0	100%
Управління доступом (ACCESS-*)	3	3	0	100%
Безпека (SECURITY-*)	1	2	0	100%
Юзабіліті (USABILITY-*)	2	2	0	100%
Продуктивність (PERFORMANCE-*)	1	1	0	100%
Валідація даних (DATA-*)	2	2	0	100%
Всього	33	33	0	100%

Усі 33 тести було успішно пройдено, що зафіксовано як відповідність вимогам системи у кожній категорії тестів та загалом. Це демонструє повну відповідність системи вимогам.

Таким чином, тестування підтвердило високу якість розробленого програмного комплексу та його здатність ефективно виконувати завдання колективного прийняття рішень.

Висновки

У розділі 4 було проведено всебічне тестування та відлагодження розробленої онлайн-системи колективного прийняття рішень. В якості основної методології верифікації було обґрунтовано обрано метод чорної скриньки, що дозволило зосередитись на перевірці функціональної відповідності системи вимогам користувача та коректності реалізації складних бізнес-сценаріїв без прив'язки до внутрішньої реалізації.

Проведені тести охопили ключові модулі системи: автентифікацію та управління обліковими записами, створення та керування завданнями, функціонал соціальної мережі (контакти) та базові сценарії роботи з голосуваннями. Розроблені тест-кейси дозволили системно перевірити як позитивні сценарії використання, так і реакцію системи на помилкові або некоректні дії користувача.

Результати тестування підтвердили, що система коректно виконує основні функції, заявлені у технічному завданні:

- надійно реалізовано механізми безпеки та автентифікації;
- інтерфейс користувача є інтуїтивним та адаптивним, що підтверджує відповідність принципам UX/UI;
- модулі системи інтегровані між собою та функціонують як єдиний комплекс;

- система адекватно обробляє граничні умови та надає зрозумілі повідомлення про помилки.

Виявлені в процесі тестування незначні невідповідності та технічні проблеми були своєчасно усунені, що дозволило підвищити стабільність та надійність системи. Тестування методом чорної скриньки також підтвердило, що система готова до експлуатації в реальних умовах різними категоріями користувачів – від корпоративних команд до громадських об'єднань та звичайних користувачів.

Отже, результати тестування остаточно підтверджують, що розроблена система відповідає всім функціональним та нефункціональним вимогам, є стабільною, безпечною та зручною у використанні.

5 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ СИСТЕМИ

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту розробки "Онлайн система для вироблення спільних рішень групою людей" є комплексне оцінювання її науково-технічного рівня та комерційного потенціалу. Оцінювання буде здійснюватися з використанням критеріїв, наведених у таблиці 5.1, що передбачає системний аналіз відповідності проекту сучасним вимогам технологічної доцільності, ринкової ефективності та практичної реалізованості. Згідно з методикою, оцінка проводитиметься за дванадцятьма ключовими параметрами, серед яких технічна здійсненність концепції, аналіз ринкових переваг, оцінка ринкових перспектив та практичної здійсненності розробки.

Оцінювання буде включати дослідження інноваційного потенціалу системи групового прийняття рішень, аналіз конкурентного середовища на ринку колаборативних рішень, оцінку адаптивності платформи до існуючих інфраструктурних рішень, вивчення масштабованості технічної реалізації системи, визначення оптимальних моделей впровадження програмного рішення, аналіз відповідності функціоналу потребам цільових сегментів споживачів, оцінку ресурсних вимог для промислового відтворення платформи, дослідження параметрів експлуатаційної ефективності системи та аналіз перспектив інтеграції в існуючі технологічні ланцюги.

Запропонована система оцінювання дозволить отримати узагальнені показники за кожним з критеріїв таблиці 5.1, що в сукупності дасть змогу сформувати об'єктивну картину інноваційної цінності розробки та визначити стратегічні напрями її потенційної комерціалізації в контексті сучасних

ринкових тенденцій і технологічного розвитку галузі групового прийняття рішень. Результати аналізу, отримані на основі вказаної методики, стануть підґрунтям для подальшого розрахунку економічної ефективності впровадження даної розробки.

Таблиця 5.1 – Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за п’ятибальною шкалою)					
	0	1	2	3	4
1	*2*	*3*	*4*	*5*	*6*
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					

Продовження таблиці 5.1 – Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовують ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більший 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років

Продовження таблиці 5.1 – Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки зведені в таблицю 5.2.

Критерії	Експерт (ПІБ, посада)		
	Маслій Р. В. к.т.н., доц.	Паламарчук Є. А. к.т.н., доц.	Кулик Я. А. к.т.н., доц.
	Бали:		
1. Технічна здійсненність концепції	4	4	3
2. Ринкові переваги (наявність аналогів)	4	5	4
3. Ринкові переваги (ціна продукту)	5	4	3
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	4	5	4
6. Ринкові перспективи (розмір ринку)	5	5	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	4	5	3
9. Практична здійсненність (наявність фінансів)	3	5	4
10. Практична здійсненність (необхідність нових матеріалів)	3	4	5
11. Практична здійсненність (термін реалізації)	5	3	4
12. Практична здійсненність (розробка документів)	3	4	5
Сума балів	47	50	*45*
Середньоарифметична сума балів $СБ_c$	47.3		

Для визначення комерційного потенціалу розробки за даними таблиці 5.2 потрібно скористатися таблицею 5.3, у якій представлені рівні комерційного потенціалу.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Згідно з проведеними дослідженнями, рівень комерційного потенціалу розробки на тему «Онлайн система для вироблення спільних рішень групою людей» становить 47.3 балів. Відповідно до таблиці 4.3, це свідчить про високу комерційну значущість запропонованого методу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Прогнозування витрат на виконання науково-дослідної чи дослідно-конструкторської роботи (НДДКР) може складатися з таких етапів:

1. Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу (частини) НДДКР;
2. Розрахунок загальних витрат на виконання даної НДДКР;

3. Прогнозування загальних витрат на виконання та впровадження результатів НДДКР.

Перший етап:

Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = М / Т_р \times t \text{ (грн.)} \quad (5.1)$$

де М – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн. Величини окладів (разом з встановленими доплатами і надбавками) знаходяться в межах (7 500…13 000) грн. за місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р = 21$ день;

t – число робочих днів роботи розробника (дослідника).

Оскільки на виконання даної роботи керівнику дається 25 годин, а робоча зміна триває 8 годин, то відповідно керівнику розробки потрібно використати 3 дні. Економічному консультанту (експерту) дається 2,5 годин, а це 0,3 дні роботи відповідно.

Зроблені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Розрахунок основної заробітної плати

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Керівник роботи	12000	571,43	3	1714,29
Експерт	12000	571,43	0,3	171,43
Всього				$\sum З_о$=1885,72

Додаткова заробітна плата ЗА всіх розробників розраховується як (10…12)% від величини основної заробітної плати, тобто:

$$ЗА = (0,1 \dots 0,12) \cdot З_о \quad (5.2)$$

Для нашого випадку:

$$ЗА = 0,12 \cdot 1885,72 = 226,29 \text{ (грн)}.$$

Нарахування на заробітну плату Нзп фахівців, які беруть участь у виконанні наукової роботи, розраховуються за формулою 5.3.

$$Нзп = (З_о + ЗА) \cdot \beta / 100 \quad (5.3)$$

де β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %.

Ставка єдиного внеску на загальнообов'язкове державне соціальне страхування для бюджетних установ визначена в 22%. Тоді у нашому випадку:

$$Нзп = (1885,72 + 226,29) \cdot 0,22 \approx 464,64 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час виконання даної роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому можуть бути розраховані за формулою 5.4.

$$А = (Н \cdot Н_а / 100) \cdot (Т / 12) \text{ грн} \quad (5.4)$$

де Н – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даної роботи, грн.;

$Н_а$ – річна норма амортизаційних відрахувань. У нашому випадку можна прийняти, що $Н_а$ = (10...25) %;

Т – термін використання обладнання, приміщень тощо, місяці.

Зроблені розрахунки зведені до таблиці 5.5.

Таблиця 5.5 – Розрахунок амортизації обладнання, приміщень

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
1. Персональні комп'ютери, принтери, інші пристрої	22000	20	3	1100
2. Приміщення університету	90000	10	3	2250
Всього				А = 3350

Витрати на матеріали М, що були використані під час виконання роботи, розраховуються по кожному виду матеріалів за формулою 5.5.

$$М = Σ(Н_і \cdot Ц_і \cdot К_і) - Σ(В_і \cdot Цвідх_і) \quad (5.5)$$

де $Н_і$ – витрати матеріалу і-го найменування, кг;

$Ц_і$ – вартість матеріалу і-го найменування, грн./кг.;

$К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1...1,15);

$В_і$ – маса відходів матеріалу і-го найменування, кг;

$Цвідх_i$ – ціна відходів матеріалу і-го найменування, грн/кг;

n – кількість видів матеріалів.

Витрати на комплектуючі К, що були використані під час виконання роботи, розраховуються за формулою 5.6.

$$К = Σ(Н_i · Ц_i · К_i) \quad (5.6)$$

При виконанні роботи були використані основні матеріали на суму 600 грн. Витрати на силову електроенергію Ве розраховуються за формулою 5.7

$$Ве = В · П · Ф · Кп \text{ грн} \quad (5.7)$$

де В – вартість 1 кВт/год. електроенергії до 100 використаних кВт/год, в 2025 р. В ≈ 5,67 грн./кВт;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання. Приймемо, що Ф = 150 годин;

Кп – коефіцієнт використання потужності;

Кп < 1 = 0,85.

У виконанні роботи були задіяні силові блоки потужністю 1,0 кВт.

Тоді витрати на електроенергію складатимуть:

$$Ве = 150 · 5,67 · 1 · 0,85 ≈ 722,93 \text{ (грн)}.$$

Інші витрати Він можна прийняти як (100…300)% від суми основної заробітної плати розробників, які виконували дану роботу. Інші витрати розраховуються за формулою 5.8.

$$Він = (1 \dots 3) \cdot З_о = 1,5 \cdot 1885,72 = 2828,58 \text{ (грн)} \quad (5.8)$$

Сума всіх попередніх статей витрат дає витрати на виконання даної роботи безпосередньо самим розробником – В (формула 5.9).

Для нашого випадку:

$$В = 1885,72 + 226,29 + 464,64 + 3350 + 600 + 722,93 + 2828,58 = 10078,16 \text{ (грн)} \quad (5.9)$$

Другий етап передбачає розрахунок загальних витрат виконання даної роботи всіма виконавцями.

Загальна вартість даної роботи визначається за Взаг формулою 5.10.

$$Взаг = В / \alpha \quad (5.10)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях.

Для нашого випадку α = 0,95, тому загальна вартість роботи буде:

$$Взаг = 10078,16 / 0,95 = 10608,59 \text{ (грн)}.$$

Третій етап передбачає прогнозування загальних витрат на розробку та впровадження результатів виконаної роботи.

Прогнозування загальних витрат здійснюється за формулою 5.11.

$$ЗВ = Взаг / \beta \quad (5.11)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Оскільки розробка знаходиться на стадії на розробки технологій, то β ≈ 0,4.

Тоді:

$$ЗВ = 10078,16 / 0,4 = 25195,40 \text{ (грн)}.$$

Отже, прогнозовані витрати на розробку онлайн система для вироблення спільних рішень групою людей становлять приблизно 25195,40 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Розробка та впровадження «Онлайн системи для вироблення спільних рішень групою людей» безпосередньо розробником на власному підприємстві дозволить отримати економічний ефект за рахунок оптимізації процесів групового прийняття рішень.

Вихідні умови для розрахунку:

а) Результати науково-технічної розробки будуть впроваджені з 2026 року;

б) Основні позитивні результати очікуються протягом 3 років після впровадження (2026-2028 рр.);

в) Покращення економічних показників діяльності підприємства:

- Скорочення часу на підготовку та проведення групових голосувань на 40%;
- Зменшення чисельності адміністративного персоналу на 0,5 штатної одиниці;

г) Обсяг діяльності: щомісячно проводиться 15 групових обговорень.

Розрахунок економічної ефективності:

Формула (5.12) - Річне збільшення чистого прибутку:

$$\Delta П = (\Delta Пя \times N + Пя \times \Delta N) = (360 \times 180 + 900 \times 36) = 97\ 200 \text{ грн} \quad (5.12)$$

де: ΔПя - економія на одній нараді (360 грн);

N - річна кількість нарад (180 од.);

Пя - собівартість наради (900 грн);

ΔN - додаткові наради (36 од.).

Формула (5.13) - Приведена вартість майбутніх доходів:

$$ПП = \Sigma[\Delta Пt/(1+\tau)\text{^}t] = 97\ 200/1{,}155 + 97\ 200/1{,}155^2 + 97\ 200/1{,}155^3 = 223\ 766{,}41 \text{ грн} \quad (5.13)$$

де: ΔПt - річний прибуток (97 200 грн);

τ - ставка дисконтування (15,5%), t - рік (1, 2, 3).

Формула (5.14) - Початкові інвестиції:

$$PV = k_{розр} \times ЗВ = 2,5 \times 25\,195,40 = 62\,988,50 \text{ грн} \quad (5.14)$$

де: kрозр - коефіцієнт витрат на впровадження (2,5);

ЗВ - витрати на розробку (25 195,40 грн);

Формула (5.15) - Чистий приведений дохід:

$$NPV = ПП - PV = 223\,766,41 - 62\,988,50 = 160\,777,91 \text{ грн} \quad (5.15)$$

де: ПП - приведена вартість доходів (223 766,41 грн);

PV - інвестиції (62 988,50 грн).

Формула (5.16) - Внутрішня норма рентабельності:

$$ERR = [\sqrt[3]{(1 + NPV/PV)}] - 1 = [\sqrt[3]{(1 + 160777,91/62988,50)}] - 1 = 51\% \quad (5.16)$$

де: NPV - чистий приведений дохід (160 777,91 грн);

PV - інвестиції (62 988,50 грн).

Формула (5.17) - Термін окупності інвестицій:

$$Ток = 1/E_e = 1/0,51 = 1,96 \text{ року} \quad (5.17)$$

де: E_e - внутрішня норма рентабельності (51%).

Висновки:

1. Чистий приведений дохід від впровадження системи становить 160 777,91 грн
2. Внутрішня норма рентабельності інвестицій складає 51%
3. Період окупності інвестицій дорівнює 1,96 року, що є меншим за 3 роки
4. Основним джерелом економічного ефекту є скорочення витрат на оплату праці
5. Впровадження системи є економічно доцільним та ефективним для підприємства

Отримані результати свідчать про високу економічну ефективність проекту, що робить його привабливим для впровадження.

Висновки

Проведені в розділі економічні розрахунки підтверджують високу ефективність та практичну цінність розробленої «Онлайн системи для вироблення спільних рішень групою людей».

Результати оцінки комерційного потенціалу розробки (47,3 бала) свідчать про її значні конкурентні переваги та перспективність для впровадження на ринку. Проведений аналіз за дванадцятьма критеріями демонструє переваги розробки порівняно з існуючими аналогами, зокрема у

сфері технічних характеристик, експлуатаційних витрат та ринкових перспектив.

Прогнозування витрат на виконання науково-дослідної роботи показало, що загальні витрати на розробку та впровадження системи становитимуть 25 195,40 грн, що є обґрунтованою сумою для проекту такого рівня складності.

Найбільш значущими є результати прогнозування комерційних ефектів, які підтверджують високу економічну ефективність впровадження системи:

- Чистий приведений дохід (NPV) становить 160 777,91 грн.
- Внутрішня норма рентабельності (ERR) досягає 51%.
- Термін окупності інвестицій складає лише 1, 96 року.

Отримані показники свідчать про те, що проект є високо ефективним і може бути успішно реалізований на підприємстві. Період окупності менше 2 років робить проект особливо привабливим для потенційних інвесторів.

Система забезпечить значну економію коштів за рахунок скорочення витрат на оплату праці, оптимізації часу проведення групових обговорень та підвищення ефективності управлінських процесів. Запропоноване рішення має значний комерційний потенціал та може бути успішно впроваджене як на підприємстві-розробнику, так і запропоноване іншим організаціям як готовий програмний продукт.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи було повністю розроблено та досліджено веб-орієнтовану онлайн-систему для колективного прийняття рішень. Основна мета роботи – створення програмного комплексу, що забезпечує структурований підхід до групового прийняття рішень – була успішно досягнута шляхом послідовного виконання всіх поставлених завдань.

Теоретична частина дослідження включала ґрунтовний аналіз філософських, психологічних та соціальних основ колективного прийняття рішень, що дозволило сформулювати науковий фундамент для розробки системи. Порівняльний огляд існуючих сервісів та інструментів, таких як Google Forms, Doodle, Tricider та Trello, виявив відсутність комплексного рішення на ринку. Більшість доступних засобів орієнтовані на вирішення вузькоспеціалізованих завдань і не забезпечують цілісного підходу, що охоплює всі етапи – від ініціації та обговорення ідей через процедуру голосування до контролю виконання та аналізу ефективності прийнятих рішень.

Практична реалізація системи була виконана з використанням сучасного технологічного стеку: серверна частина реалізована на платформі ASP.NET Core 8, що забезпечило високу продуктивність, безпеку та крос-платформеність; клієнтська частина побудована на фреймворці Bootstrap 5, що дозволило створити повністю адаптивний та інтуїтивно зрозумілий інтерфейс; для зберігання даних обрана система керування базами даних SQL Server, що гарантує надійність, транзакційну цілісність та безпеку даних. Система підтримує різноманітні методи голосування, включаючи багатокритеріальну оцінку з ваговими

коефіцієнтами, ранжування альтернатив, метод «За і Проти» та бальну систему, що забезпечує її універсальність для різних сценаріїв використання. Додатково реалізовано модулі управління завданнями, соціальної взаємодії через мережу контактів та потужної аналітики результатів з візуалізацією.

Функціональність та працездатність системи були перевірені методом чорної скриньки, що підтвердило повну відповідність системи сформованим функціональним вимогам, коректність її роботи та зручність інтерфейсу для користувачів з різним рівнем технічної підготовки. Економічне обґрунтування розробки засвідчило її значний комерційний потенціал, конкурентоспроможність на ринку та економічну доцільність впровадження розробки.

Шляхом подальшого вдосконалення системи є розширення переліку методів голосування, зокрема впровадження багатоетапного голосування, яке дозволить проводити відбір альтернатив у кілька турів, поступово відсіюючи менш популярні варіанти та концентруючись на обранні найбільш прийнятного рішення з короткого списку. Це підвищить точність та легітимність результатів, особливо в ситуаціях з великою кількістю варіантів або за відсутності явної більшості на першому етапі, зробивши процес прийняття колективних рішень більш гнучким та ефективним.

Таким чином, в рамках даної магістерської роботи було створено готовий до використання програмний інструмент, призначений для підвищення якості, швидкості та об'єктивності колективних рішень. Система може бути успішно впроваджена в різних сферах діяльності, включаючи корпоративний сектор, наукові установи, громадські організації, освітні заклади та державні установи, надаючи їм сучасний і ефективний механізм організації групової взаємодії та прийняття спільних рішень в умовах цифрового середовища.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Любунь О. І. Онлайн система для вироблення спільних рішень групою людей // Молодь в науці: дослідження, проблеми, перспективи: матеріали Міжнар. наук.-практ. інтернет-конф. (Вінниця, 16–17 квіт. 2026 р.). Вінниця : ВНТУ, 2026. Секція «Інтелектуальні інформаційні технології та автоматизація».
URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/schedConf/presentations (дата звернення: 05.12.2025).

2. Арістотель. Політика. К. : Видавництво Соломії Павличко, 2002. 389 с.

3. Bentham J. An Introduction to the Principles of Morals and Legislation. London : Oxford University Press, 1996. 312 p.

4. Habermas J. The Theory of Communicative Action. Boston : Beacon Press, 1985. 465 p.

5. Sunstein C. Infotopia: How Many Minds Produce Knowledge. Oxford : Oxford University Press, 2006. 273 p.

6. Arrow K. Social Choice and Individual Values. New Haven : Yale University Press, 1963. 124 p.

7. Apicroren. Політика. К. : Видавництво Соломії Павличко, 2002. 389 с.

8. Arrow K. Social Choice and Individual Values. New Haven : Yale University Press, 1963. 124 p.

9. Sunstein C. Infotopia: How Many Minds Produce Knowledge. Oxford : Oxford University Press, 2006. 273 p.

10. Simon H. A. Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization. New York : Free Press, 1997. 368 p.

11. Janis I. L. Groupthink: Psychological Studies of Policy Decisions and Fiascoes. Boston : Houghton Mifflin, 1982. 276 p.

12. Brown R. Group Processes: Dynamics within and between Groups. Oxford : Blackwell Publishing, 2000. 344 p.

13. Arrow K. Social Choice and Individual Values. New Haven : Yale University Press, 1963. 124 p.

14. Saaty T. L. The Analytic Hierarchy Process. New York : McGraw-Hill, 1980. 287 p.

15. Keeney R. L., Raiffa H. Decisions with Multiple Objectives: Preferences and Value Trade-offs. Cambridge : Cambridge University Press, 1993. 569 p.

16. Janis I. L. Groupthink: Psychological Studies of Policy Decisions and Fiascoes. Boston : Houghton Mifflin, 1982. 276 p.

17. Brown R. Group Processes: Dynamics within and between Groups. Oxford : Blackwell Publishing, 2000. 344 p.

18. Sunstein C. Infotopia: How Many Minds Produce Knowledge. Oxford : Oxford University Press, 2006. 273 p.

19. Tajfel H., Turner J. C. The Social Identity Theory of Intergroup Behavior. Psychology of Intergroup Relations. Chicago : Nelson-Hall, 1986. P. 7–24.

20. Magee J. C., Galinsky A. D. Social Hierarchy: The Self-Reinforcing Nature of Power and Status. The Academy of Management Annals. 2008. Vol. 2, No. 1. P. 351–398.

21. Hofstede G. Culture's Consequences: Comparing Values, Behaviors, Institutions and Organizations Across Nations. Thousand Oaks : Sage Publications, 2001. 596 p.

22. DeSanctis G., Gallupe R. B. A Foundation for the Study of Group Decision Support Systems. Management Science. 1987. Vol. 33, No. 5. P. 589–609.

23. Nunamaker J. F. et al. Electronic Meeting Systems to Support Group Work. Communications of the ACM. 1991. Vol. 34, No. 7. P. 40–61.

24. Power D. J. Decision Support Systems: Concepts and Resources for Managers. Westport : Greenwood Publishing Group, 2002. 258 p.

25. Swan M. Blockchain: Blueprint for a New Economy. Sebastopol : O'Reilly Media, 2015. 152 p.

26. Chen H., Chiang R. H., Storey V. C. Business Intelligence and Analytics: From Big Data to Big Impact. MIS Quarterly. 2012. Vol. 36, No. 4. P. 1165–1188.

27. Бойко М. В., Савчук Т. П. Моделі та методи групового прийняття рішень в умовах невизначеності. Науковий вісник НТУУ "КПІ". 2021.

28. Гриценко О. В. Методологічні засади аналізу колективного інтелекту в цифровому суспільстві. Філософські дослідження. 2020.

29. Ковальчук В. М. Динаміка групових рішень в онлайн-середовищі: соціально-психологічний аналіз. Психологія і суспільство. 2021.

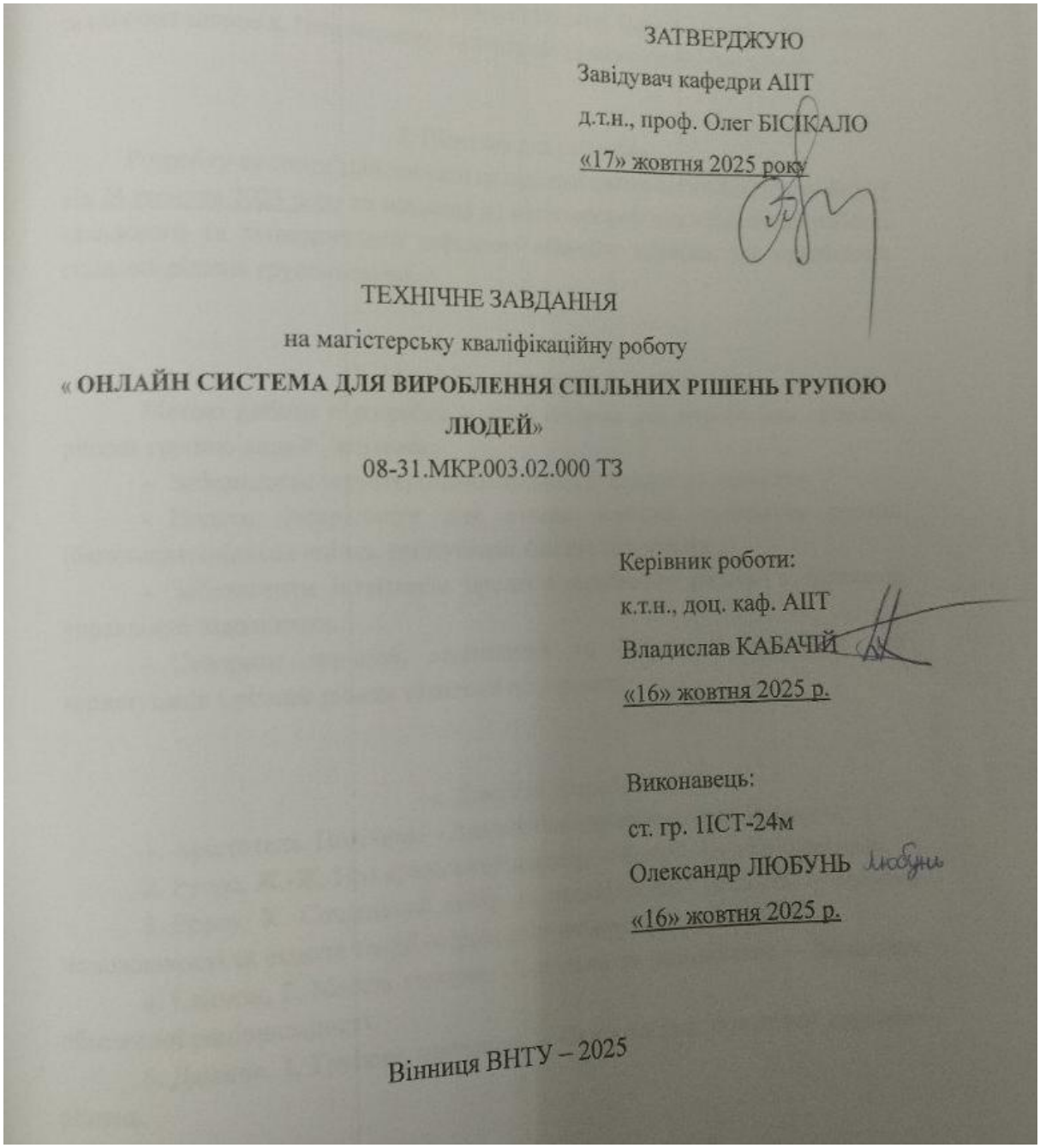
30. Петренко С. О. Архітектура інтегрованих систем підтримки прийняття колективних рішень. Інформаційні технології та комп'ютерна інженерія. 2019.

31. Мельник Л. І., Шевченко О. В. Емпіричне оцінювання ефективності методів голосування в корпоративному середовищі. Вісник Київського національного університету імені Тараса Шевченка. Серія: Економіка. 2022.

ДОДАТКИ

Додаток А (обов’язковий)

Технічне завдання



ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
« ОНЛАЙН СИСТЕМА ДЛЯ ВИРОБЛЕННЯ СПІЛЬНИХ РІШЕНЬ ГРУПОЮ ЛЮДЕЙ»
08-31.МКР.003.02.000 ТЗ

Керівник роботи:
к.т.н., доц. каф. АІІТ
Владислав КАБАЧІЙ
«16» жовтня 2025 р.

Виконавець:
ст. гр. 1ІСТ-24м
Олександр ЛЮБУНЬ
«16» жовтня 2025 р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

Онлайн система для вироблення спільних рішень групою людей.

Інформаційні системи та технології. Бізнес та корпоративне управління. Державне управління та громадянське суспільство. Освіта та наука. Медицина та охорона здоров'я. Некомерційні організації та міжнародні інституції.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № 313 від 24 вересня 2025 року та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Онлайн система для вироблення спільних рішень групою людей »

3. Мета та призначення розробки

Метою роботи є розробка онлайн система для вироблення спільних рішень групою людей , яка може:

- Забезпечити структурований процес групового голосування.
- Надати інструменти для різних методів прийняття рішень (багатокритеріальна оцінка, ранжування, бальне голосування).
- Забезпечити інтеграцію процесів прийняття рішень з системою управління завданнями.
- Створити зручний, адаптивний та безпечний інтерфейс для користувачів з різним рівнем технічної підготовки.

4. Джерела розробки

1. Арістотель. Політика. – Античні засади колективної мудрості.

2. Руссо, Ж.-Ж. Про суспільний договір. – Концепція загальної волі.

3. Ерроу, К. Соціальний вибір та індивідуальні цінності. – Теорема неможливості та основи теорії соціального вибору.

4. Саймон, Г. Моделі людини. Соціальне та раціональне. – Концепція обмеженої раціональності.

5. Джанис, І. Групове мислення. – Психологічні механізми групових рішень.

5. Показники призначення

5.1. Основні технічні характеристики системи

Функціональні можливості:

5.1.1. Модуль управління користувачами та автентифікації:

- Реєстрація нового користувача з валідацією унікальності логіна, перевіркою складності пароля.
- Автентифікація (вхід) за логіном та паролем з опцією "Запам'ятати мене".
- Безпечний вихід із системи (logout).
- Управління профілем: перегляд та редагування основних даних, зміна пароля.

5.1.2. Модуль соціальної мережі (контакти):

- Пошук користувачів системи за іменем, прізвищем, логіном.
- Надсилання, перегляд, прийняття та відхилення запитів на додавання до контактів.
- Управління списком підтверджених контактів (перегляд, видалення).

5.1.3. Модуль створення та управління сесіями голосування:

- Створення нової сесії з параметрами: назва, опис, тип (приватна/публічна), дата початку та завершення.
- Вибір учасників приватної сесії зі списку контактів.
- Визначення альтернатив рішення (варіантів вибору).
- Налаштування критеріїв оцінювання з можливістю призначення ваги (важливості) кожному критерію.
- Перегляд голосувань.

5.1.4. Модуль участі в голосуванні та оцінювання:

- Перегляд списку голосувань.
- Інтерфейс для голосування за обраним методом: для "Багатокритеріальної оцінки" заповнення матриці оцінок (альтернатива × критерій); для "Ранжування" встановлення порядку пріоритету для альтернатив.
- Підтвердження та фіксація відправленого голосу.

5.1.5. Модуль аналізу та візуалізації результатів:

- Автоматичний розрахунок підсумків голосування відповідно до обраного методу.

- Формування детальних статистичних звітів: табличні дані, середні оцінки, розподіл голосів.

- Побудова діаграм для візуалізації результатів: стовпчасті діаграми рейтингу альтернатив.

5.1.6. Модуль управління завданнями:

- Створення, редагування, видалення завдань з атрибутами: назва, опис, дедлайн, пріоритет, статус.

- Призначення та керування спільним доступом до завдань для інших користувачів.

- Фільтрація та пошук завдань за різними критеріями (активні, завершені, важливі).

5.2. Мінімальні системні вимоги

5.2.1. Вимори до клієнтського середовища (користувач):

- Браузер: сучасна версія Google Chrome, Mozilla Firefox, Microsoft Edge, Safari з підтримкою HTML5, CSS3, ES6+.
- Роздільна здатність екрана: підтримка адаптивного інтерфейсу для мобільних пристроїв (від 320px) та десктопів (від 1024px).
- Інтернет-з'єднання: стабільне з'єднання для роботи з інтерактивними елементами інтерфейсу.

5.3. Вхідні дані

- Дані користувача при реєстрації: логін, пароль, ім'я, прізвище.

- Дані для створення голосування: метадані: назва, опис, тип, діапазон дат; список учасників; список альтернатив (варіантів вибору); список критеріїв оцінювання з вагами.

- Дані голосування (від кожного учасника): для методу "Багатокритеріальна оцінка": числові оцінки для кожної пари "альтернатива-критерій"; для методу "Ранжування": впорядкований список альтернатив.

5.4. Результати роботи програми

5.4.1. Налаштована та готова до роботи система:

- Налаштований веб-застосунок з інтерфейсом користувача.

- Ініціалізована база даних із структурою сутностей.

- Функціонуюча система безпеки та автентифікації.

5.4.2. Результати голосування (для кожної сесії):

- Детальний аналітичний звіт: у веб-інтерфейсі системи з таблицями та графіками.

- Зведена інформація: переможна альтернатива(и) згідно з обраним методом; розподіл голосів між усіма альтернативами; статистика за критеріями (середня оцінка, важливість); рівень участі (кількість учасників, які проголосували).

6. Економічні показники

До економічних показників входять:

- витрати на розробку – до 25 195,40 тис. грн.
- термін окупності – до 3х років

7. Стадії розробки:

1. Розділ 1 «Теоретичні основи та сучасний стан дослідження колективного прийняття рішень» має бути виконаний до 05.10.2025 р.
2. Розділ 2 «Аналіз доцільності розробки онлайн системи для вироблення спільних рішень групою людей» має бути виконаний до 25.10.2025 р.
3. Розділ 3 «Розробка онлайн системи для вироблення спільних рішень групою людей» має бути виконаний до 20.11.2025 р.
4. Розділ 4 «Тестування та відлагодження онлайн системи для вироблення спільних рішень групою людей» та 5 «Економічний розділ» має бути виконаний до 01.12.2025 р.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 14.11.2025.
2. Попередній захист магістерської кваліфікаційної роботи провести до 02.12.2025.
3. Захист магістерської кваліфікаційної роботи провести в період з 15.12.2025 р. до 19.12.2025 р.

Додаток Б (обов’язковий)

Ілюстративна частина

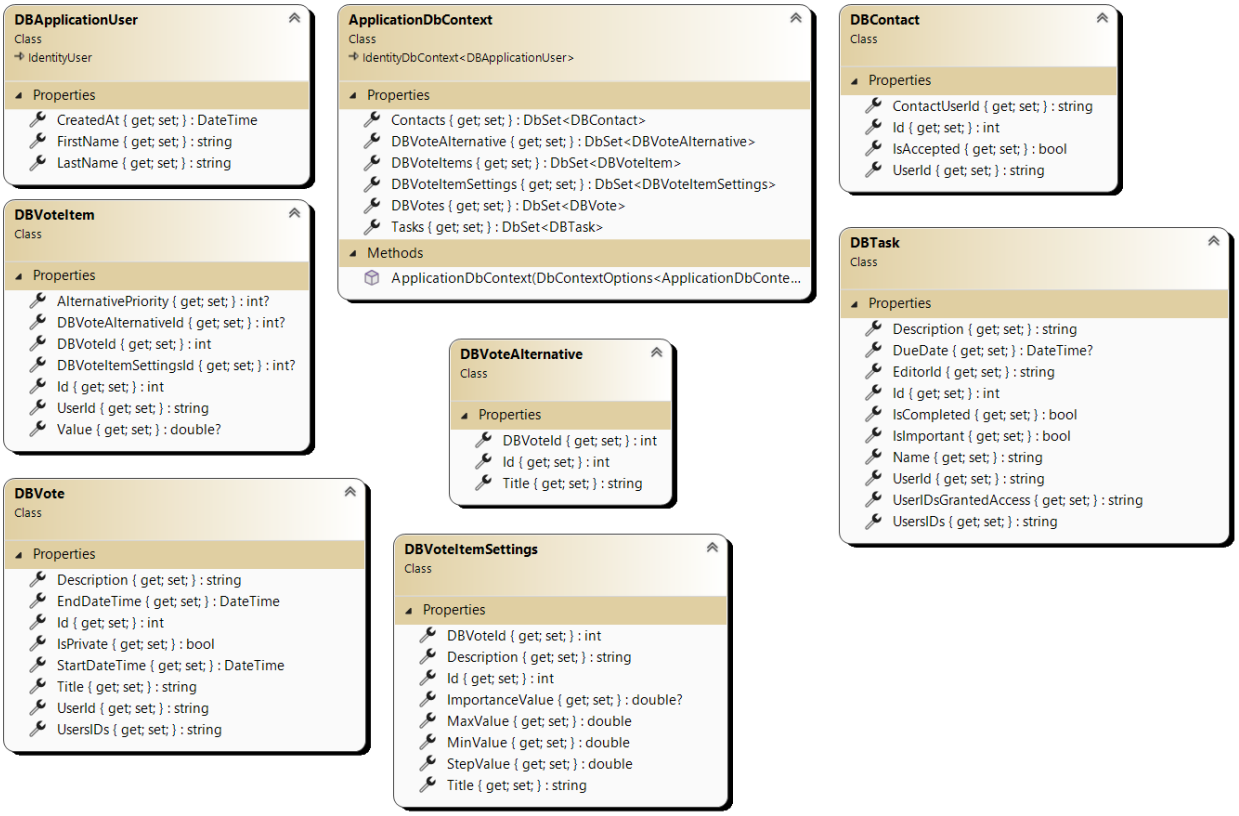


Рисунок Б.1 – UML-діагрма «Сутнісно-орієнтованна модель бази даних»

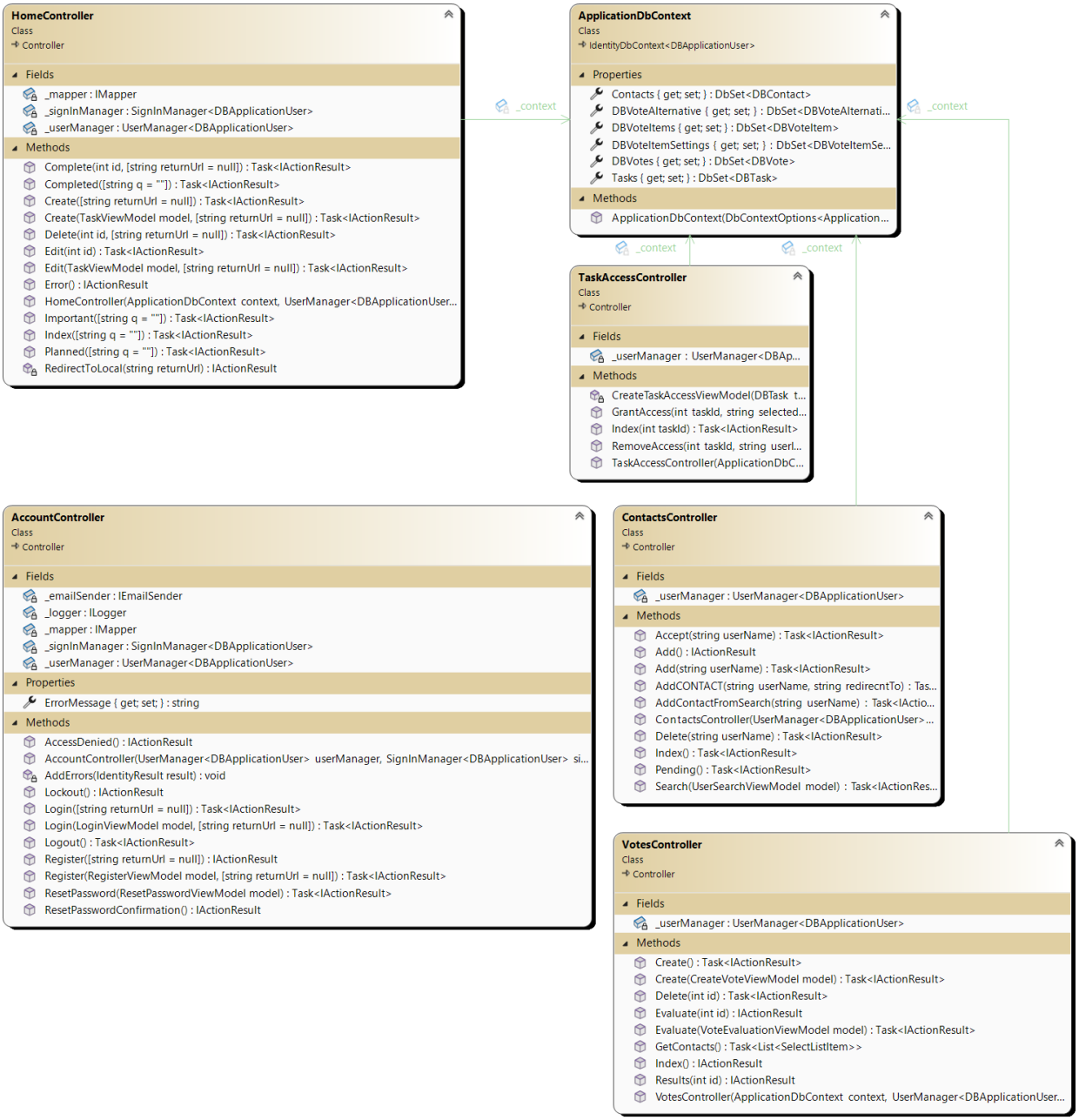


Рисунок Б.2 – UML-діаграма «Класи контролери»

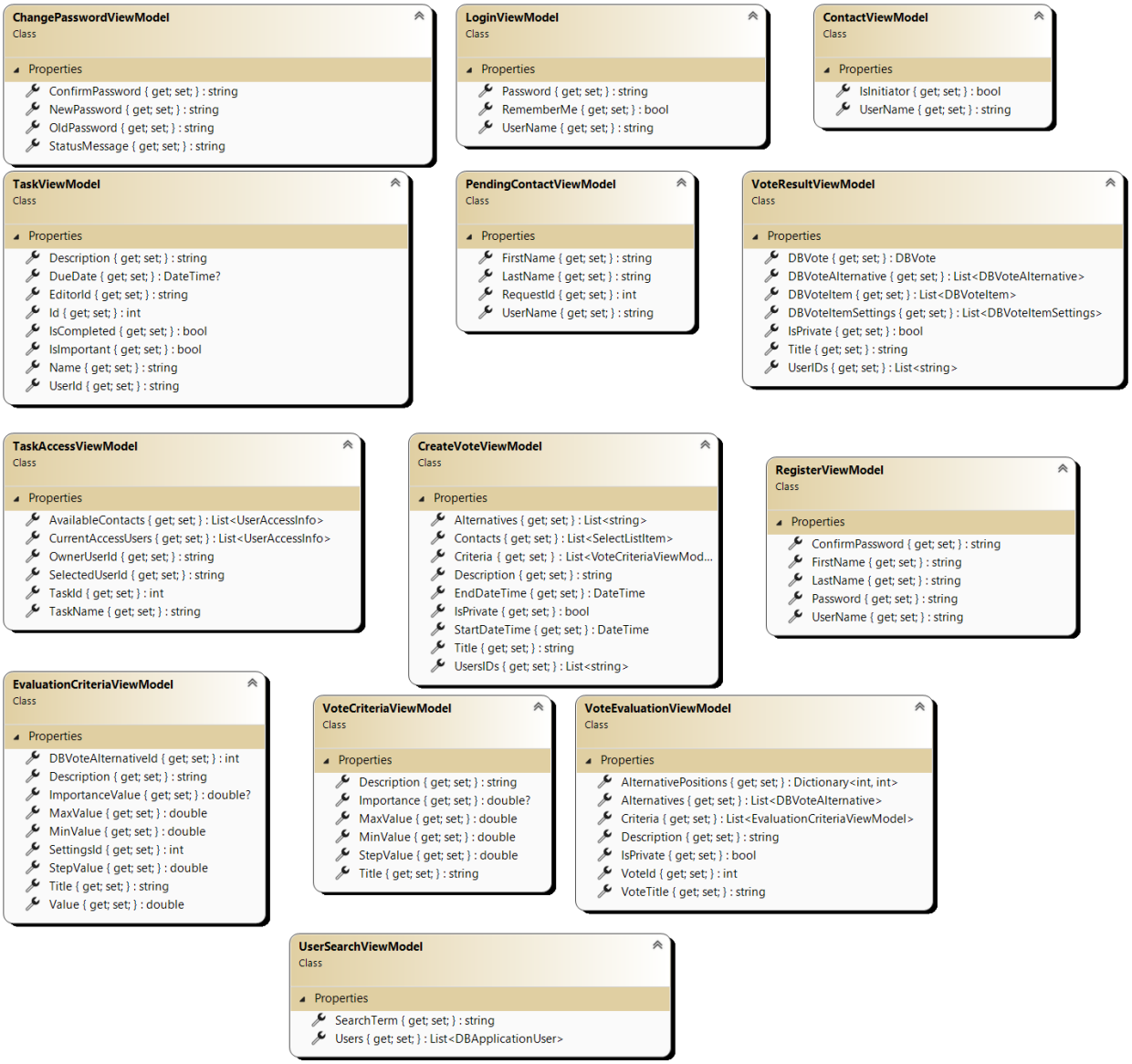


Рисунок Б.3 – UML-діаграма «Класи представлень»

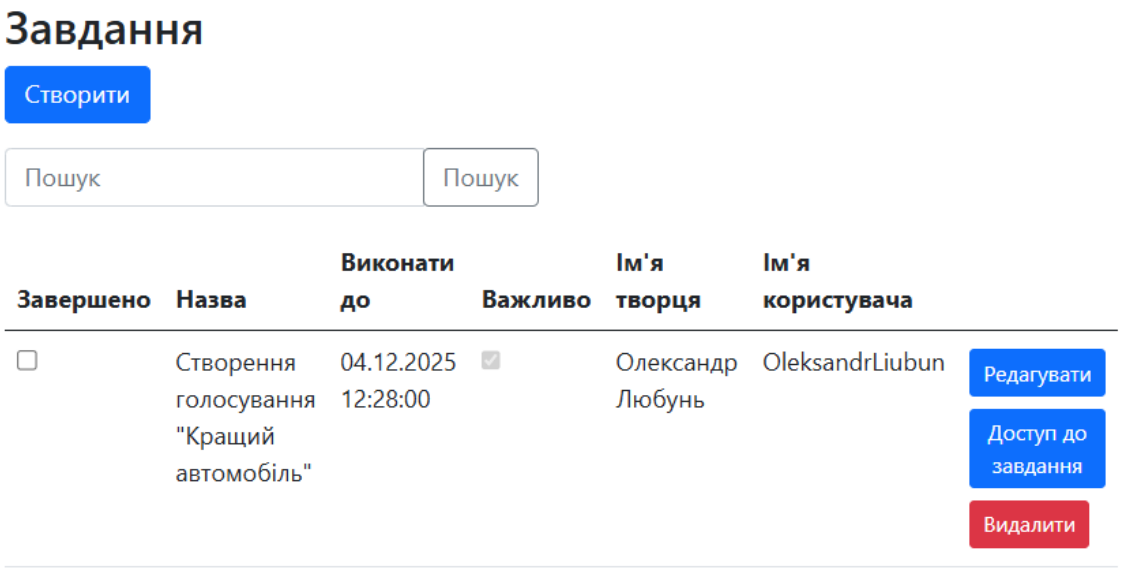


Рисунок Б.4 – Розділ сайту для перегляду завдань

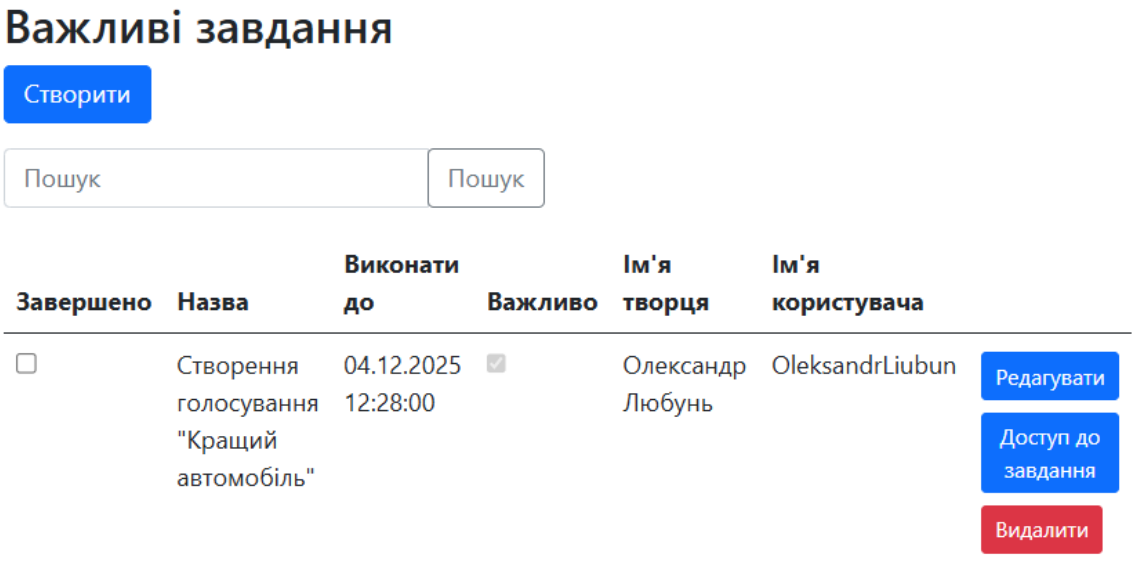


Рисунок Б.5 – Розділ сайту для важливих завдань

Продовження додатку Е

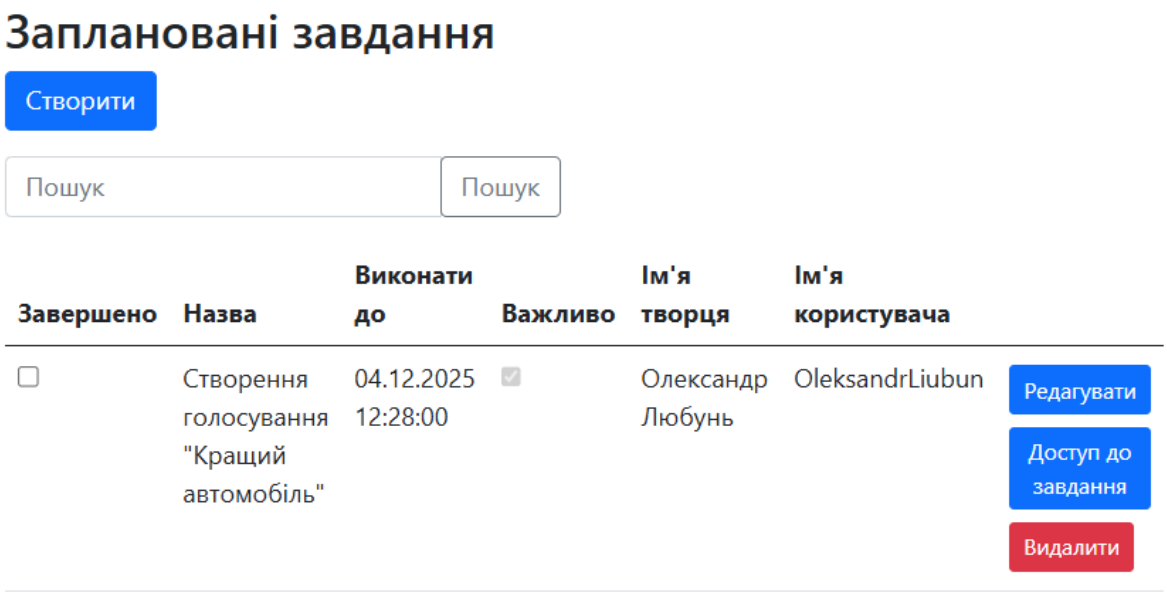


Рисунок Б.6 – Розділ сайту для запланованих завдань

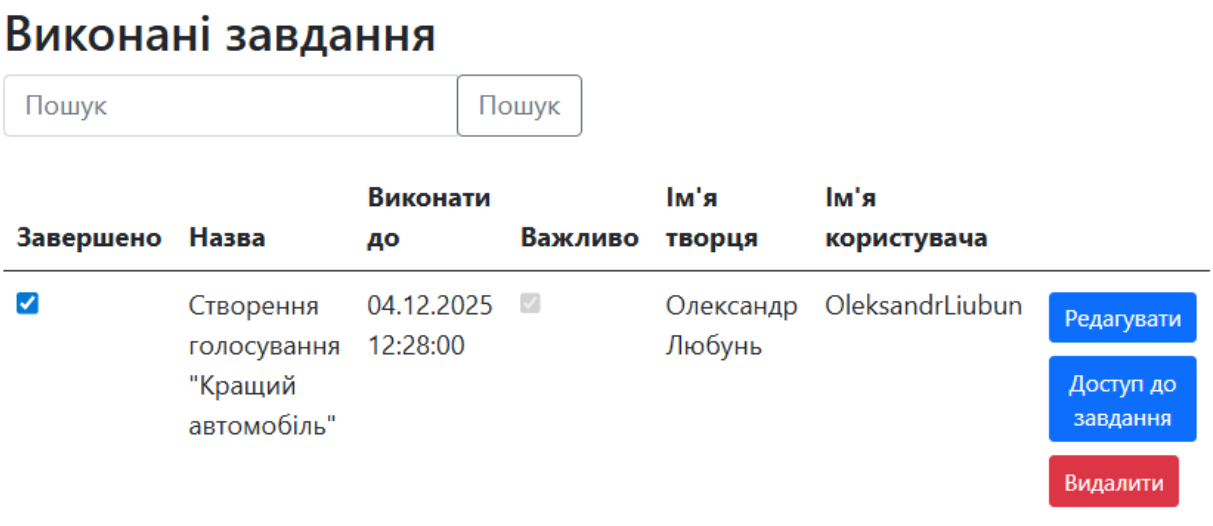


Рисунок Б.7 – Розділ сайту для виконаних завдань

Продовження додатку Е

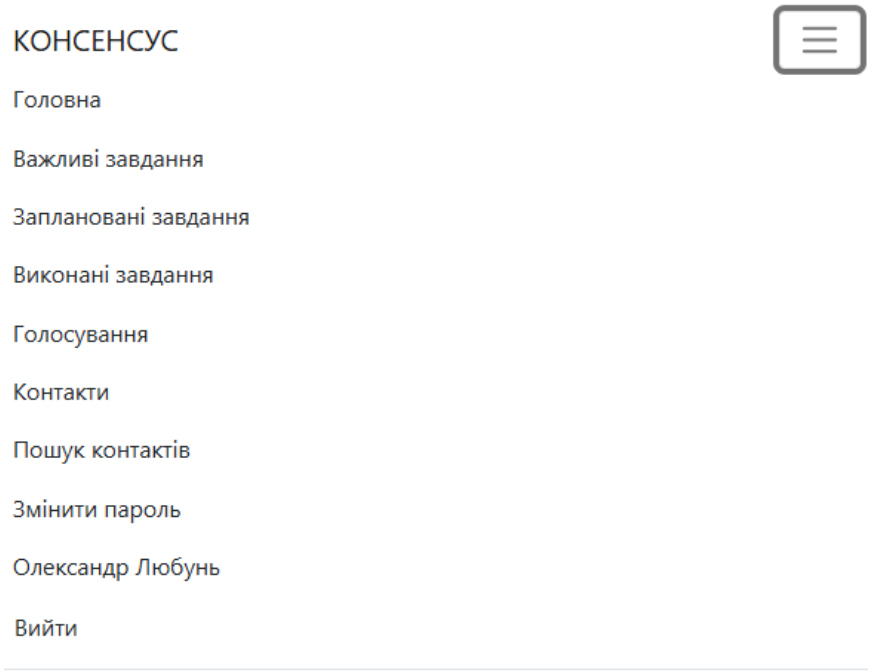


Рисунок Б.8 – Головне меню в мобільній версії додатку

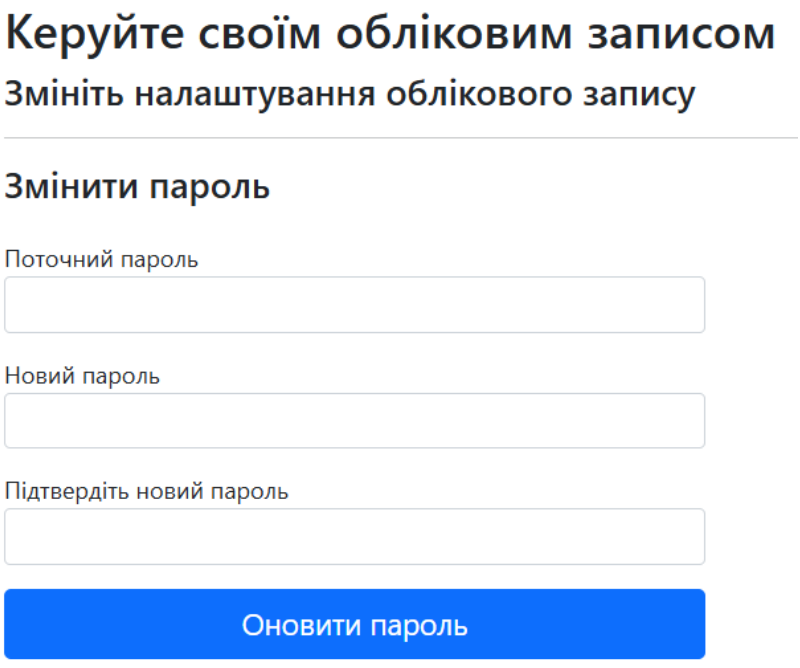


Рисунок Б.9 – Форма зміни паролю

Додаток В (обов'язковий)
Лістинг онлайн додатку

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
public class UserSearchViewModel
{
    public string SearchTerm { get; set; }
    public List<DBApplicationUser> Users { get; set; } = new List<DBApplicationUser>();
}

[Authorize]
public class ContactsController : Controller
{
    private readonly UserManager<DBApplicationUser> _userManager;
    private readonly ApplicationDbContext _context;

    public ContactsController(UserManager<DBApplicationUser> userManager, ApplicationDbContext context)
    {
        _userManager = userManager;
        _context = context;
    }
    public async Task<IActionResult> Search(UserSearchViewModel model)
    {
        var users = _userManager.Users.AsQueryable();

        if (!string.IsNullOrEmpty(model.SearchTerm))
        {
            var searchTerm = model.SearchTerm.Trim().ToLower();

            users = users.Where(u =>
                u.FirstName.ToLower().Contains(searchTerm) ||
                u.LastName.ToLower().Contains(searchTerm) ||
                u.UserName.ToLower().Contains(searchTerm) ||
                u.Email.ToLower().Contains(searchTerm)
            );
        }
        model.Users = await users.OrderBy(u => u.LastName)
                            .ThenBy(u => u.FirstName)
                            .ToListAsync();

        return View(model);
    }

    // GET: Contacts/Add
    public IActionResult Add()
    {
        return View();
    }
    // POST: Contacts/Add
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> AddContactFromSearch(string userName)
    {
        return await AddCONTACT(userName, "Search");
    }
    // POST: Contacts/Add
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Add(string userName) {
        return await AddCONTACT(userName, "Pending");
    }
  public async Task<IActionResult> AddCONTACT(string userName, string redirecntTo)
{
```

```
    var currentUser = await _userManager.GetUserAsync(User);
    if (currentUser == null) return Challenge();

    var contactUser = await _userManager.FindByNameAsync(userName);
    if (contactUser == null)
    {
        ModelState.AddModelError("", "User not found.");
        return View();
    }

    if (currentUser.Id == contactUser.Id)
    {
        ModelState.AddModelError("", "You cannot add yourself as a contact.");
        return View();
    }

    // Check if contact already exists in either direction
    bool contactExists = await _context.Contacts
        .AnyAsync(c => (c.UserId == currentUser.Id && c.ContactUserId == contactUser.Id) ||
                       (c.UserId == contactUser.Id && c.ContactUserId == currentUser.Id));

    if (contactExists)
    {
        ModelState.AddModelError("", "Contact relationship already exists.");
        return View();
    }

    var contact = new DBContact
    {
        UserId = currentUser.Id,
        ContactUserId = contactUser.Id,
        IsAccepted = false
    };

    _context.Contacts.Add(contact);
    await _context.SaveChangesAsync();

    TempData["Success"] = "Запит на контакт успішно надіслано!";
    return RedirectToAction(redirecntTo);
}

// POST: Contacts/Delete
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Delete(string userName)
{
    var currentUser = await _userManager.GetUserAsync(User);
    if (currentUser == null) return Challenge();

    var contactUser = await _userManager.FindByNameAsync(userName);
    if (contactUser == null)
    {
        TempData["Error"] = "User not found.";
        return RedirectToAction("Index");
    }

    var contacts = await _context.Contacts
        .Where(c => (c.UserId == currentUser.Id && c.ContactUserId == contactUser.Id) ||
                    (c.UserId == contactUser.Id && c.ContactUserId == currentUser.Id))
        .ToListAsync();

    if (!contacts.Any())
    {


        TempData["Error"] = "Contact not found.";
```

```
return RedirectToAction("Index");
        }

        _context.Contacts.RemoveRange(contacts);
        await _context.SaveChangesAsync();

        TempData["Success"] = "Contact removed successfully!";
        return RedirectToAction("Index");
    }

    // POST: Contacts/Accept
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Accept(string userName)
    {
        var currentUser = await _userManager.GetUserAsync(User);
        if (currentUser == null) return Challenge();

        var contactUser = await _userManager.FindByNameAsync(userName);
        if (contactUser == null)
        {
            TempData["Error"] = "User not found.";
            return RedirectToAction("Pending");
        }

        var contact = await _context.Contacts
            .FirstOrDefaultAsync(c => c.UserId == contactUser.Id &&
                                      c.ContactUserId == currentUser.Id &&
                                      !c.IsAccepted);

        if (contact == null)
        {
            TempData["Error"] = "Contact request not found.";
            return RedirectToAction("Pending");
        }

        contact.IsAccepted = true;
        await _context.SaveChangesAsync();

        TempData["Success"] = "Contact request accepted!";
        return RedirectToAction("Index");
    }

    public async Task<IActionResult> Pending()
    {
        var currentUser = await _userManager.GetUserAsync(User);
        if (currentUser == null) return Challenge();

        // Get pending requests where current user is the recipient
        var pendingRequests = await _context.Contacts
            .Where(c => c.ContactUserId == currentUser.Id && !c.IsAccepted)
            .ToListAsync();

        // Get usernames for each pending request
        var pendingViewModels = new List<PendingContactViewModel>();
        foreach (var request in pendingRequests)
        {
            var user = await _userManager.FindByIdAsync(request.UserId);
            if (user != null)
            {
                pendingViewModels.Add(new PendingContactViewModel
                {
                    FirstName = user.FirstName,
                    LastName = user.LastName,
                    UserName = user.UserName,
                    RequestId = request.Id
```

```
});
            }
        }
        return View(pendingViewModels);
    }

    // GET: Contacts/Index
    public async Task<IActionResult> Index()
    {
        var currentUser = await _userManager.GetUserAsync(User);
        if (currentUser == null) return Challenge();

        // Get all accepted contacts where current user is either UserId or ContactUserId
        var contacts = await _context.Contacts
            .Where(c => (c.UserId == currentUser.Id || c.ContactUserId == currentUser.Id) && c.IsAccepted)
            .ToListAsync();

        var contactViewModels = new List<ContactViewModel>();
        foreach (var contact in contacts)
        {
            string otherUserId;
            bool isInitiator;

            if (contact.UserId == currentUser.Id)
            {
                otherUserId = contact.ContactUserId;
                isInitiator = true;
            }
            else
            {
                otherUserId = contact.UserId;
                isInitiator = false;
            }

            var otherUser = await _userManager.FindByIdAsync(otherUserId);
            if (otherUser != null)
            {
                contactViewModels.Add(new ContactViewModel
                {
                    UserName = otherUser.UserName,
                    IsInitiator = isInitiator
                });
            }
        }

        return View(contactViewModels);
    }
}

// ViewModels for display
public class ContactViewModel
{
    public string UserName { get; set; }
    public bool IsInitiator { get; set; }
}

public class PendingContactViewModel
{
    public string UserName { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
public int RequestId { get; set; }
}
using AutoMapper;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
```

Продовження додатку В

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics;

[Authorize]
public class HomeController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly UserManager<DBApplicationUser> _userManager;
    private readonly SignInManager<DBApplicationUser> _signInManager;
    private readonly IMapper _mapper;
    public HomeController(
        ApplicationDbContext context,
        UserManager<DBApplicationUser> userManager,
        SignInManager<DBApplicationUser> signInManager,
        IMapper mapper)
    {
        _context = context;
        _userManager = userManager;
        _signInManager = signInManager;
        _mapper = mapper;
    }

    [AllowAnonymous]
    public async Task<IActionResult> Index(string q = "")
    {
        var user = await _userManager.GetUserAsync(User);
        if (!_signInManager.IsSignedIn(User))
        {
            return Redirect("/Account/Login");
        }
        return View(_mapper.Map<IEnumerable<DBTask>, IEnumerable<TaskViewModel>>(await _context.Tasks.Where(item => (item.UserId ==
user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && !item.IsCompleted && (string.IsNullOrEmpty(q) ? true :
item.Name.ToLower().Contains(q))).ToListAsync()));
    }

    public async Task<IActionResult> Important(string q = "")
    {
        var user = await _userManager.GetUserAsync(User);
        return View(_mapper.Map<IEnumerable<DBTask>, IEnumerable<TaskViewModel>>(await _context.Tasks.Where(item => (item.UserId ==
user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.IsImportant && !item.IsCompleted && (string.IsNullOrEmpty(q) ? true :
item.Name.ToLower().Contains(q))).ToListAsync()));
    }

    public async Task<IActionResult> Planned(string q = "")
    {
        var user = await _userManager.GetUserAsync(User);
        return View(_mapper.Map<IEnumerable<DBTask>, IEnumerable<TaskViewModel>>(await _context.Tasks.Where(item => (item.UserId ==
user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.DueDate != null && !item.IsCompleted && (string.IsNullOrEmpty(q) ? true :
item.Name.ToLower().Contains(q))).ToListAsync()));
    }

    public async Task<IActionResult> Completed(string q = "")
    {
   var user = await _userManager.GetUserAsync(User);
        return View(_mapper.Map<IEnumerable<DBTask>, IEnumerable<TaskViewModel>>(await _context.Tasks.Where(item => (item.UserId ==
user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.IsCompleted && (string.IsNullOrEmpty(q) ? true :
item.Name.ToLower().Contains(q))).ToListAsync()));

    }

    public async Task<IActionResult> Create(string returnUrl = null)
    {
        return View();
    }
```

Продовження додатку В

```
public async Task<IActionResult> Edit(int id)
    {
        var user = await _userManager.GetUserAsync(User);
        return View(_mapper.Map<TaskViewModel>(_context.Tasks.First(item => (item.UserId == user.Id ||
item.UserIDsGrantedAccess.Contains(user.Id)) && item.Id == id)));
    }

    [HttpPatch]
    public async Task<IActionResult> Complete(int id, string returnUrl = null)
    {
        var user = await _userManager.GetUserAsync(User);
        var task = _context.Tasks.First(item => (item.UserId == user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.Id == id);
        task.IsCompleted = !task.IsCompleted;
        _context.Update(task);
        await _context.SaveChangesAsync();
        return RedirectToLocal(returnUrl);
    }

    [HttpPost]
    public async Task<IActionResult> Edit(TaskViewModel model, string returnUrl = null)
    {
        var user = await _userManager.GetUserAsync(User);
        var task = _context.Tasks.First(item => (item.UserId == user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.Id == model.Id);
        model.UserId = task.UserId;
        _mapper.Map(model, task);
        task.EditorId = user.Id;
        _context.Update(task);
        _context.SaveChanges();
        return RedirectToLocal(returnUrl);
    }

    [HttpPost]
    public async Task<IActionResult> Create(TaskViewModel model, string returnUrl = null)
    {
        var user = await _userManager.GetUserAsync(User);
        if (!ModelState.IsValid)
        {
            return View(model);
        }
        model.UserId = user.Id;
        var dBTask = _mapper.Map<DBTask>(model);
        dBTask.UserIDsGrantedAccess = user.Id;
        dBTask.UsersIDs = user.Id;
        dBTask.EditorId = user.Id;
        _context.Tasks.Add(dBTask);
        await _context.SaveChangesAsync();
        return RedirectToLocal(returnUrl);
    }

    [HttpDelete]
    public async Task<IActionResult> Delete(int id, string returnUrl = null)
    {
        var user = await _userManager.GetUserAsync(User);
    _context.Remove(_context.Tasks.First(item => (item.UserId == user.Id || item.UserIDsGrantedAccess.Contains(user.Id)) && item.Id == id));
        _context.SaveChanges();
        return RedirectToLocal(returnUrl);
    }


    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }

    private IActionResult RedirectToLocal(string returnUrl)
    {
```

Продовження додатку В

```
 if (!string.IsNullOrEmpty(returnUrl))
        {
            return Redirect(returnUrl);
        }
        else
        {
            return RedirectToAction(nameof(HomeController.Index), "Home");
        }
    }
}
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
public class TaskAccessViewModel
{
    public int TaskId { get; set; }
    public string TaskName { get; set; }
    public string OwnerUserId { get; set; }
    public List<UserAccessInfo> CurrentAccessUsers { get; set; } = new List<UserAccessInfo>();
    public List<UserAccessInfo> AvailableContacts { get; set; } = new List<UserAccessInfo>();
    public string SelectedUserId { get; set; }
}

public class UserAccessInfo
{
    public string UserId { get; set; }
    public string DisplayName { get; set; }
}
public class TaskAccessController : Controller
{
    private readonly ApplicationDbContext _context;
    private readonly UserManager<DBApplicationUser> _userManager;

    public TaskAccessController(ApplicationDbContext context, UserManager<DBApplicationUser> userManager)
    {
        _context = context;
        _userManager = userManager;
    }

    public async Task<IActionResult> Index(int taskId)
    {
        var task = await _context.Tasks.FirstOrDefaultAsync(t => t.Id == taskId);
        if (task == null)
        {
            return NotFound();
        }

        var model = await CreateTaskAccessViewModel(task);
return View(model);
    }

[HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> GrantAccess(int taskId, string selectedUserId)
    {

        var task = await _context.Tasks.FirstOrDefaultAsync(t => t.Id == taskId);
        if (task == null)
        {
            return NotFound();
        }

        if (!string.IsNullOrEmpty(selectedUserId))
        {
            var currentAccessUsers = task.UserIDsGrantedAccess?.Split(',', StringSplitOptions.RemoveEmptyEntries).ToList() ?? new List<string>();
```

```
if (!currentAccessUsers.Contains(selectedUserId))
            {
                currentAccessUsers.Add(selectedUserId);
                task.UserIDsGrantedAccess = string.Join(",", currentAccessUsers);
                await _context.SaveChangesAsync();
            }
        }

        var model = await CreateTaskAccessViewModel(task);
        return View("Index", model);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> RemoveAccess(int taskId, string userId)
    {
        var task = await _context.Tasks.FirstOrDefaultAsync(t => t.Id == taskId);
        if (task == null || task.UserId == userId)
        {
            return NotFound();
        }

        if (!string.IsNullOrEmpty(userId))
        {
            var currentAccessUsers = task.UserIDsGrantedAccess?.Split(',', StringSplitOptions.RemoveEmptyEntries).ToList() ?? new List<string>();
            currentAccessUsers.Remove(userId);
            task.UserIDsGrantedAccess = currentAccessUsers.Any() ? string.Join(",", currentAccessUsers) : null;
            await _context.SaveChangesAsync();
        }

        var model = await CreateTaskAccessViewModel(task);
        return View("Index", model);
    }

    private async Task<TaskAccessViewModel> CreateTaskAccessViewModel(DBTask task)
    {
        var model = new TaskAccessViewModel
        {
            TaskId = task.Id,
            TaskName = task.Name,
            OwnerUserId = task.UserId
        };

        // Get current users with access
        if (!string.IsNullOrEmpty(task.UserIDsGrantedAccess))
        {
            var userIds = task.UserIDsGrantedAccess.Split(',', StringSplitOptions.RemoveEmptyEntries);
            foreach (var userId in userIds)        {
                var user = await _userManager.FindByIdAsync(userId);
                 model.CurrentAccessUsers.Add(new UserAccessInfo
                {
                    UserId = userId,
                    DisplayName = $"{user?.FirstName} {user?.LastName} / {user?.UserName}"
                });


            }
        }

        // Get available contacts for the task owner
        var contacts = await _context.Contacts
            .Where(c => (c.UserId == task.UserId || c.ContactUserId == task.UserId) && c.IsAccepted)
            .ToListAsync();

        var currentAccessUserIds = model.CurrentAccessUsers.Select(u => u.UserId).ToList();
```

Продовження додатку В

```
 foreach (var contact in contacts)
        {
            if (!currentAccessUserIds.Contains(contact.ContactUserId) && (contact.ContactUserId != task.UserId) ||
!currentAccessUserIds.Contains(contact.UserId) && (contact.UserId != task.UserId))
            {
                var user = await _userManager.FindByIdAsync(contact.ContactUserId == task.UserId ? contact.UserId : contact.ContactUserId);
                model.AvailableContacts.Add(new UserAccessInfo
                {
                    UserId = user.Id,
                    DisplayName = $"{user?.FirstName} {user?.LastName} / {user?.UserName}"
                });
            }
        }

        return model;
    }
}
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.Security.Claims;
public class CreateVoteViewModel
{
    [Required(ErrorMessage = SharedResource.RequireMessage)]
    [StringLength(200)]
    [Display(Name = "Назва")]
    public string Title { get; set; }
    [Display(Name = "Приватне")]
    public bool IsPrivate { get; set; }
    [Display(Name = "Опис")]
    public string Description { get; set; }

    [Display(Name = "Дата початку")]
    public DateTime StartDateTime { get; set; } = DateTime.Now;

    [Display(Name = "Дата завершення")]
    public DateTime EndDateTime { get; set; } = DateTime.Now.AddDays(7);

    [Display(Name = "Учасники")]
    public List<string> UsersIDs { get; set; }
    public List<string> Alternatives { get; set; } = new();
    public List<VoteCriteriaViewModel> Criteria { get; set; } = new();
    public List<SelectListItem> Contacts { get; set; } = new();
} public class VoteCriteriaViewModel
{
    [Required(ErrorMessage = SharedResource.RequireMessage)]
    [Display(Name = "Назва")]
    public string Title { get; set; }


    [Display(Name = "Опис")]
    public string Description { get; set; }

    [Range(1, 100)]
    [Display(Name = "Важливість")]
    public double? Importance { get; set; }
    [Display(Name = "Мінімальне значення")]
    public double MinValue { get; set; } = 0;
    [Display(Name = "Крок")]
    public double StepValue { get; set; } = 0;
    [Display(Name = "Максимальне значення")]
    public double MaxValue { get; set; } = 10;
}
```

Продовження додатку В

```
public class VoteEvaluationViewModel
{
    public int VoteId { get; set; }
    [Display(Name = "Назва")]
    public string VoteTitle { get; set; }
    [Display(Name = "Опис")]
    public string Description { get; set; }
    [Display(Name = "Приватне")]
    public bool IsPrivate { get; set; }
    public List<DBVoteAlternative> Alternatives { get; set; } = new();
    public List<EvaluationCriteriaViewModel> Criteria { get; set; } = new();
    [Display(Name = "Позиції альтернатив")]
    public Dictionary<int, int> AlternativePositions { get; set; } = new Dictionary<int, int>();
}

public class EvaluationCriteriaViewModel
{
    public int SettingsId { get; set; }
    public int DBVoteAlternativeId { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }

    [Range(0, 100)]
    public double? ImportanceValue { get; set; }

    public double MinValue { get; set; }
    public double MaxValue { get; set; }
    public double StepValue { get; set; }
    [Range(0, double.MaxValue)]
    [Display(Name = "Значення")]
    public double Value { get; set; }
}
public class VoteResultViewModel
{
    public string Title { get; set; }
    public bool IsPrivate { get; set; }
    public DBVote DBVote { get; set; }
    public List<string> UserIDs { get; set; }
    public List<DBVoteItemSettings> DBVoteItemSettings { get; set; }
    public List<DBVoteItem> DBVoteItem { get; set; }
    public List<DBVoteAlternative> DBVoteAlternative { get; set; }}

public static class LinqExtensions
{
    public static double SafeAverage<TSource>(this IEnumerable<TSource> source, Func<TSource, double> selector)

    {
        if (source == null || !source.Any())
            return 0.0;

        return source.Average(selector);
    }
}

@model IEnumerable<DBVote>
@{
    ViewData["Title"] = "Голосування";
    var currentUserId = User.FindFirst(System.Security.Claims.ClaimTypes.NameIdentifier)?.Value;
}

@if (TempData["Success"] != null)
{
    <div class="alert alert-success">@TempData["Success"]</div>
}
```

Продовження додатку В

```
@if (TempData["Error"] != null)
{
    <div class="alert alert-danger">@TempData["Error"]</div>
}

<h2>@ViewData["Title"]</h2>

<p>
    <a asp-action="Create" class="btn btn-primary">Створити</a>
</p>

<table class="table">
    <thead>    <tr>
            <th>Заголовок</th>
            <th>Опис</th>
            <th>Дата початку</th>
            <th>Дата завершення</th>
            <th>Дії</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>@item.Title</td>
                <td>@item.Description</td>
                <td>@item.StartDateTime.ToString("g")</td>
                <td>@item.EndDateTime.ToString("g")</td>
                <td>
                    <form asp-action="Delete" method="post">
                        <input type="hidden" name="id" value="@item.Id" />
                        @if (item.EndDateTime > DateTime.Now && item.StartDateTime < DateTime.Now)
                        {
                            <a asp-action="Evaluate" asp-route-id="@item.Id" class="btn btn-success mb-1">Оцінити</a>
                        }
                        <a asp-action="Results" asp-route-id="@item.Id" class="btn btn-info mb-1">Результати</a>
                        @if (item.UserId == currentUserId)
                        {
                            <button type="submit" class="btn btn-danger mb-1">Видалити</button>
                        }
                    </form>
                </td>
            </tr>
        }
    </tbody>
</table>
@model VoteResultViewModel
@inject UserManager<DBApplicationUser> UserManager
@{
    ViewData["Title"] = "Results: " + ViewBag.VoteTitle;
    var totalVotes = Model.DBVoteItem.Count();
    var uniqueUsers = Model.DBVoteItem.Select(v => v.UserId).Distinct().Count();
}

<h2>Результати: @Model.Title</h2>
<div class="row">
    <div class="col-md-12">
        <div class="card mb-4">
            <div class="card-header">
                <h4>Підсумок голосування</h4>
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-md-3">
                        <strong>Загальна кількість голосів:</strong> @uniqueUsers
                    </div>
```

Продовження додатку В

```
<div class="col-md-3">
                    <strong>Дата початку:</strong> @Model.DBVote.StartDateTime.ToString("MMM dd, yyyy hh:mm:ss")
                </div>
                <div class="col-md-3">
                <strong>Дата закінчення:</strong> @Model.DBVote.EndDateTime.ToString("MMM dd, yyyy hh:mm:ss")
                </div>
                <div class="col-md-3">
                    <strong>Статус:</strong>
                    <span class="badge @(DateTime.Now < Model.DBVote.EndDateTime ? "bg-success" : "bg-secondary")">
                        @(DateTime.Now < Model.DBVote.EndDateTime ? "Активне" : "Завершене")
                    </span>
                </div>
            </div>
            @if (!string.IsNullOrEmpty(Model.DBVote.Description))
            {
                <div class="mt-2">
                    <strong>Опис:</strong> @Model.DBVote.Description
                </div>
            }
        </div>
    </div>
    @if (Model.DBVoteAlternative.Any())
    {
        @if(Model.DBVoteItemSettings.Count() == 0){
            <div class="card mb-4">
  <div class="card-header">
    <h4>Ранжування альтернатив за пріоритетом</h4>
  </div>
  <div class="card-body">
    <div class="table-responsive">
      <table class="table table-striped">
        <thead>
          <tr>
            <th>Місце</th>
            <th>Альтернатива</th>
          </tr>
      </thead>
        <tbody>
          @{
            var priorityRankings = Model.DBVoteItem
              .Where(v => v.AlternativePriority.HasValue)
              .GroupBy(v => v.DBVoteAlternativeId)
              .Select(g => new
              {
                AlternativeId = g.Key,
                Alternative = Model.DBVoteAlternative.FirstOrDefault(a => a.Id == g.Key),
                AveragePriority = g.Average(v => v.AlternativePriority.Value),
                VoteCount = g.Count(),
                MinPriority = g.Min(v => v.AlternativePriority.Value),
                MaxPriority = g.Max(v => v.AlternativePriority.Value)
              })
              .Where(a => a.Alternative != null)
              .OrderBy(a => a.AveragePriority)
              .ToList();

            int rank = 1;
            foreach (var item in priorityRankings)
            {
              <tr>
                <td>@rank</td>
                <td>@item.Alternative.Title</td>
              </tr>
              rank++;
            }
          }
        </tbody>
```

Продовження додатку В

```
</table>
      </div>
   </div>
</div>
        }
        else
        {
        <div class="card mb-4">
           <div class="card-header">
              <h4>Огляд альтернатив</h4>
           </div>
           <div class="card-body">
              <div class="table-responsive">
                 <table class="table table-striped">
                    <thead>
                       <tr>
                          <th>Альтернатива</th>
                          <th>Середньозважений бал</th>
                          <th>Кількість голосів</th>
                       </tr>
                    </thead>
                    <tbody>
                       @foreach (var alternative in Model.DBVoteAlternative)
                       {
                          var alternativeVotes = Model.DBVoteItem.Where(v => v.DBVoteAlternativeId == alternative.Id).ToList();
                          var distinctUsers = alternativeVotes.Select(v => v.UserId).Distinct().Count();

                          // Calculate weighted average score considering importance
                          double weightedSum = 0;
                          double totalWeight = 0;

                          foreach (var vote in alternativeVotes)
                          {
                             // Get the criterion settings for this vote
                    var criterionSettings = Model.DBVoteItemSettings.FirstOrDefault(s => s.Id == vote.DBVoteItemSettingsId);
                             if (criterionSettings != null)
                             {
                                // Use the importance value from settings (default) or from user if overridden
                                double importance = criterionSettings.ImportanceValue.GetValueOrDefault(100);
                                double normalizedImportance = importance / 100.0; // Convert to 0-1 scale

                                weightedSum += vote.Value.GetValueOrDefault() * normalizedImportance;
                                totalWeight += normalizedImportance;
                             }
                          }

                          var weightedAverageScore = totalWeight > 0 ? weightedSum / totalWeight : 0;

                          <tr>
                             <td>@alternative.Title</td>
                             <td>
                                <div class="progress">
                                   <div class="progress-bar" role="progressbar"
                                      style="width: @Math.Ceiling((weightedAverageScore * 10))%"
                                      aria-valuenow="@weightedAverageScore"
                                      aria-valuemin="0"
                                      aria-valuemax="10">
                                     @weightedAverageScore.ToString("F2")
                                   </div>
                                </div>
                             </td>
                             <td>@distinctUsers</td>
                          </tr>
                       }
                    </tbody>
                 </table>
```

Продовження додатку В

```
</div>
            </div>
        </div>
        }
    }

    <!-- Detailed Results by Vote Item -->
    @if(Model.DBVoteItemSettings.Count() > 0)
    {
    <div class="card mb-4">
        <div class="card-header">
            <h4>Детальні результати за критеріями</h4>
        </div>
        <div class="card-body">
            @foreach (var itemSetting in Model.DBVoteItemSettings)
            {
                var itemVotes = Model.DBVoteItem.Where(v => v.DBVoteItemSettingsId == itemSetting.Id).ToList();
                var averageValue = itemVotes.Any() ? itemVotes.Average(v => v.Value) : 0;
                var minValue = itemVotes.Any() ? itemVotes.Min(v => v.Value) : 0;
                var maxValue = itemVotes.Any() ? itemVotes.Max(v => v.Value) : 0;

                <div class="vote-item mb-4 p-3 border rounded">
                    <h5>@itemSetting.Title</h5>
                    @if (!string.IsNullOrEmpty(itemSetting.Description))
                    {
                        <p class="text-muted">@itemSetting.Description</p>
                    }

                    <div class="row">
                        <div class="col-md-6">
                            <strong>Статистика:</strong>
                            <ul class="list-unstyled">
                             <li>Середнє значення: <strong>@averageValue.GetValueOrDefault().ToString("F2")</strong></li>
                                <li>Діапазон значень: @minValue.GetValueOrDefault().ToString("F2") -
@maxValue.GetValueOrDefault().ToString("F2")</li>
                                <li>Кількість голосів: @itemVotes.Count</li>
                            </ul>
                        </div>
                        @if (itemSetting.ImportanceValue != 100)
                        {
                            <div class="col-md-6">
                                <strong>Важливість:</strong> @itemSetting.ImportanceValue
                            </div>
                        }
                    </div>

                    @if (Model.DBVoteAlternative.Any())
                    {
                        <div class="mt-3">
                            <strong>Розподіл за альтернативами:</strong>
                            <div class="table-responsive mt-2">
                                <table class="table table-sm table-bordered">
                                    <thead>
                                        <tr>
                                            <th>Альтернатива</th>
                                            <th>Середнє значення</th>
                                            <th>Кількість голосів</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        @foreach (var alternative in Model.DBVoteAlternative)
                                        {
                                            var altVotes = itemVotes.Where(v => v.DBVoteAlternativeId == alternative.Id).ToList();
                                            var altAverage = altVotes.Any() ? altVotes.Average(v => v.Value) : 0;
                                     <tr>
                                            <td>@alternative.Title</td>
```

Продовження додатку В

```
<td>@altAverage.GetValueOrDefault().ToString("F2")</td>
                                    <td>@altVotes.Count</td>
                                </tr>
                            }
                        </tbody>
                    </table>
                </div>
            </div>
        }
        @if (totalVotes != 0)
        {
            <div class="mt-3">
                <strong>Розподіл зачень:</strong>
                <div class="distribution-chart mt-2">
                    @{
                        var valueGroups = itemVotes.GroupBy(v => Math.Round(v.Value.GetValueOrDefault(), 1))
                        .OrderBy(g => g.Key)
                        .Select(g => new { Value = g.Key, Count = g.Count() })
                        .ToList();
                        var maxCount = valueGroups.Any() ? valueGroups.Max(g => g.Count) : 1;
                    }
                    @foreach (var group in valueGroups)
                    {
                        var percentage = (group.Count * 100) / Math.Max(maxCount, 1);
                        <div class="d-flex align-items-center mb-1">
                            <small class="text-muted" style="width: 40px;">@group.Value</small>
                            <div class="progress flex-grow-1" style="height: 20px;">
                                <div class="progress-bar bg-primary" role="progressbar"
                                    style="width: @percentage%"
                                    aria-valuenow="@group.Count"
                            aria-valuemin="0"
                                    aria-valuemax="@maxCount">
                                    @group.Count
                                </div>
                            </div>
                        }
                    </div>
                </div>
            }
        </div>
    }
    <!-- User Participation Summary -->
    @if (!Model.IsPrivate)
    {
        <div class="card">
            <div class="card-header">
                <h4>Підсумки участі</h4>
            </div>
            <div class="card-body">
                @{
                    var votesPerUser = Model.DBVoteItem.GroupBy(v => v.UserId)
                    .Select(g => new { UserId = g.Key, VoteCount = g.Count() })
                    .OrderByDescending(x => x.VoteCount)
                    .ToList();
                }
                <div class="row">
                    <div class="col-md-12">
                        <strong>Загальна кількість учасників:</strong> @Model.UserIDs.Distinct().Count()
                    </div>
                    <div class="col-md-12">
                        <strong>Кількість учасників, які прийняли участь в опитуванні:</strong> @uniqueUsers
                    </div>
```

Продовження додатку В

```
</div>
                <div class="mt-3">
                @if (votesPerUser.Any())
                {
                   <strong>Учасники, які взяли участь в опитуванні:</strong>
                   <div class="table-responsive mt-2">
                      <table class="table table-sm">
                         <thead>
                            <tr>
                               <th>Ім'я</th>
                               <th>Ім'я користувача</th>
                            </tr>
                         </thead>
                         <tbody>
                            @foreach (var user in votesPerUser)
                            {
                               var u = UserManager.FindByIdAsync(user.UserId).Result;
                               <tr>
                                  <td>@u.FirstName @u.LastName</td>
                                  <td>@u.UserName</td>
                               </tr>
                            }
                         </tbody>
                      </table>
                   </div>
                }
                   <strong>Усі учасники:</strong>
                   <div class="table-responsive mt-2">
                     <table class="table table-sm">
                         <thead>
                            <tr>
                               <th>Ім'я</th>
                               <th>Ім'я користувача</th>
                            </tr>
                         </thead>
                         <tbody>

                            @foreach (var id in Model.UserIDs.Distinct())
                            {
                               var u = UserManager.FindByIdAsync(id).Result;
                               <tr>
                                  <td>@u.FirstName @u.LastName</td>
                                  <td>@u.UserName</td>
                               </tr>
                            }
                         </tbody>
                      </table>
                   </div>

                </div>

             </div>
          </div>
          <!-- All Votes Table -->
          <div class="card mb-4">
    <div class="card-header">
       <h4>Усі голоси</h4>
    </div>
    <div class="card-body">
       <div class="table-responsive">


          <table class="table table-striped table-bordered">
                   @{
                      var isRanking = Model.DBVoteItemSettings.Count() > 0;
                   }
```

Продовження додатку В

```
<thead class="table-dark">
            <tr>
              <th>Користувач</th>
              <th>Альтернатива</th>
              <th @(!isRanking ? "hidden":"")>Критерій</th>
              <th @(!isRanking ? "hidden":"")>Оцінка</th>
              <th @(!isRanking ? "hidden":"")>Важливість</th>
              <th @(isRanking ? "hidden":"")>Приорітет альтернативи</th>
            </tr>
          </thead>
          <tbody>
            @foreach (var vote in Model.DBVoteItem.OrderBy(v => v.DBVoteAlternativeId).ThenBy(v => v.DBVoteItemSettingsId))
            {
              var user = await UserManager.FindByIdAsync(vote.UserId);
              var alternative = Model.DBVoteAlternative.FirstOrDefault(a => a.Id == vote.DBVoteAlternativeId);
              var itemSetting = Model.DBVoteItemSettings.FirstOrDefault(s => s.Id == vote.DBVoteItemSettingsId);

              <tr>
                <td>
                  @if (user != null)
                  {
                    <text>@user.FirstName @user.LastName (@user.UserName)</text>
                  }
                  else
                  {
                    <text>Невідомий користувач</text>
                  }
                </td>
               <td>@(alternative?.Title ?? "Невідома альтернатива")</td>
                <td @(!isRanking ? "hidden":"")>@(itemSetting?.Title ?? "Невідомий критерій")</td>
                <td @(!isRanking ? "hidden":"")>
                  <span class="badge bg-primary">@vote.Value.GetValueOrDefault().ToString("F1")</span>
                </td>
                <td @(!isRanking ? "hidden":"")>
                  @{
                    var importanceValue = itemSetting?.ImportanceValue != null ? itemSetting.ImportanceValue.ToString() : "-";
                  }
                  <span class="badge bg-info">@importanceValue</span>
                </td>
                      <td @(Model.DBVoteItemSettings.Count() > 0 ? "hidden" : "")>
                  @(vote?.AlternativePriority != null ? vote?.AlternativePriority : "")
                </td>
              </tr>
            }
          </tbody>
        </table>
      </div>

      <!-- Summary statistics -->
      <div class="row mt-3">
        <div class="col-md-3">
          <div class="card text-center">
            <div class="card-body">
              <h5 class="card-title">@Model.DBVoteItem.Count()</h5>
              <p class="card-text">Кількість оцінок</p>
            </div>
          </div>
        </div>
        <div class="col-md-3">
          <div class="card text-center">
            <div class="card-body">
              <h5 class="card-title">@Model.DBVoteItem.Select(v => v.UserId).Distinct().Count()</h5>
              <p class="card-text">Кількість користувачів, які проголосували</p>
            </div>
          </div>
        </div>
```

Продовження додатку В

```
<div class="col-md-3">
            <div class="card text-center">
               <div class="card-body">
                  <h5 class="card-title">@Model.DBVoteAlternative.Count()</h5>
                  <p class="card-text">Альтернатив</p>
               </div>
            </div>
         </div>
         <div class="col-md-3">
            <div class="card text-center">
               <div class="card-body">
                  <h5 class="card-title">@Model.DBVoteItemSettings.Count()</h5>
                  <p class="card-text">Критеріїв</p>
               </div>
            </div>
         </div>
      </div>
   </div>
</div>
@if(Model.DBVoteItemSettings.Count() > 0)
{

<!-- Alternative Votes Summary -->
<div class="card mb-4">
      <h4>Підсумок голосів за альтернативами</h4>
   </div>
   <div class="card-body">
      <div class="table-responsive">
         <table class="table table-hover">
            <thead>
               <tr>
                  <th>Альтернатива</th>
                  <th>Кількість голосів</th>
                  <th>Середній бал</th>
                  <th>Унікальні користувачі</th>
                  <th>Деталі</th>
               </tr>
            </thead>
            <tbody>
               @foreach (var alternative in Model.DBVoteAlternative)
               {
                  var alternativeVotes = Model.DBVoteItem.Where(v => v.DBVoteAlternativeId == alternative.Id).ToList();
                  var distinctUsers = alternativeVotes.Select(v => v.UserId).Distinct().Count();
                  var averageScore = alternativeVotes.Any() ? alternativeVotes.Average(v => v.Value) : 0;
                  <tr>
                     <td><strong>@alternative.Title</strong></td>
                     <td>@alternativeVotes.Count</td>
                     <td>
                        <div class="d-flex align-items-center">
                           <div class="progress flex-grow-1 me-2" style="height: 10px;">
                              <div class="progress-bar bg-success"
                                 style="width: @Math.Ceiling((averageScore.GetValueOrDefault() * 10))%"
                                 title="@averageScore.GetValueOrDefault().ToString("F2")">
                              </div>                              </div>
                           <span>@averageScore.GetValueOrDefault().ToString("F2")</span>
                        </div>
                     </td>
                     <td>@distinctUsers</td>
                     <td>
                        <button class="btn btn-sm btn-outline-primary"
                           type="button"
                           data-bs-toggle="collapse"
                           data-bs-target="#collapseAlternative@(alternative.Id)"
                           aria-expanded="false"
                           aria-controls="collapseAlternative@(alternative.Id)">
                           Показати деталі
```

Продовження додатку В

```
 </button>
                    </td>
                </tr>
                <tr class="collapse-row">
                    <td colspan="5" class="p-0">
                        <div class="collapse" id="collapseAlternative@(alternative.Id)">
                            <div class="p-3 bg-light">
                                <h6>Детальні результати для: @alternative.Title</h6>
                                <div class="table-responsive">
                                    <table class="table table-sm">
                                        <thead>
                                            <tr>
                                                <th>Критерій</th>
                                                <th>Кількість голосів</th>
                                                <th>Середній бал</th>
                                                <th>Мін.</th>
                                                <th>Макс.</th>
                                            </tr>
                                        </thead>                                    <tbody>
                                            @foreach (var itemSetting in Model.DBVoteItemSettings)
                                            {
                                                var criterionVotes = alternativeVotes.Where(v => v.DBVoteItemSettingsId == itemSetting.Id).ToList();
                                                var criterionAvg = criterionVotes.Any() ? criterionVotes.Average(v => v.Value) : 0;
                                                var criterionMin = criterionVotes.Any() ? criterionVotes.Min(v => v.Value) : 0;
                                                var criterionMax = criterionVotes.Any() ? criterionVotes.Max(v => v.Value) : 0;

                                                <tr>
                                                    <td>@itemSetting.Title</td>
                                                    <td>@criterionVotes.Count</td>
                                                    <td>@criterionAvg.GetValueOrDefault().ToString("F2")</td>
                                                    <td>@criterionMin.GetValueOrDefault().ToString("F1")</td>
                                                    <td>@criterionMax.GetValueOrDefault().ToString("F1")</td>
                                                </tr>
                                            }
                                        </tbody>
                                    </table>
                                </div>
                            </div>
                        </div>
                    </td>
                </tr>
            }
        </tbody>
    </table>
</div>
</div>
</div>
        }
}
    </div>       background-color: #f8f9fa;
        border-left: 4px solid #007bff !important;
    }
    .distribution-chart .progress {
        background-color: #e9ecef;
    }

    .badge {
        font-size: 0.75em;
    }

    .table th {
        border-top: none;
        font-weight: 600;
    }
</style>
```

Додаток Г (обов'язковий)
Протокол перевірки кваліфікаційної роботи

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Онлайн система для вироблення спільних рішень групою людей»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КПІ) 0.2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Маслій Р.В.
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Кабачій В. В., доц. каф. АІІТ
(прізвище, ініціали, посада)

Здобувач (підпис) Любунь О. І.
(прізвище, ініціали)