Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія багатокористувацького ранжування зображень»

Виконав: студент 2-го курсу, групи 2КН-21м
спеціальності 122 – Комп'ютерні науки
Дикун В. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент
Колодний В. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
«15» 12 2022 р.

Опонент: к. т. н., доц. каф. КСУ
Юхимчук М. С.
(прізвище та ініціали)
«15» 12 2022 р

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри ____

«14» 09 2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дикуну Володимиру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Інформаційна технологія багатокористувацького ранжування зображень»
керівник роботи к. т. н., доцент Колодний В. В,
затверджені наказом вищого навчального закладу від «14» вересня 2022 року № 203
2. Строк подання студентом роботи 18 листопада 2022 року.
3. Вихідні дані до роботи: зручний інтерфейс для взаємодії користувача з системою, мова програмування – можливість веб-розробки; час завантаження WEB-сторінки 1400 мс; час до взаємодії зі сторінкою 2500 мс; кількість ступенів ранжування – не менше 4.
4. Зміст текстової частини: вступ, аналіз сучасного стану розвитку систем багатокористувацького ранжування зображень, проектування інтелектуальної моделі системи багатокористувацького ранжування зображень, програмна реалізація системи багатокористувацького ранжування зображень, висновок, перелік використаної літератури.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема проведення багатокористувацького ранжування, загальна структурна схема систми багатокористувацького ранжування зображень, алгоритм роботи розробленого методу багатокористувацького ранжування зображень, схема інтерфейсу головної сторінки системи багатокористувацького ранжування зображень, загальний вигляд головної сторінки програного модуля багатокористувацького ранжування зображень, загальний вигляд прикладу результату багатокористувацького ранжування зображень.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконан прийня
Спеціальна частина. Розділ 1-3.		14.09.2022	14.09.2022
Економічна частина	Бурбенікова М.В. д.е.н. проф. кафедри ЕПВМ		

7. Дата видачі завдання 14.09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Пр мі
1	Аналіз сучасного стану розвитку систем багатокористувацького ранжування зображень	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1	14.09.20 – 01.10.20
2	Моделювання процесу роботи системи	Моделі, розділ 2	02.10.20 – 11.10.20
3	Розробка системи багатокористувацького ранжування зображень	Розділ 3	12.10.20 – 02.11.20
4	Програмна реалізація системи багатокористувацького ранжування зображень	Розділ 3	08.11.20 – 21.11.20
5	Апробація та/або впровадження результатів дослідження	Тези доповідей/акт впровадження	23.11.20 – 01.12.20
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	Пояснювальна записка, графічний матеріал, презентація	02.12.20 – 14.12.20

Студент _____ (підпис) Дикун В. О.

Керівник роботи _____ (підпис) Колодний

АНОТАЦІЯ

УДК 004.054

Дикун В. О.. Інформаційна технологія багатокористувацького ранжування зображень. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма – системи штучного інтелекту. Вінниця: ВНТУ, 2022. 102 с.

Укр. мовою. Бібліогр.: 21 назв; рис.: 22; табл. 9.

Дана магістерська кваліфікаційна робота присвячена розробці інформаційної технології та програмного забезпечення для багатокористувацького ранжування зображень. Удосконалено математичну модель багатокористувацького ранжування зображень. Розроблений програмний продукт, призначений для багатокористувацького ранжування. Для розробки проекту використано програмне середовище Atom, мова програмування JavaScript. Для розробки застосунку використано бібліотеку React, для збереження даних використовувалась об’єктно-орієнтована база даних MongoDB. Визначено, що при використанні даної технології, результати ефективності збільшуються у щонайменше 1,2 рази. Графічна частина складається з 7 плакатів із результатами проектування та реалізації.

Ключові слова: ранжування; прийняття рішень; зображення; веб-додаток.

ABSTRACT

UDC 004.054

Dykun V. O. Information technology of multiuser image ranking. Master's thesis on specialty 122 - computer science, educational program - artificial intelligence systems. Vinnytsia: VNTU, 2022. 102 p.

In Ukrainian speech Bibliography: 21 titles; Fig.: 22; table 9.

This Master's thesis is dedicated to the development of information technology and software for multiuser image ranking. A mathematical model and ranking method were developed. A software product for multi-user ranking has been developed. The Atom software environment and the JavaScript programming language were used to develop the project. The React library was used to develop the application, and the MongoDB object-oriented database was used to store data. It was determined that when using this technology, the efficiency results increase by at least 1.2 times. The graphic part consists of 7 posters with the results of design and implementation.

Keywords: ranking; decision-making; image; web application.

Зміст

ВСТУП

Зі швидким зростанням обсягів інформації навколо нас та бурхливим розвитком інформаційних технологій вміння швидко прийняти правильне рішення стає все важливішим. Під постійним тиском надлишку інформації часто важко зосередитись та утримати в голові думки й ідеї, правильно оцінити їх. Ключ до уникнення помилок полягає в тому, щоб мати добре організований і ефективний спосіб обробки інформації. Швидкість, з якою ми можемо приймати рішення, залежить від кількох факторів, наприклад від стану розуму, попереднього досвіду і знання ситуації – але крім цього її підвищити можуть ефективніші методи та технології [1].

Одними з-поміж найскладніших є неструктуровані задачі прийняття рішень в умовах невизначеності. Очевидно, що безглуздо залучати експертів для визначення «точних» оцінок варіантів за критеріями та шкалами, якщо ухвалення рішення здійснюється в умовах невизначеності щодо цілей та критеріїв.

У такому разі набагато краще спрацює звернення до особи (осіб), яка приймає рішення, з простими запитаннями, що не вимагають значних інтелектуальних зусиль та не марнують багато часу, тобто варто активізувати швидку систему інтуїтивного мислення. Інтуїція, тобто здатність людини несвідомо вловлювати істину, вгадувати щось, спираючись на попередній досвід та знання, є найважливішим способом швидкого прийняття рішень [1, 2].

Групове прийняття рішень – це ситуація, коли люди колективно роблять вибір із альтернатив, що перед ними стоять. Група повинна спільно прийняти рішення на основі представленої інформації та думок її членів [3]. Це можна зробити різними способами, але найпоширенішим є голосування. Зокрема преференційне голосування, тобто не вибір однієї альтернативи, а подача ранжування виборів, є цікавою перспективою, тому методи

особистого прийняття рішень за допомогою ранжування можна використати для групової стратегії.

Актуальність теми дослідження полягає в оригінальному поданні програмного модуля для багатокористувацького ранжування зображень.

Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей програмних засобів по обобці зображень за допомогою розробки інформаційної технології багатокористувацького ранжування зображень за якістю. Для досягнення поставленої мети необхідно розв’язати такі **завдання**:

- проаналізувати сучасний стан предметної області багатокористувацького ранжування зображень;
- провести аналіз відомих технічних рішень багатокористувацького ранжування зображень;
- провести порівняльний аналіз характеристики програмних систем багатокористувацького ранжування зображень;
- удосконалити математичну модель багатокористувацького ранжування зображень;
- розробити метод багатокористувацького ранжування зображень;
- спроектувати інтелектуальну модель системи багатокористувацького ранжування зображень;
- програмно реалізувати систему багатокористувацького ранжування зображень;
- протестувати систему багатокористувацького ранжування зображень;
- розрахувати економічну ефективність науково-технічної розробки за її можливої комерціалізації потенційним.

Об’єктом дослідження є процес ранжування зображень за якістю.

Предметом дослідження є програмні засоби ранжування зображень за якістю.

Методи дослідження – методи та засоби розробки систем багатокористувацького ранжування зображень, методи теорії прийняття рішень, методи ранжування, методи обчислення результатів, методи та підходи до розробки застосунків.

Наукова новизна отриманих результатів полягає в наступному:

Розроблено інформаційну технологію багатокористувацького ранжування зображень, що відрізняється від наявних використанням моделі поєднання методів особистого ранжування та спільного преференційного голосування, що дає змогу підвищити швидкодію процесу ранжування.

Практичне значення отриманих результатів полягає у наступному:

1. Удосконалено алгоритм багатокористувацького ранжування зображень.
2. Розроблено програмне забезпечення багатокористувацького ранжування зображень, що відрізняється від наявних поєднанням особистого ранжування та спільного преференційного голосування.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з теорією та збігом результатів математичного моделювання з отриманими під час впроваджування розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, що наведені в магістерській кваліфікаційній роботі, отримані магістрантом самостійно.

Апробація результатів роботи. Результаті дослідження апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН2023)», Вінниця [1].

Публікації. За результатами дослідження опубліковані одні тези доповіді [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ БАГАТОКОРИСТУВАЦЬКОГО РАНЖУВАННЯ ЗОБРАЖЕНЬ

1.1 Аналіз предметної області багатокористувацького ранжування зображень

Прийняття рішень (також ухвалення рішень; decision-making) – це особливий процес людської діяльності, який спрямований на вибирання найкращої альтернативи. Прийняття рішень трактують як когнітивний процес, результатом якого є вибір переконань чи дій серед інших можливих варіантів. У буденному житті воно може бути більш чи менш раціональним, спиратися на особисті вподобання, цінності, явне та неявне знання різною мірою тощо [2].

Під найкращим варіантом розуміють варіант, який призведе до отримання найкращого результату. Для переважної більшості рішень, які приймає людина, неможливо точно розрахувати і оцінити всі наслідки. Можна лише припускати, що вибір певного варіанта дій в майбутньому приведе до деякого найкращого результату. Тому зазвичай важливо оцінити можливі наслідки своїх дій і прийняти рішення, виходячи з власних конкретних цінностей та потреб. Але, як і з іншими задачами, важливо не марнувати час та ресурси дарма – для простих виборів часто достатньо швидкої реакції.

Під час процесу прийняття рішень люди можуть відігравати різні ролі. Головною дійовою особою є особа, яка приймає рішення (ОПР) – людина, яка фактично здійснює вибір варіанта дій. Ця особа (або колектив) має мету, яка служить мотивом пошуків розв’язку проблеми.

В будь-яких процесах прийняття рішень можна виділити три етапи:

1) пошук інформації;

2) пошук та знаходження множини допустимих альтернатив;

3) вибір найкращої альтернативи.

Перші два етапи є слабоформалізованими, але на третьому етапі можливе застосування наукових підходів.

В загальному випадку проблеми прийняття рішення можуть бути структуризовані, тобто описані, такою п’ятіркою:

<Q, A, I, Ψ, D>,

де Q – це критерії;

A – множина альтернатив;

I – наявна інформація;

Ψ – особливості ОПР;

D – правила вибору (прийняття) рішення [2].

Критерії оцінювання альтернатив – це показники їхньої привабливості або непривабливості для учасників процесу вибору. Альтернативи – це варіанти дій, які може вибирати ОПР. Для постановки задачі прийняття рішень потрібно мати не менше двох альтернатив, максимум – не обмежений.

Ранг – ступінь відмінності за будь-якою ознакою, а ранжування – процес визначення рангів, відносних кількісних оцінок ступенів відмінностей, впорядкування множини альтернатив. Ранжування застосовується у випадках, коли неможлива або недоцільна безпосередня оцінка [4].

Метод попарних порівнянь полягає у визначенні переваг у парах елементів. Це часто зображають як порівняння елементів, розташованих у стовпці з елементами, розташованими у рядках таблиці. При цьому складається матриця, в рядках і стовпцях якої мають порівнювані об'єкти.

У випадку некретиріального оцінювання альтернатив перспективним є застосування гештальт-ранжування, тобто одночасне пропонування ОПР для швидкого інтуїтивного впорядкування певного числа випадкового обраних альтернатив [5].

Для ОПР найпростішим і найнадійнішим буде одночасне порівняння щонайменшої кількості об’єктів – двох, яке добре описано апаратом бінарних відношень та попарних порівнянь. Щоправда, такий підхід не буде враховувати ефект контексту вибору, коли включення або виключення з множини альтернатив деякого варіанту істотно впливає на вибір.

Також можна розглядати кількість рівнів порівняння. Що менше їх, то менше зусиль потрібно, проте зменшується точність. Дедалі більша кількість уточнюватиме порівняння, але будь-яка інтервальна шкала не уможливить визначення у скільки разів одна альтернатива краща за іншу – для такого треба використовувати числову оцінку, тобто пропорційну шкалу.

Групове прийняття рішень (спільне прийняття рішень, колективне прийняття рішень) – це ситуація, з якою люди стикаються, коли колективно роблять вибір з наявних перед ними альтернатив. Після цього рішення більше не приписується жодному окремому члену групи. Це відбувається тому, що всі індивіди та соціальні групові процеси, наприклад соціальний вплив, сприяють результату. Рішення, прийняті групами, часто відрізняються від рішень, прийнятих окремими людьми. На робочому місці спільне прийняття рішень є однією з найуспішніших моделей для залучення зацікавлених сторін, досягнення консенсусу та заохочення творчості. Згідно з ідеєю синергії, рішення, прийняті колективно, також мають тенденцію бути ефективнішими, ніж рішення, прийняті окремою особою [3].

У цьому ключі певні домовленості про співпрацю можуть генерувати кращі чисті результати, ніж окремі особи, які діють самостійно. За звичайних повсякденних умов спільне або групове прийняття рішень часто буде кращим і приносить більше переваг, ніж індивідуальне прийняття рішень, але за умови наявності часу для належного обдумування, обговорення та діалогу. Цього можна досягти за допомогою комітету, команд, груп, партнерства або інших групових соціальних процесів.

Проте у деяких випадках групове ухвалення рішень може мати і недоліки. У екстремальних надзвичайних ситуаціях або кризових ситуаціях інші форми прийняття рішень можуть бути кращими, оскільки надзвичайні дії можуть бути вжиті швидше з меншим часом на обдумування. Також під час оцінки доцільності системи прийняття рішень також необхідно враховувати додаткові міркування. Наприклад, часом може виникати можливість групової поляризації, що спонукає групи приймати екстремальніші рішення у напрямку індивідуальних вподобань, ніж рішення окремих її членів. Існують також інші приклади, коли рішення, прийняті групою, є помилковими, наприклад, існує ефект групового мислення, за якого в згуртованій групі домінує пошук згоди, а не реалістична оцінка можливих альтернативних дій [6].

Фактори, які впливають на поведінку інших соціальних груп, також впливають на колективи, що ухвалюють групові рішення. Наприклад, було помічено, що групи з високим рівнем згуртованості в поєднанні з іншими попередніми умовами (наприклад, ідеологічною однорідністю та ізоляцією від незгодних думок) негативно впливають на групове прийняття рішень і, отже, на ефективність групи. Крім того, коли люди приймають рішення як частина групи, існує тенденція проявляти упередженість приділяти більше часу обговоренню відомої усім інформації і менше тій, яку знають не всі.

Преференційне голосування ((з)ранжоване, рейтингове; ranked voting) – це будь-яка система голосування, за якої виборці розташовують своїх кандидатів (або варіанти) по порядку у своїх відповідних бюлетенях. Системи такого голосування відрізняються залежно від того, як позначаються бюлетені, як зводяться та підраховуються вподобання, скільки місць заповнено та чи дозволено виборцям давати однаковий ранг кандидатам. Виборча система, яка використовує рейтингове голосування, використовує один із багатьох доступних методів підрахунку для вибору кандидата або кандидатів-переможців. Виборчі системи з рейтинговим голосуванням також відрізняються тим, що в деяких системах рейтингового

голосування офіційні особи вимагають від виборців ранжувати певну кількість кандидатів, іноді всіх; в інших громадяни можуть ранжувати стільки кандидатів, скільки вважають за потрібне.

Преференційне голосування дає змогу виборцям точніше відображати їхні вподобання, ніж звичайна система відносної більшості (first-past-the-post voting). Проте вони складніші для учасників. Якщо є велика кількість кандидатів, що є досить поширеним явищем на виборах зі системою єдиного перехідного голосу, то, ймовірно, багато моделей голосування за переваги будуть унікальними для окремих виборців, що може дозволити ідентифікацію виборців за бюлетенем у контексті корупції чи залякування, порушення таємниці голосування.

Зображення – це об’єкт, який передає зорове сприйняття, схоже на певний предмет, тобто подає його в графічній формі. Зображення можуть представляти як конкретні предмети, зображені на ньому, так і складні ідеї, зокрема асоціативно; крім того, саме зображення може бути альтернативою.

1.2 Порівняльний аналіз наявних програмних засобів ранжування зображень

Натепер є кілька інструментів, які дають змогу проранжувати певні альтернативи, проте загалом вони не популярні, часто прив’язані до окремих типів файлів чи галузі, не завжди мають групові опції. Розгляньмо два з них детальніше.

Rcv123 – сайт, який дає змогу безкоштовно створити преференційне голосування [7]. Інструмент дозволяє створити, налаштувати та провести преференційне голосування за текстові варіанти. Як вже згадано, преференційне голосування є методом голосування, насамперед виборного, за якого учасники не просто обирають один варіант, а подають ранжування варіантів згідно зі своїми вподобаннями. Це дозволяє враховувати не тільки основний голос, а й наступні вподобання, тобто допомагає точніше висловити свою думку щодо предмету голосування.

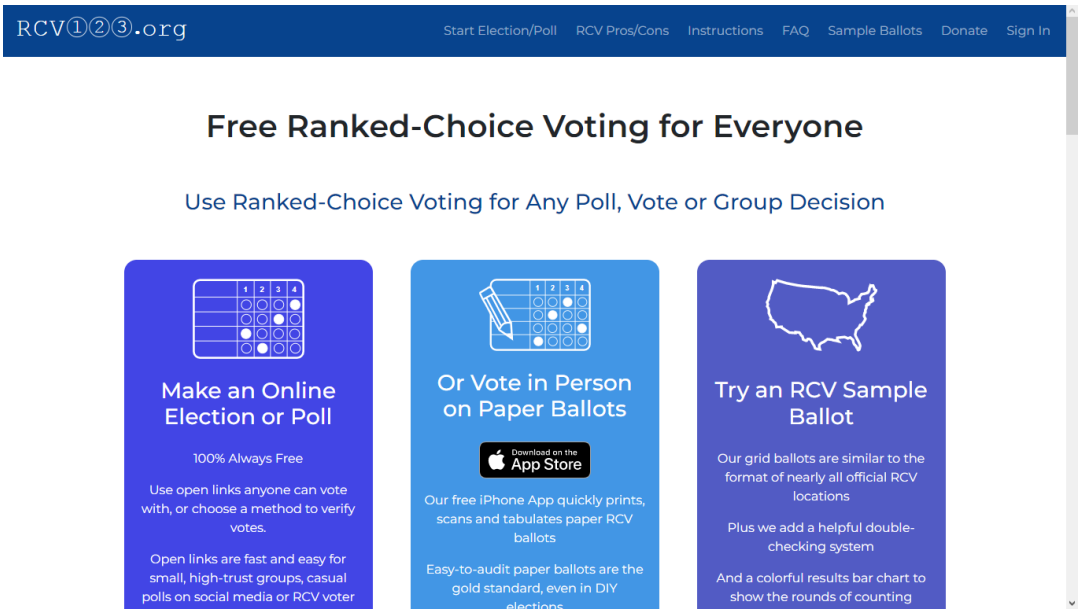


Рисунок 1.1 – Стартова сторінка сайту

Для роботи на сайті потрібно створити та налаштувати голосування. Сайт надає можливість розмістити опитування на відкритому посиланні або ж зі запитом деякої особистої інформації (верифікація). Наступна сторінка, яка зображена на рисунку 1.2, пропонує налаштувати процес, а саме обрати кількість «кандидатів», переможців, ранжованих виборів (для великої кількості варіантів корисним може бути дозвіл обрати й ранжувати лише невелику кількість виборів), а також назву голосування та альтернативи.

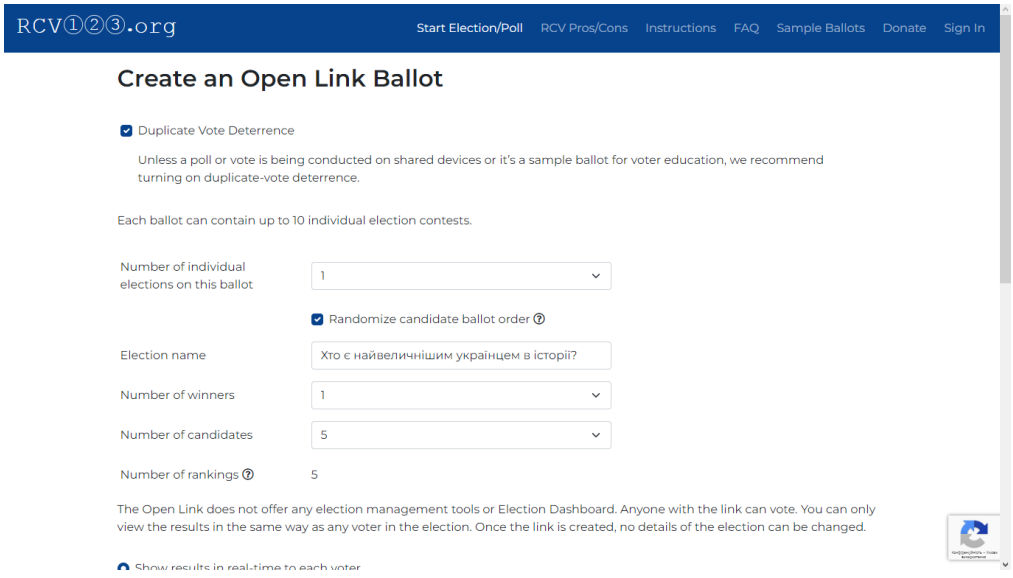


Рисунок 1.2 – Створення голосування

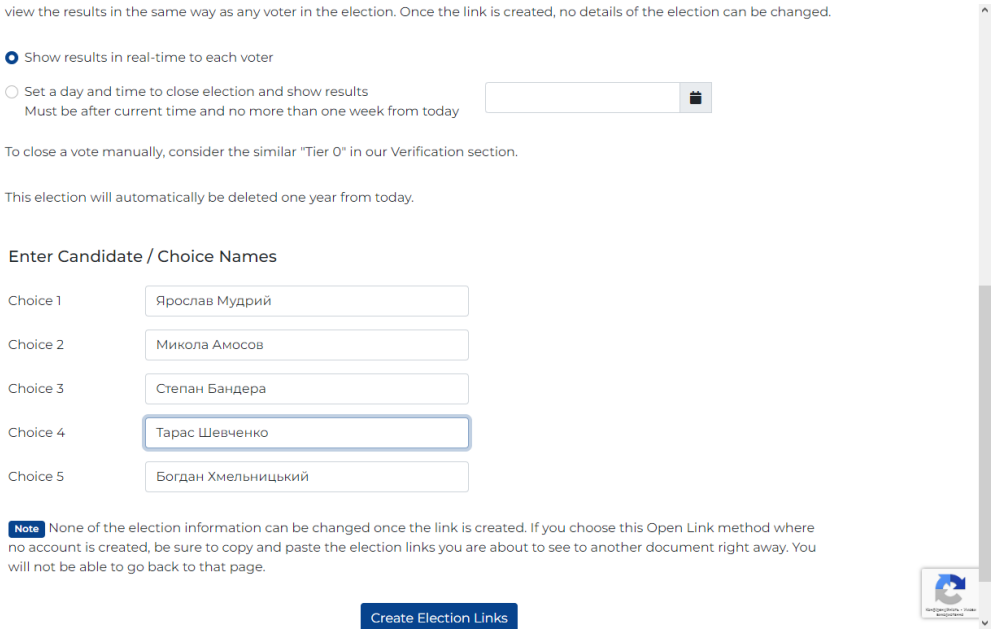


Рисунок 1.3 – Створення варіантів

Далі система надасть посилання для проведення голосування та перегляду результатів, які необхідно поширити серед учасників голосування. За ними можна проголосувати ранжуванням, як показано на рисунку 1.4. У таблиці зліва потрібно відзначити відповідні радіокнопки так, щоб кожній альтернативі в рядках відповідало місце в ранжуванні з стовпчиків.

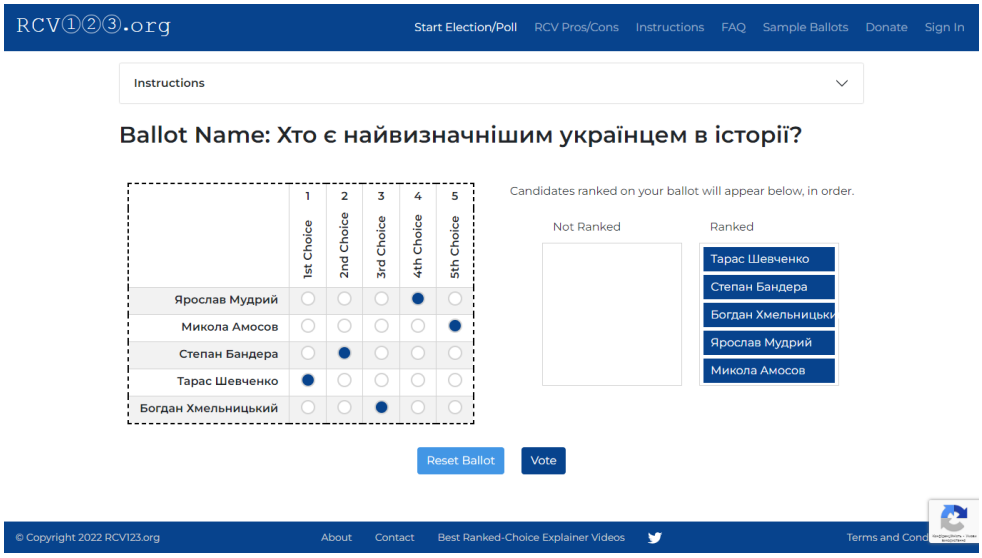


Рисунок 1.4 – Сторінка-бюлетень

За другим посиланням можна побачити результати, як на рисунку 1.5. Згори коротко описано суть способу підрахунку голосів, нижче на сторінці є детальніший аналіз та усі дані.

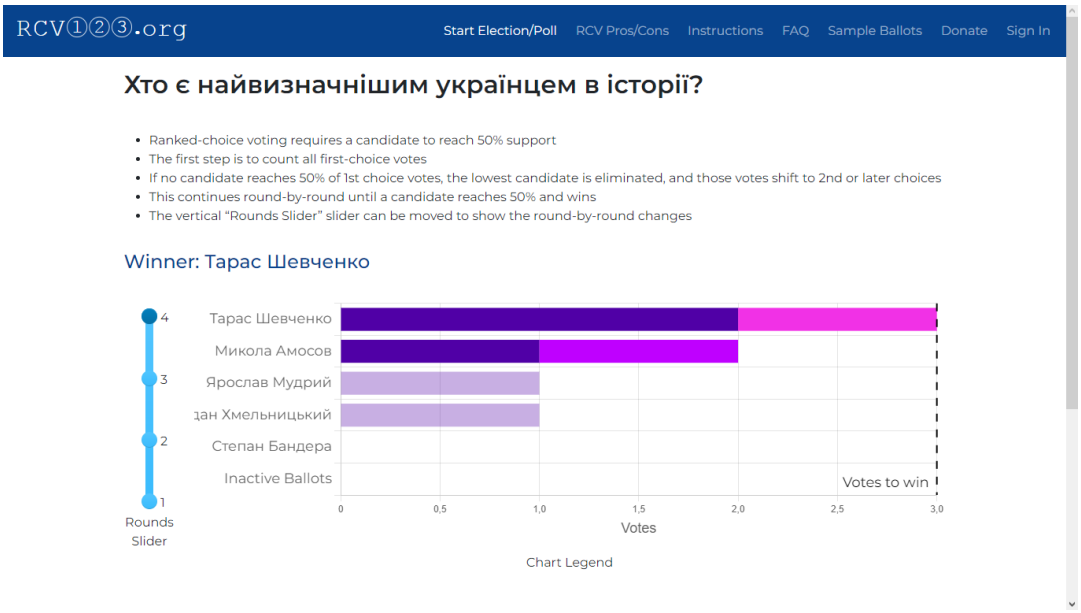


Рисунок 1.6 – Результати голосування

Ranker – відкрите програмне забезпечення у формі комп'ютерної програми, який призначений для ручного ранжування зображень та відео. Інструмент дозволяє завантажити набір зображень та проранжувати їх – редагуючи оцінки окремо чи за допомогою попарних порівнянь. Цей програмний продукт призначений для особистого використання, а не групових рішень [8].

На початковому екрані пусто, для старту потрібно обрати «Завантажити каталог» та обрати папку з зображеннями. Після виконання цього, з'являться доступні картинки (рис 1.5).

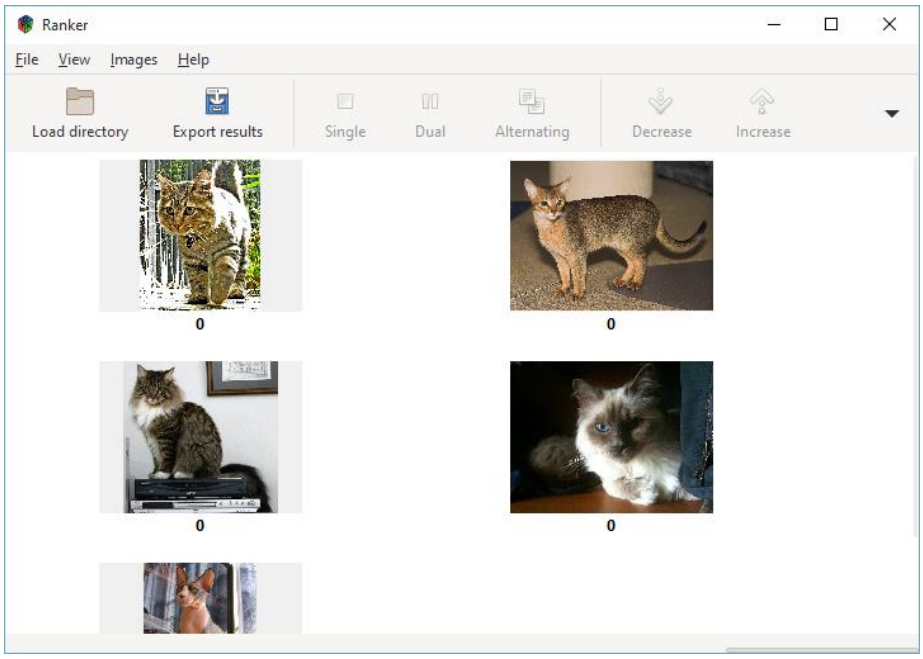


Рисунок 1.5 – Завантажені картинки

Далі користувач може визначити оцінку для кожної картинки окремо за допомогою кнопок на верхній панелі або меню правої клавіші миші (рис. 1.6) або ж обрати ранжування методом попарного порівняння (рис. 1.7).

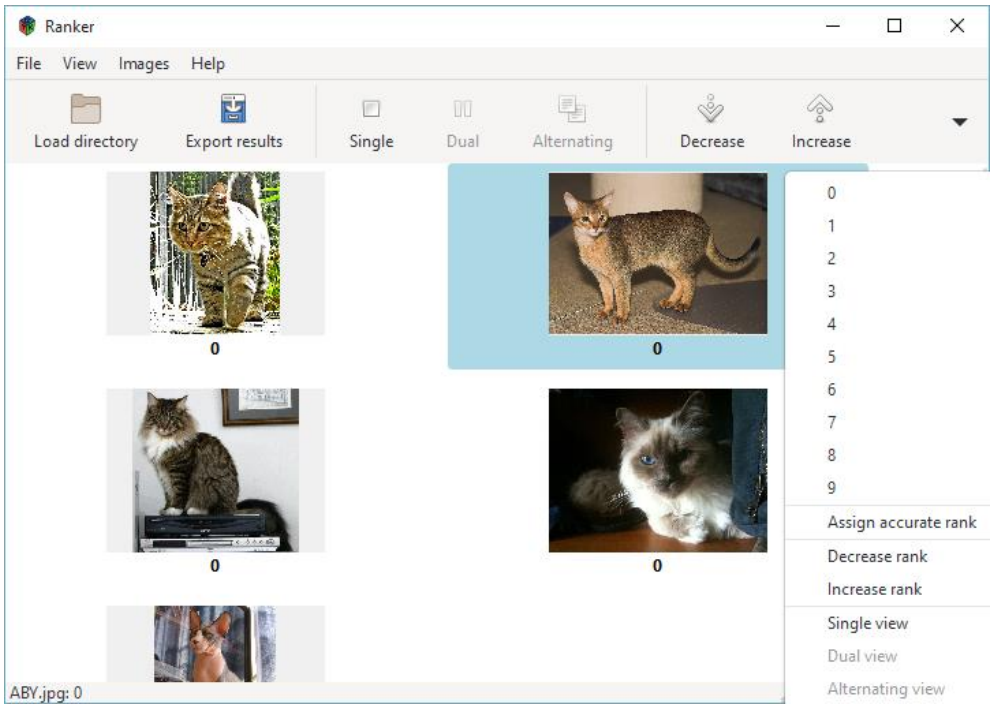


Рисунок 1.6 – Меню правої клавіші миші

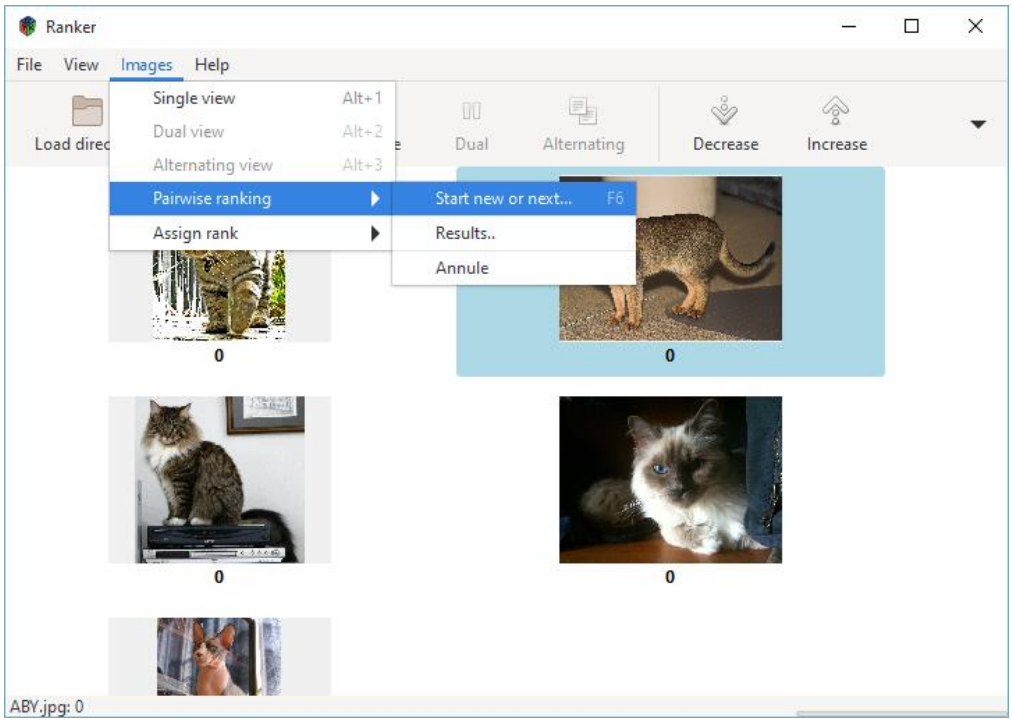


Рисунок 1.7 – Початок попарного ранжування

Під час ранжування на екран виводиться по дві картинки й пропонується обрати одну з них (рис. 1.8).

Рисунок 1.8 – Процес попарного ранжування

Наприкінці користувачу запропонують опції щодо перетворення та збереження результатів (рис. 1.9).

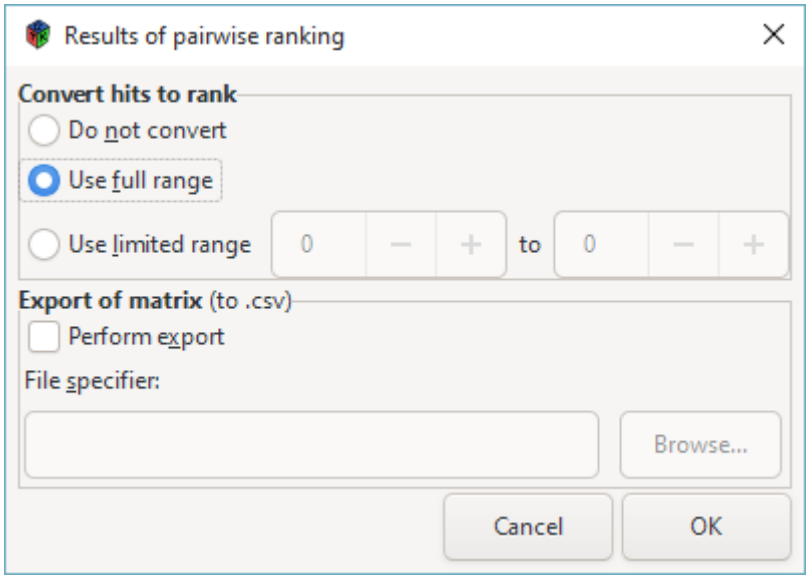


Рисунок 1.9 – Опції перетворення та збереження результаів

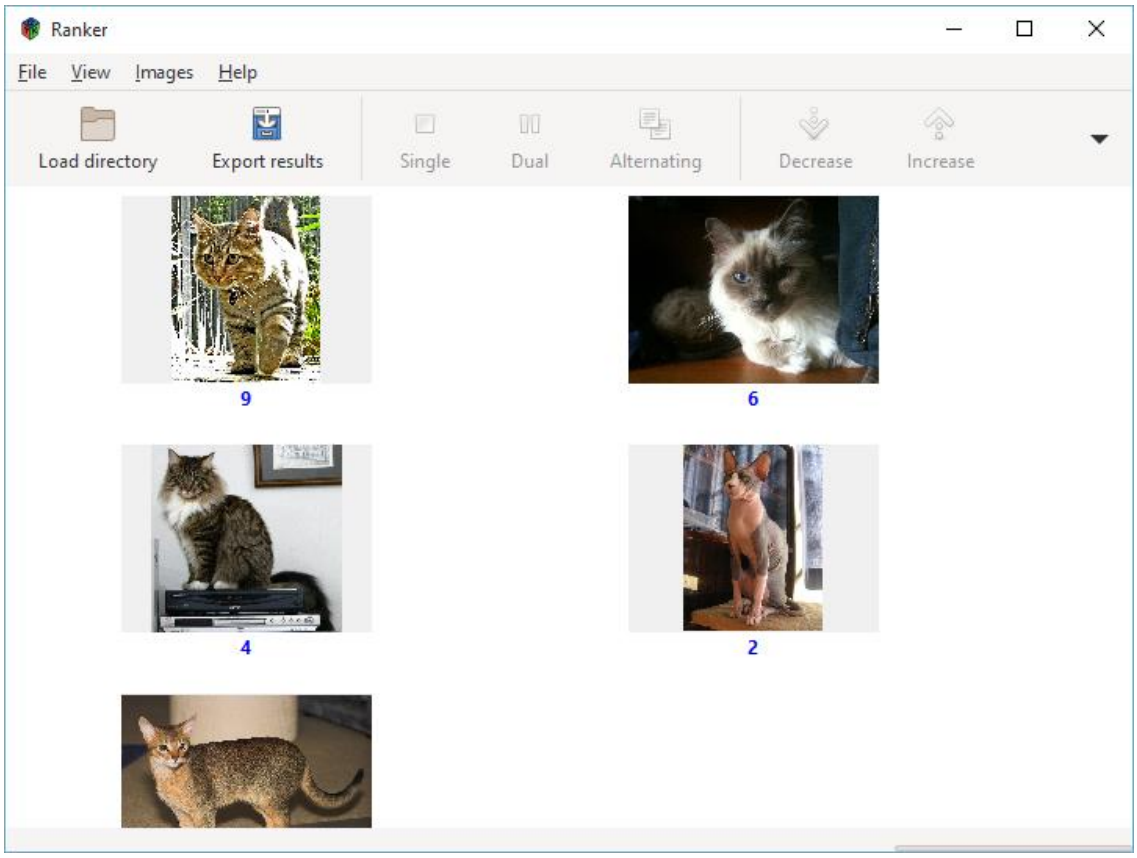


Рисунок 1.10 – Збережені результати ранжування

1.3 Аналіз об’єкту проектування

Основною метою розробки програмного засобу є надання користувачам можливості легкого ранжування та подальшого використання її для надання користувачам вирішення їхніх задач.

Вимоги до програмного модуля:

– Реєстрація користувачів
– Можливість користувачу створювати власні набори зображень
– Доступність ранжування
– Збереження результатів

Це завдання є актуальним з огляду на розвиток інтелектуальної комп’ютерної техніки, що забезпечує вирішення конкретних завдань (консультації, навчання, діагностика, тестування, проектування тощо). Будь-які зображення можуть бути використані як вхідні дані для ранжування.

Платформа для створення експертних систем повинна коректно працювати на сучасних ПК.

Для забезпечення функціонування роботи програмного модуля для багатокористувацького ранжування зображення він повинен оперувати вхідними і вихідними даними. Для нього основними вхідними даними є та інформація яку подає користувач в своєму особистому кабінеті при створенні набору зображень, вихідною – результати ранжування.

1.4 Висновок

В даному розділі було проаналізовано предметну область групове ранжування зображень, визначено принцип її роботи. Проведено порівняння наявних програмних засобів та проведений поверхневий

порівняльний аналіз. Визначена постановка задачі, необхідні критерії та вимоги.

На підставі попередніх аналізів предметної області, програмних засобів, отримано можливість визначити основні проблеми та недоліки. Обрана оптимальна структура середовища, визначено критичні негативні параметри у програм аналогів та поставлені цілі для їх усунення.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ БАГАТОКОРИСТУВАЦЬКОГО РАНЖУВАННЯ ЗОБРАЖЕНЬ

2.1 Математична модель багатокористувацького ранжування зображень

Теорія прийняття рішень — дослідження процесу вироблення та ухвалення рішень. Ця галузь обґрунтовує прийняття найкращих (в певному обраному сенсі) рішень [2].

Головною дійовою особою прийняття рішень є особа, що приймає рішення – але в багатьох випадках нею є група осіб.

Однією з основних функцій групи є вирішення проблем прийняття рішень. Для будь-якої справді невеликої групи важко прийняти єдине рішення з питання, яке має значення для команди. Проте група не може існувати як єдине ціле без спільного підходу і думки з нагальних або принципових питань. Крім того, важливо знайти найбільш ефективне загальне рішення. Прийняття групового рішення необов’язково є результатом групового компромісу, а оптимальним вирішенням важливого для всієї групи та кожного її члена завдання [3].

Найважливішими формальними системами прийняття рішень є:

- Знаходження консенсусу – пошук рішення, яке схвалить якнайбільше учасників, прагнення уникнути «переможців» і «переможених» зміною небажаних хоч для якої меншості пунктів;
- Голосування – загальна думка формулюється підрахунком голосів учасників, має різні модифікації, наприклад: дозвіл обирати кілька варіантів, вимога мати абсолютну чи кваліфіковану (конституційну) більшість.

Існують і інші методи, наприклад, метод Делфі, дотмократія (голосування точками, dotmocracy, dot-voting), але вони не такі поширені та практичні. Також важлива організація групи – збори, (під)комітет, колегіальність тощо; окремі характеристики способу – анонімність,

випадковість, делегування, усереднювання, різні типи більшості, одностайність, консультації, обговорення тощо.

Голосування — процес ухвалення рішення групою людей (зборами, електоратом), при якому загальна думка формується шляхом підрахунку голосів членів групи. Технологія голосування широко застосовується в окремих галузях програмного забезпечення, наприклад, у соціальних мережах (термінологічно там частіше застосовують слово «опитування», адже зазвичай йдеться про не дуже важливі питання та загалом варіанти без дій та ухвал).

Існує багато різновидів виборчих систем. Один з найістотніших поділів – різниця між мажоритарними та пропорційними системами. Проте з погляду теорії прийняття рішень, група є єдиним ОПР, тому пропорційність у багатьох питаннях неможлива.

Преференційне голосування – це система голосування, за якої виборці розташовують варіанти по порядку у своїх відповідних бюлетенях. Преференційне голосування дає змогу виборцям точніше відображати їхні вподобання, ніж звичайна система відносної більшості (first-past-the-post voting), даючи змогу проголосувати ранжуванням і таким чином виявити ступінь згоди [9].

Ранжування – процес визначення рангів, відносних кількісних оцінок ступенів відмінностей, впорядкування множини альтернатив. Ранжування застосовується у випадках, коли неможлива або недоцільна безпосередня оцінка.

Попарне порівняння – це процес, за яким ОПР порівнює між собою пари об’єктів із деякого списку за деяким критерієм, щоразу вказуючи варіант, якому віддає перевагу. Метод можна зобразити таблицею, де варіанти розташовані у рядках та стовбцях, а кожна клітинка показує переважання варіанта рядка над стовбцем (рис 2.1). З цього випливає, що за наявності n альтернатив потрібно порівняти щонайбільше $m = \sum_{k=1}^{n-1} k$ пар,

	K_1	K_2	K_3	Л.Пр.
K_1	1	2	3	0,540
K_2	1/2	1	2	0.297
K_3	1/3	1/2	1	0.163

Рисунок 2.1 – Приклад таблиці попарних порівнянь

оскільки оцінки ab та ba мають бути оберненими, а оцінки типу aa виражатимуть рівність. У практичній реалізації і це число можна зменшити, якщо надати більше ваги кожному вибору та вважати, що вони впливають один на одного. Наприклад, прийнявши транзитивність за істину, хоч вона й іноді порушується в буденних ситуаціях. Якщо оцінка є числом, в цілому достатньо порівняти лише один об’єкт зі всіма, а інші відповідності виснувати з наявних даних [10].

У випадку некретиріального оцінювання альтернатив перспективним є застосування гештальт-ранжування, тобто одночасне пропонування ОПР для швидкого інтуїтивного впорядкування певного числа випадкового обраних альтернатив [5].

Для ОПР найпростішим і найнадійнішим буде одночасне порівняння якнайменшої кількості об’єктів – двох. Таке порівняння добре описано апаратом бінарних відношень та попарних порівнянь. Щоправда, такий підхід не буде враховувати ефект контексту вибору, коли включення або виключення з множини альтернатив деякого варіанту істотно впливає на вибір. На додачу такий метод не перевірятиме, чи може особа проранжувати більше альтернатив одночасно, що могло б точніше передати вподобання.

Також можна змінювати кількість рівнів порівняння. Що менше їх, то менше зусиль потрібно, проте зменшується точність. Дедалі більша кількість уточнюватиме порівняння, але проте завелика перестане бути легко зрозумілою. Крім того, будь-яка інтервальна шкала не уможливить визначення у скільки разів одна альтернатива краща за іншу – для такого треба використовувати числову оцінку, тобто пропорційну шкалу.

2.2 Удосконалення інформаційної технології багатокористувацького ранжування зображень

На основі вище наведеного огляду удосконалимо інформаційну технологію багатокористувацького ранжування для інтелектуальної системи.

Система спроектуємо таким чином, щоб спочатку оброблялися вибори окремого користувача та формувалося його особисте ранжування, а потім результати усіх користувачів об’єднувалися в підсумкове ранжування. Враховуючи усі використані методи та дії, які має виконати користувач, схему проведення багатокористувацького ранжування можна зобразити як показано на рисунку 2.2.

Рисунок 2.2 – Схема проведення багатокористувацького ранжування

Для особистого ранжування використаємо метод попарного порівняння, а саме надаватимемо користувачу по два зображення. Як вже сказано, це найпростіший і найнадійніший спосіб, який застосовує бінарні відношення та попарні порівняння. Недоліками буде ігнорування контексту вибору та можливості, що ОПР здатен на ширше ранжування, проте простота і швидкість переважають їх.

Порівняння буде вважатися однокритеріальним – згідно з метою отримати швидке впорядкування на основі швидкої реакції користувача не варто вводити додаткові оцінки, а задовольнитися єдиною.

Формування особистого ранжування буде складатися з таких кроків:

1. Структуризувати задачу у вигляді ієрархічної структури
2. Заповнити матрицю попарних порівнянь
3. Обчислити кількісний індикатор якості кожної альтернативи
4. Впорядкувати альтернативи

Розгляньмо їх детальніше.

Задача структурована як ієрархічна структура так:

Ціль: впорядкувати альтернативи;

Критерії: один загальний;

Альтернативи: зображення, які надасть користувач.

Заповнення матриці попарних порівнянь відбуватиметься відповідно до того, що вноситиме користувач. Розгляньмо приклад матриці, показаний у таблиці 2.1.

Таблиця 2.1 – Приклад матриці попарного порівняння

	a_1	a_2	a_3	Власний вектор	Нормовані значення	Порядок
a_1	1	1/2	3	1,14	0,31	2
a_2	2	1	5	2,15	0,58	1
a_3	1/3	1/5	1	0,41	0,11	3
				Σ=3,7	Σ=1	

Оцінка «1» означає рівність елементу рядка та стовпчика; «9» – найбільшу перевагу; усі числа між ними відповідно різні ступені переваги. В оберненій парі оцінка буде оберненим дробом.

Елементи власного вектора обчислюються як корінь степеня N із добутку елементів відповідного рядка матриці:

$$b = \sqrt[N]{\prod_{i=1}^{N} a_i}\,.$$

Ці значення унормовуємо (ділимо кожен окремий на суму всіх) й отримуємо вагові коефіцієнти, за якими впорядковуємо альтернативи:

$$w = \frac{b_i}{\sum_i b}\,.$$

Підрахунок голосів усіх користувачів можна здійснювати різними способами. Найдоцільнішим виглядає рейтингове (альтернативне) голосування (instant-runoff voting etc.). Від учасників вимагається заповнити ранжування альтернативами. Спочатку підраховуються всі голоси за перше місце. Якщо хтось здобув достатню порогову кількість голосів – він здобуває місце. Також в кожному раунді найнижче місце вибуває, а голоси переходять до другого у відповідному ранжуванні і так до кінця. Для суперечливих випадків розраховуватимемо тайбрейкер, для якого кожне наступне місце важить стільки, скільки і 2/3 попереднього.

2.3 Розробка структури інформаційної технології багатокористувацького ранжування зображень

Першим модулем системи, з яким взаємодіятиме користувач, є частина програми, що відповідає за вхід та реєстрацію. Цей модуль необхідний, оскільки система потребує механізми для опрацювання облікових записів.

Програма використає класичну форму входу з логіном та паролем, відповідну реєстрацію та алгоритм, який наведено на рисунку 2.3.

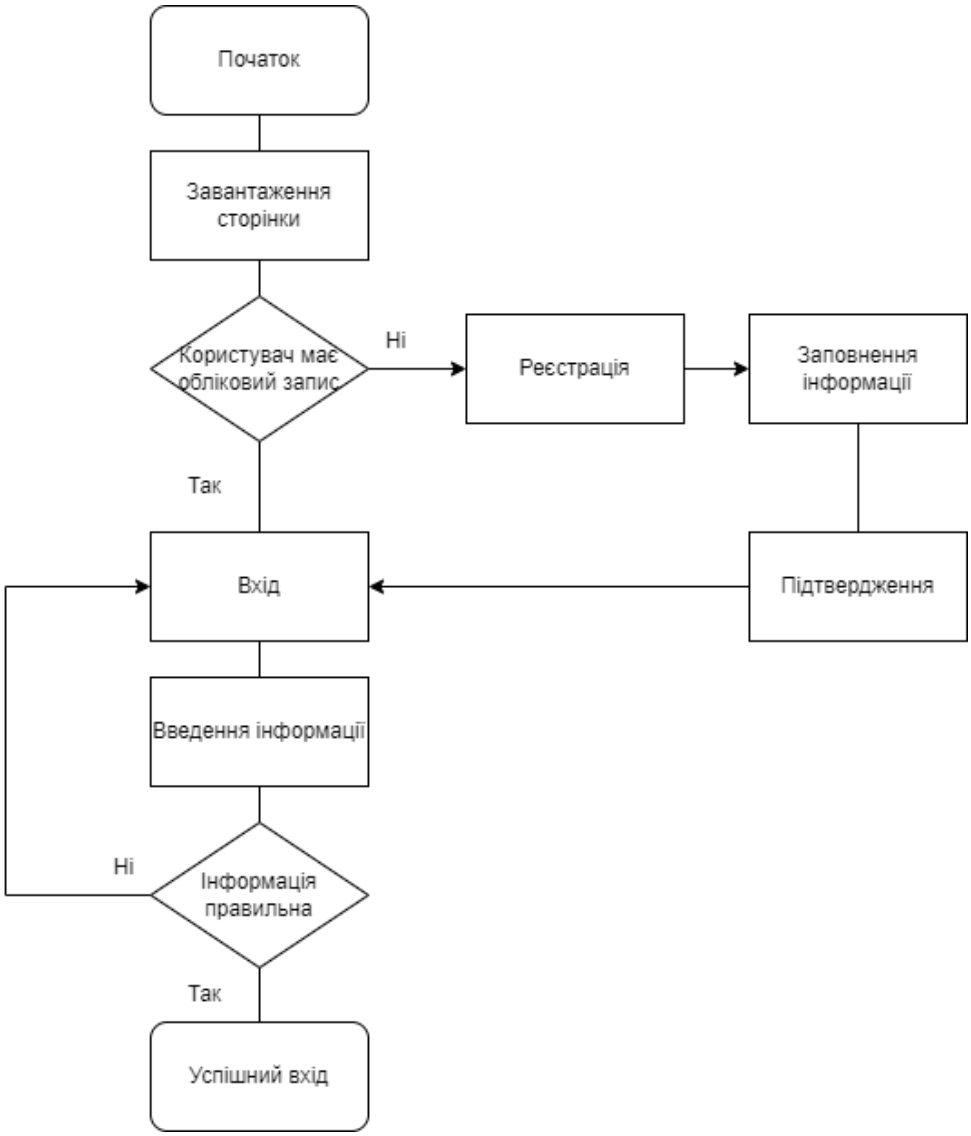


Рисунок 2.3 – Алгоритм входу

Головною частиною програми є надмодуль, який відповідає за процес особистого ранжування: вибір альтернатив для порівняння, організація взаємодії з користувачем, запис результатів вибору користувача, обчислення показників альтернативи, впорядкування ранжування. Алгоритм роботи цієї частини програми матиме такий вигляд (рис. 2.2).

Рисунок 2.4 – Схема алгоритму роботи програмного модуля процедури ранжування

Продовження рисунку 2.4

Загалом цей алгоритм можна описати двома частинами: головне меню та екран ранжування.

- Головне меню: дає змогу почати процес ранжування, змінити налаштування, подивитися історію, вийти з програми.
- Екран ранжування: пропонує користувачу висловити свої вподобання за допомогою оцінки.

Щоб показати функціонування модулів програми, наведено діаграму класів на рисунку 2.1.

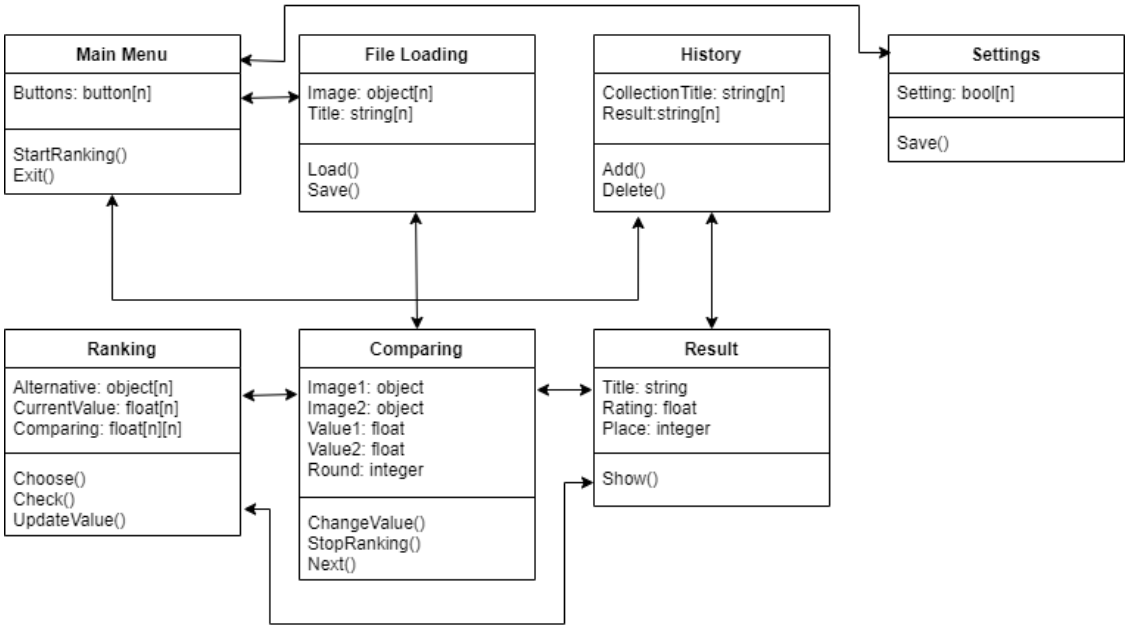


Рисунок 2.5 – Діаграма класів програмного модулю

Програмний модуль складається з таких класів:

- Клас головної сторінки – відповідає за початкове меню програми, реєстрацію, вхід, початок роботи.
- Клас налаштувань – відповідає за користувацькі налаштування.
- Клас завантаження файлів – відповідає за завантаження зображень, які є альтернативами.
- Клас історії – відповідає за збереження історії.
- Клас ранжування – обирає зображення для порівняння та сприймає відповіді користувача щодо переваги одного зображення над іншим, робить необхідні обчислення.
- Клас порівняння – відповідає за організацію екрану порівняння, де користувач вказуватиме свої вподобання.
- Клас результатів – відповідає за показ результатів.

Головним екраном програми є екран ранжування, де користувачу пропонуватимуть висловити свої вподобання. Його схематично зображено на рисунку 2.3:

1 – Область зображення – місце, де розміщено зображення для ранжування.

2 – Повзунки – повзунки для оцінювання зображень користувачем.

3 – Назва – назви зображень.

4 – Кнопка «Завершити» – кнопка, яка дає змогу перервати ранжування й перейит до поточного підсумку.

5 – Кнопка «Далі» – кнопка переходу до наступної пари.

6 – Показник процесу – показує прогрес ранжування.

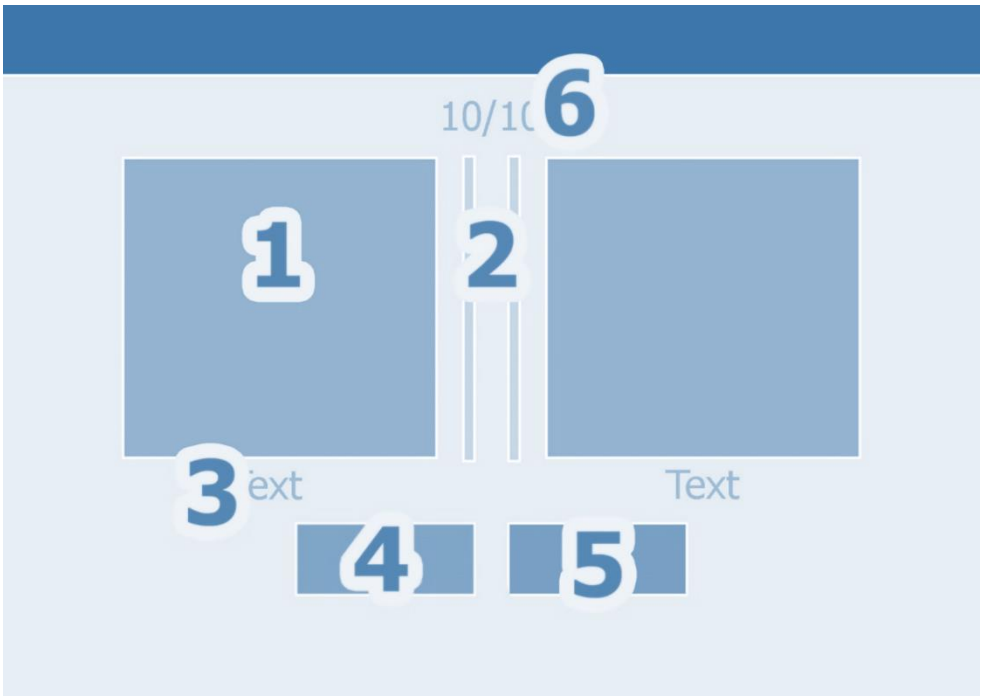


Рисунок 2.6 – Схематичне зображення екрану ранжування

2.4 Висновок

У даному розділі було проаналізовано математичні методи багатокористувацького ранжування зображень. Досліджено методи багатокористувацького ранжування зображень. Розроблено метод, який буде використано в інформаційній технології.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ БАГАТОКОРИСТУВАЦЬКОГО РАНЖУВАННЯ ЗОБРАЖЕНЬ

3.1 Вибір інструментарію для розробки системи багатокористувацького ранжування зображень

Найдоцільнішою формою програмної реалізації багатокористувацького ранжування зображень можна вважати сайт. Насамперед це найзручніше через потребу обслуговувати кілька користувачів, розділяти їхню особисту інформацію, але мати можливість спільної роботи. Подання системи як сайту дасть змогу використати звичні для веб-сайтів концепції автентифікації та авторизації. Також це зручно та звично для користувача, адже реєстрація та вхід існує на багатьох сайтах.

Вебсайт — сукупність веб-сторінок та залежного вмісту, доступних у мережі Інтернет, які об'єднані як за змістом, так і за навігацією під єдиним доменним ім'ям. Фізично сайт може розміщуватися як на одному, так і на кількох серверах. Сайтом також називають вузол мережі Інтернет, комп'ютер, за яким закріплена унікальна IP-адреса, і взагалі будь-який об'єкт в Інтернеті, за яким закріплена адреса, що ідентифікує його в мережі (FTP-site, WWW-site тощо). Набір зв'язаних між собою інформаційних онлайнових ресурсів, призначених для перегляду через комп'ютерну мережу за допомогою спеціальних програм — браузерів. Вебвузол може бути набором документів в електронному вигляді, онлайновою службою [11].

Найголовнішим інструментом створення сайту є HTML. HTML (HyperText Markup Language — мова розмітки гіпертексту) — стандартизована мова розмітки документів веб-сторінок, які відображаються у браузері. Веб-браузери отримують HTML-документ від

сервера за стандартними протоколами HTTP/HTTPS або відкривають з локального диска. Після цього код інтерпретується в інтерфейс, який відображається на екрані. Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, наприклад інтерактивні форми, можна вбудувати в сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML обрамлені тегами, написаними з використанням кутових дужок. Теги на зразок <img/> чи <input /> безпосередньо додають новий вміст на сторінку. Інші теги, такі як <p>, форматують текст і надають інформацію про нього. Різні теги можуть вміщувати інші теги як піделементи. Браузери не виводять теги HTML на екран, а лише використовують їх для інтерпретації вмісту сторінки.

HTML надає засоби для реалізації таких можливосткй:

- створення структурованого документа через позначення структурного вмісту тексту: заголовки, абзаци, списки, таблиці, цитати тощо;
- отримання інформації з інтернету через гіперпосилання;
- створення інтерактивних форм;
- додавання зображень, звуку, відео чи інших об'єктів в текст [12].

CSS (Cascading Style Sheets, Каскадні таблиці стилів) — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду, але не зачіпає їхній вміст; вся інформація написана мовами розмітки даних.

CSS використовується розробниками веб-сторінок, щоб задати кольори, шрифти, розташування елементів та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (контент, наповнення; зазвичай HTML, XML чи схожа мова розмітки) від вигляду документу.

Таке розділення покращує сприйняття та доступність контенту, забезпечує більшу гнучкість та контроль за відображенням контенту в різних умовах, робить контент структурованішим та простішим, прибирає повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою, у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.) [13].

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає змогу на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Для роботи з цими мовами потрібно обрати редактор. Atom — розроблений компанією «GitHub» вільний текстовий редактор і редактор коду, який може використовуватися як самодостатнє рішення, так і у ролі технологічного стека для побудови різних спеціалізованих рішень. Зокрема, на платформі Atom побудовані середовища розробки «Visual Studio Code» від компанії «Microsoft» і «Nuclide» від «Facebook».

Проєкт був представлений компанією GitHub у лютому 2014 року. Перший стабільний випуск 1.0 побачив світ 25 червня 2015-го. Сирцевий код проєкту поширюється під ліцензією «MIT».

Atom надає засоби кросплатформового редагування коду, включає вбудований пакетний менеджер і інтерфейс навігації файловою системою, надає засоби для одночасної спільної роботи з кодом, має інтелектуальну систему автодоповнення вводу, надає режими сумісності з Vim і Emacs, підтримує API для розробки розширень. Кілька файлів можуть бути відкриті в різних вкладках і одночасно показані з використанням вертикального або горизонтального розбиття панелей. Інтерфейс може налаштовуватися через теми оформлення, підтримуються вкладки, закладки, розумний контекстний пошук коду, схлопування блоків коду, одночасне використання декількох курсорів і областей виділення, наочна позначка змін, автодоповнення та перевірка коду для різних мов (Ruby, Python, SQL, PHP, Perl, Objective-C, C/C++, JavaScript, Java, Go тощо). Для формування статей та документації може бути використана розмітка Markdown [14].

React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Meta (раніше Facebook) і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують мінливі з часом дані, без перезавантаження сторінки. Завдання бібліотеки полягає в тому, щоб зробити сайт швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках, тобто відповідає виду в шаблоні модель-вид-контролер (MVC). Бібліотека може бути використана разом з іншими JavaScript бібліотеками або в великих фреймворках, наприклад AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux [15].

3.2 Програмна реалізація системи багатокористувацького ранжування зображень

Зараз розробка веб-застосунку з використанням тільки HTML та CSS є застарілим підходом до проектування. Навіть з додатковими інструментами, як XML-документи чи PHP, це все ще не оптимальний план, зокрема через надлишкові звертання до сервера, що знижує швидкість відповіді користувачу. Доцільніше проводити проектування, застосовуючи концепцію односторінкового застосунку (single-page application).

Односторінковий застосунок або односторінковий інтерфейс – це веб-застосунок чи веб-сайт, який вміщується на одній сторінці для того, щоб забезпечити користувачу досвід схожий на використання настільної програми. В такому застосунку весь потрібний код – HTML, JavaScript, та CSS – завантажується разом зі сторінкою або динамічно довантажується на вимогу, зазвичай у відповідь на дії користувача. Сама сторінка не перезавантаєуться і не направляє користувача до інших сторінок під час роботи. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з сервером [16].

Структурна схема, яку наведено на рисунку 3.1, показує зв’язки між модулями у веб-застосунку.

Рисунок 3.1 – Структурна схема архітектури системи багатокористувацького ранжування зображень

React дає змогу створити односторінковий інтерфейс. З самого початку React був спроектований для поступового вбудовування, його можна використовувати рівно стільки, скільки необхідно.

React-компонент легко додати до наявної HTML-сторінки. Найпростішим, але доволі незграбний спосіб є таким:

1. Додати до сторінки порожній елемент , наприклад:

```
<div id="like_button_container"></div>
```

2. Додати три теги скриптів:

```
<script
src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
```

```
  <script       src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>

<script src="like_button.js"></script>
```

Перші два теги завантажують React. Третій завантажує код компонента.

3. Створити файл з назвою like_button.js в директорії з вашою HTML-сторінкою та вставити такий скрипт:

```
'use strict';

const e = React.createElement;

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked: false };
  }

  render() {
    if (this.state.liked) {
      return 'You liked this.';
    }

    return e(
      'button',
      { onClick: () => this.setState({ liked: true }) },
      'Like'
    );
  }
}

const                    domContainer                    =
document.querySelector('#like_button_container');
```

```
const root = ReactDOM.createRoot(domContainer);
root.render(e(LikeButton));
```

У такий спосіб отримаємо дуже простий React-функціонал: натискання на просту кнопку призводить до появи напису.

Звісно, для створення повноцінного застосунку варто застосувати кращий набір інструментів. Для простого створення односторінкового інтерфейсу під час опановування роботи з React можна використати середовище Create React App.

Алгоритм роботи розробленого методу багатокористувацького ранжування зображень, як вже описано вище, можна розділити на дві частини: одна з яких відповідає за особисте ранжування, друга – за створення підсумкового ранжування. Також такий поділ грубо збігається з розділенням на частину, яка взаємодіє з користувачем – авторизація, керування обліковим записом, створення набору зображень, проведення ранжування – та частина, яка лише зберігає та обчислює. Спрощений загальний алгоритм роботи розробленого методу багатокористувацького зображення зображено на рисунку 3.2.

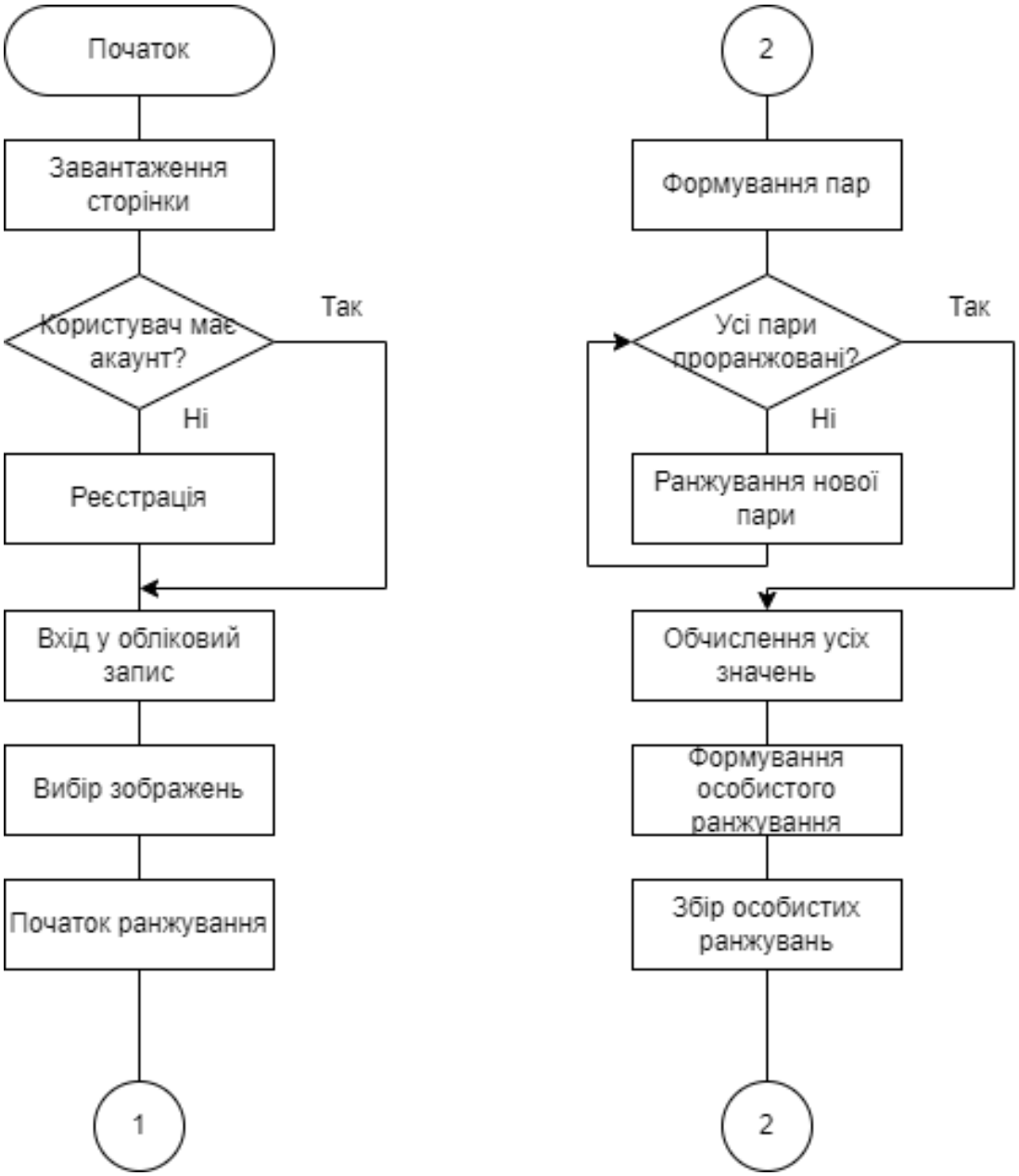


Рисунок 3.2 – Алгоритм роботи розробленого методу багатокористувацького ранжування зображень

Продовження рисунку 3.2

Важливою концепцією, з якою працює React, є розбиття коду. Так, усі програми складаються з кількох файлів, які об’єднують у процесі зшивання (бандлінг). Така дія об’єднує імпортовані файли в один, що дозволяє оперувати ними як одним цілим. Зазвичай цим займається інструментарій розробки автоматично.

Проте важливо збалансувати єдність модульністю. Крім того, що вона приносить ясність, логічність, обмеженість, універсальність, замкнутість, мінімалізм, вона також забезпечує можливість «лінивого завантажування» (lazy load) – активне завантаження лише необхідних файлів, завантаження на вимогу. Це може значно покращити продуктивність програми. Хоч це й не скорочує обсяг коду, так можна уникнути завантаження того коду, який

ймовірно ніколи не знадобитться користувачеві, а також скоротити обсяг коду, що необхідний на початку завантаження. З простіших прикладів таке можна реалізувати динамічним імпортом:

```
import("./math").then(math => {
  console.log(math.add(16, 26));
});
```

Але є і потужніші способи.

Вирішувати, де саме запроваджувати розбиття коду, часом буває непросто. Слід намагатися обирати місця для розбиття таким чином, щоб розбивати бандли рівномірно, але при цьому не створювати проблем для користувача.

Почати можна із маршрутів програми. Більшість людей звикли до того, що перехід між сторінками займає певний час. Крім того, звичайною справою є перезавантажити цілу сторінку одразу, тому користувачі навряд чи будуть взаємодіяти з іншими елементами на сторінці в цей час.

Нижче наведено приклад налаштування розбиття коду на основі маршрутів, використовуючи бібліотеку React Router за допомогою React.lazy.

```
import React, { Suspense, lazy } from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

const Home = lazy(() => import('./routes/Home'));
const About = lazy(() => import('./routes/About'));

const App = () => (
  <Router>
    <Suspense fallback={<div>Завантаження...</div>}>
      <Switch>
        <Route exact path="/" component={Home}/>
```

```
            <Route path="/about" component={About}/>
          </Switch>
        </Suspense>
      </Router>
    );
```

Результати обох типів ранжування будуть подаватися у вигляді таблиці. Схематично це показано на рисунку 3.3.

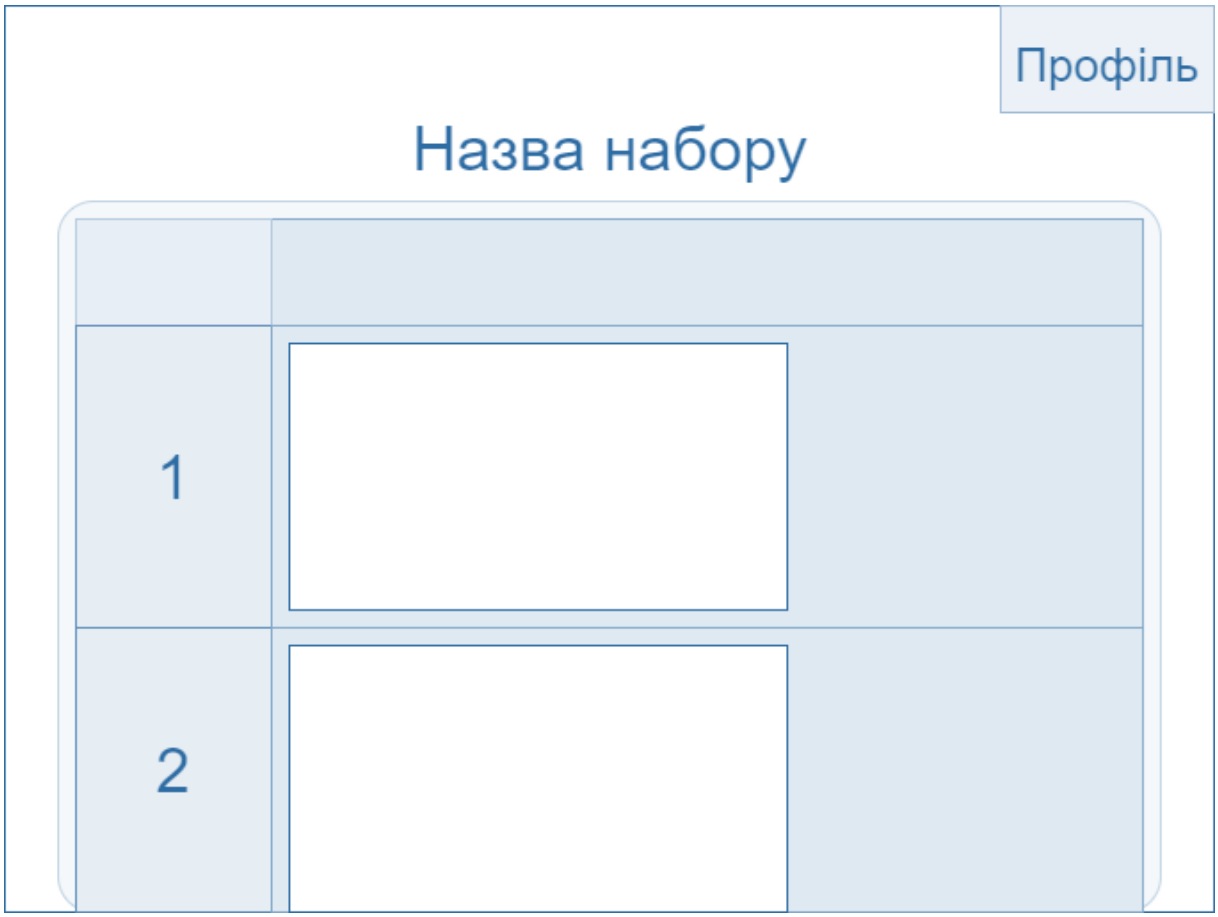


Рисунок 3.3 – Схематичне зображення результатів ранжування

Складені таблиці такого типу легко зробити за допомогою базового HTML, який уже має універсальні, хоч і не найгнучкіші, інструменти для такої розмітки, зокрема не тільки просто як таблиця, а й використовуючи теги та – найпростіші контейнери; проте для роботи в React він матиме десь такий вигляд, щоб врахувати мінливість:

```
    const Table = ({data}) => {
        return (
            <div className="table_base">
                <div className="col s8 m2">
                    <div className="table">
                        <div className="table-image">
                            <img src={data.tableImg} />
                            <span className="table-
title">{data.tableTitle}</span>
                        </div>
                        <div className="table-content">
                            <div>{data.tableText}</div>
                        </div>
                        <div className="table-action">
                            <a href="#">X</a>
                        </div>
                    </div>
                </div>
            </div>
```

3.3 Тестування програмного забезпечення багатокористувацького ранжування зображень

Для початку тестування треба запустити застосунок в локальній мережі.

Через деякий час проект повністю завантажиться, та автоматично відкриється посилання у локальній мережі (рис. 3.4).

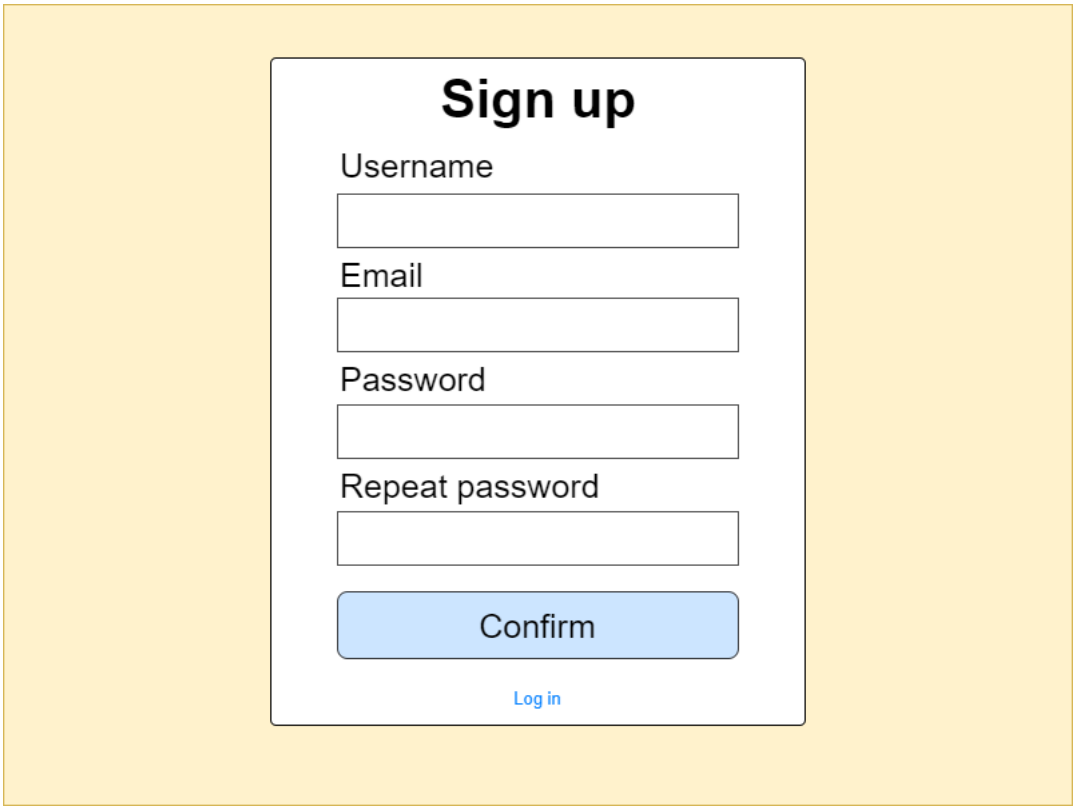


Рисунок 3.4 – Сторінка авторизації

Далі потрібно зареєструвати користувача на сайті, заповнивши форму реєстрації (рис. 3.5).

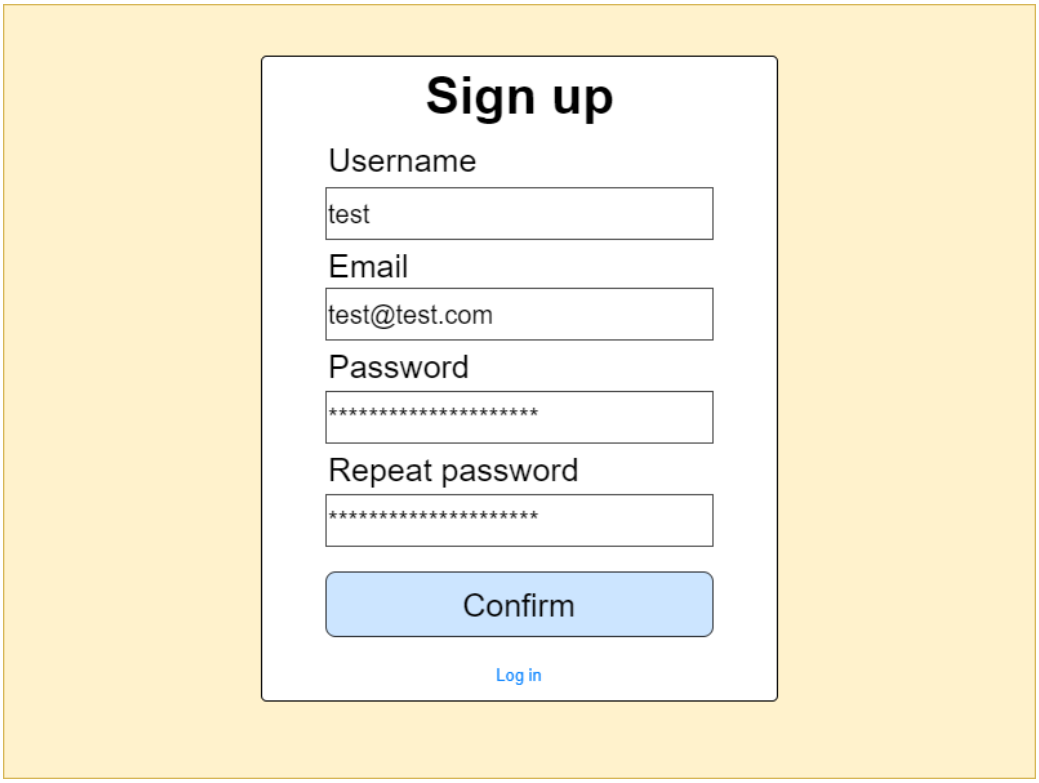


Рисунок 3.5 – Реєстрація користувача

Після реєстрації та входу відбудеться перенаправлення на головну сторінку. У нового користувача вона буде порожньою з підказкою для створення ранжування, але для користувача з наявними ранжуваннями головна сторінка буде відображати це.

Створити набір зображень можна, натиснувши кнопку «Створити» та обравши потрібні файли в провіднику (рисунок 3.6).

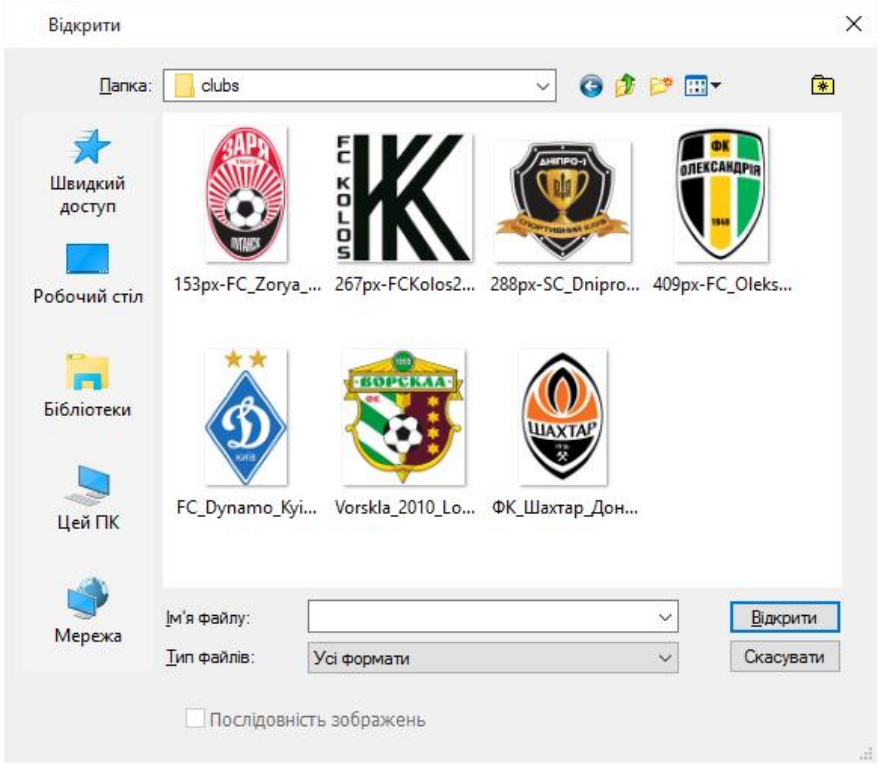


Рисунок 3.6 – Провідник

Процес ранжування зображено на рисунку 3.7. Зображення виводяться парами, користувач має вказати свої вподобання. Крім того, видно прогрес набору.

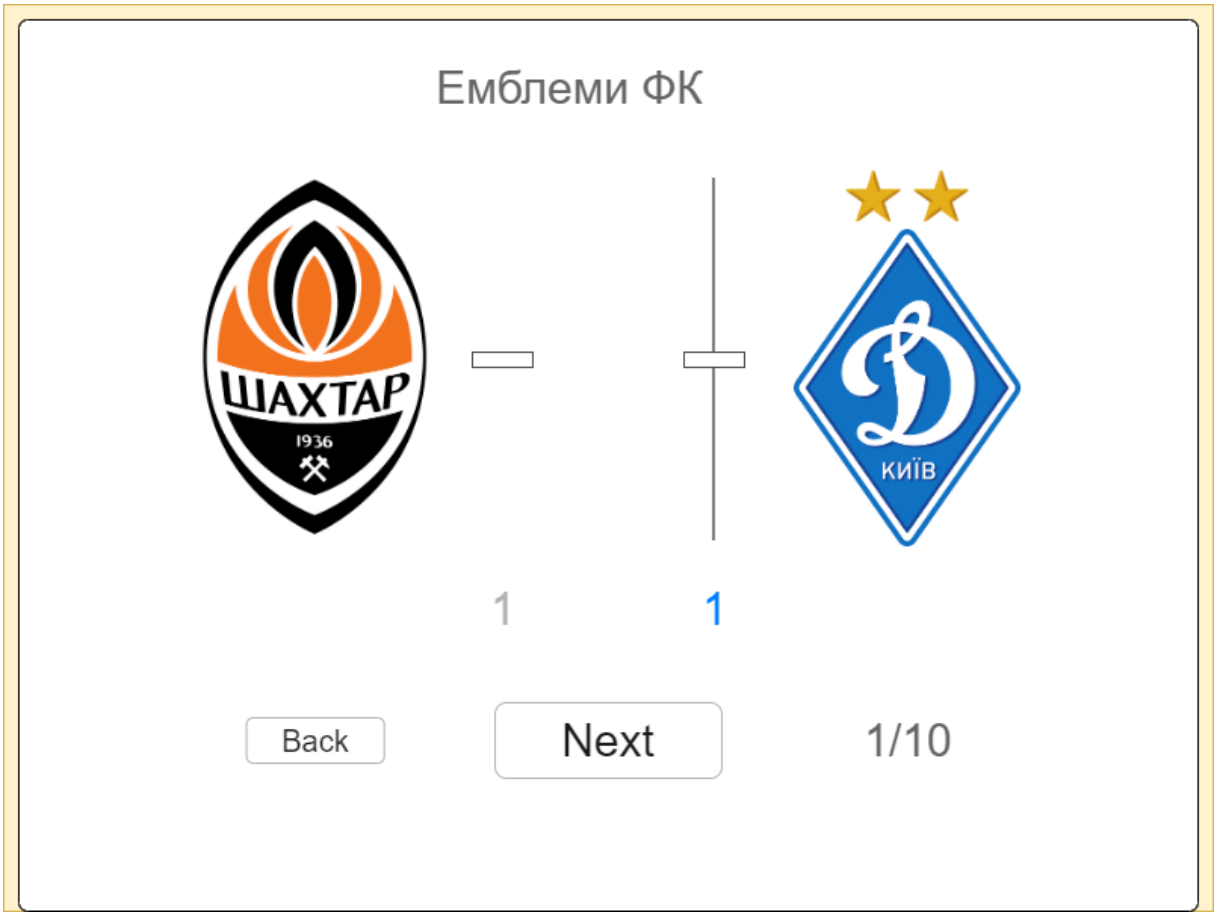


Рисунок 3.7 – Процес ранжування

Особисті та спільні ранжування можна переглянути на сторінці з результати, приклад наведено на рисунку 3.8.

Рисунок 3.8 – Результати ранжування

У результаті розробки досягнено поставленої мети за рахунок реалізації функцій, що відсутні в програмах-аналогах, а саме:

- Реєстрація та авторизація
- Можливість роботи зі зображеннями
- Підтримка багатокористувацької роботи
- Попарне порівняння
- Задання числових оцінок
- Використання преференційного голосування

Ці функції описано в таблиці 3.1.

Таблиця 3.1 – Таблиця порівняння наявних функцій в програмах

	Реєстрація та авторизація	Можливість роботи зі зображеннями	Підтримка багатокористувацької роботи	Попарне порівняння	Задання числових оцінок	Використання преференційного голосування
Розроблюваний продукт	+	+	+	+	+	+
Rcv123	-	-	+	-	-	+
Ranker	-	+	-	+	-	-

З таблиці можна побачити, що розроблений програмний засіб має покращений функціонал, щонайменше на 2 можливості у порівнянні з програмами-аналогами, що забезпечить вищу продуктивність користувача.

3.4 Висновок

У цьому розділі обрано мову програмування та середовище розробки. Програмно реалізоване багатокористувацьке ранжування зображень. Протестовано інформаційну технологію.

Описана загальна структурна схема функціонування системи. Наведений та описаний алгоритм роботи розробленого методу багатокористувацького ранжування зображень.

Спроектований схематичний вигляд інтерфейсу головної сторінки системи.

Проведено тестування розробленої системи багатокористувацьке ранжування зображень.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою цього розділу магістерської кваліфікаційної роботи є проведення технологічного аудиту нового програмного продукту для багатокористувацького ранжування зображень.

Аналогом може бути програмна система Rcv123. Її основним недоліком є відсутність роботи зі зображеннями, вартість програмного комплексу складає кілька тисяч доларів.

Для проведення комерційного та технологічного аудиту потрібно не менше трьох незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендовано здійснювати п’ятибальною системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних
Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому	Кілька аналогів на великому	Один аналог на великому ринку	Продукт не має аналогів на великому

Продовження таблиці 4.1

3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче 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					
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
Практик на 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

Продовження таблиці 4.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування	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 використовуються 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	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
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 вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	4	4
Наявність аналогів на ринку	4	4	4
Цінова політика	3	4	4
Технічні та споживчі властивості виробу	3	3	4
Експлуатаційні витрати	3	4	4
Ринок збуту	3	3	3
Конкурентоспроможність	3	3	4
Фахівці з технічної і комерційної реалізації	3	3	3
Фінансування	3	4	3
Матеріально-технічна база	4	4	3
Термін реалізації ідеї	4	4	3
Супровідна документація	4	3	3
Сума	41	43	42
Середньоарифметична сума балів	(41+43+42) / 3 = 42		

За даними таблиці 4.2 можна зробити висновок щодо потенціалу розробки, спираючись на рекомендації, наведені в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

З таблиці випливає, що рівень комерційного потенціалу розроблюваного програмного продукту є високим. Це досягається завдяки відмінностям від аналогів, що дало змогу підвищити швидкодію процесу ранжування.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_p} \cdot t, \tag{4.1}$$

де M – місячний посадовий оклад конкретного розробника (дослідника), грн.; T_p – число робочих днів в місяці, 24 днів; t – число днів роботи розробника (дослідника).

Результати розрахунків наведемо в таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	19200	800	30	24000,0
Інженер	15000	625	30	18750,00
Всього				42750,0

У розробці цього програмного продукту розробник виступає одночасно і як основний робітник, і як тестувальник розроблюваного програмного продукту.

Обрахуємо додаткову заробітну плату розробників, які брали участь в розробці обладнання.

Додаткову заробітну плату прийнято розраховувати як 13 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot \frac{13\%}{100\%}, \quad (4.2)$$

$$Зд = (42750{,}00 \cdot \frac{13\%}{100\%}) = 5557{,}50 \text{ (грн.)}.$$

Нарахування на заробітну плату розробників. Згідно з чинним законодавством нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати:

$$H_з = (З_о + З_д) \cdot \frac{22\%}{100\%}, \quad (4.3)$$

$$H_з = (42750{,}00 + 5557{,}50) \cdot \frac{22\%}{100\%} = 10627{,}65 \text{ (грн.)}.$$

Оскільки для розробки не потрібно витрачати матеріали та комплектувальні деталі, то витрати на них дорівнюють нулю.

Так саме для розробки програмного забезпечення не потрібні наукові (експериментальні) роботи, тому витрати на спецустаткування для них рівні нулю.

Балансову вартість програмного забезпечення для статті «Програмне забезпечення для наукових (експериментальних) робіт» розраховують за формулою

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} С_{прг.і} К_i$$

де *Ціпрг* – ціна придбання одиниці програмного засобу цього виду, грн; *Спрг і* . – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.; *Кі* – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (*Кі = 1,10…1,12);* *k* – кількість найменувань програмних засобів.

Таблиця 4.5 – Програмне забезпечення для наукових (експериментальних) робіт

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Visual Studio (Professional)	2	1665	3330
Всього			3330

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ (грн.)}, \quad (4.4)$$

де $Ц$ – балансова вартість обладнання, грн.; $Т$ – термін корисного використання обладнання згідно податкового законодавства, років; $t_{вик}$ – термін використання під час розробки, місяців.

Розрахуємо амортизаційні витрати на комп'ютер, балансова вартість якого становить 15000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 2 місяці:

$$\text{А}_{\text{обл}} = \frac{15000}{2} \times \frac{2}{12} = 1250 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$\text{А}_{\text{н.р.}} = \text{Ц}_{\text{Н.р.}} * \text{Н}_{\text{а}} * \frac{t_{вик}}{12}. \quad (4.5)$$

Норму амортизації H_a приймемо за 10 %.

Таблиця 4.5 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер	15000	2	2	1250
Приміщення	400000	20	2	3333,33

Продовження таблиці 4.5.

Ліцензійна ОС, та спеціалізовані ліцензійні нематеріальні ресурси (Microsoft Office Professional 2021)	13500	-	2	154,69
Всього				4738,02

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. Крім того, тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній) будуть різними. Також розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \times t_i \times Ц_е \times К_{впі}}{\eta_i}, \qquad (4.6)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год; $Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії); $К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі}$ <1; η_i – коефіцієнт корисної дії обладнання, η_i<1.

$$Ц_е = \left(Ц_{опт} + Ц_{розп} + Ц_{пост}\right)\left(1 + \frac{ПДВ}{100\%}\right), \qquad (4.7)$$

де $Ц_{опт}$ – середня оптова ціна електроенергії, яка визначається оператором ринку (без ПДВ), грн за 1 кВт·год; $Ц_{розп}$ - вартість розподілу електроенергії окремою енергорозподільчою компанією (без *ПДВ*), грн за 1 кВт·год; $Ц_{пост}$ – вартість постачання електроенергії від енергорозподільчої компанії до конкретного споживача (без *ПДВ*), грн за 1 кВт·год; *ПДВ* - величина податку на додану вартість, %, у 2022 році *ПДВ*=20%.

$$Ц_е = (3{,}350 + 1{,}257 + 0{,}346) * (1 + 0{,}2) = 5{,}05 \text{ грн.}$$

Розрахуємо, для прикладу, витрати на електроенергію для комп'ютера потужність якого становить 0,09 кВт, тривалість роботи за 30 повних робочих дні – 240 годин, а коефіцієнт, що враховує використання потужності – 0,8 кВт/год

$$B_{ек} = W_{yi} \times t_i \times Ц_е \times К_{впі} = 0{,}09 * 240 * 5{,}05 * 0{,}8 = 87{,}26 \text{ грн.}$$

Аналогічно визначаємо витрати на електроенергію на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.6.

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер	0,09	30*8=240,00	87,26
Сервер	0,13	30*24=720,00	378,14
Приміщення	0,102	30*8=240,00	98,90
Всього			564,3

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = 42750{,}0 * 0{,}2 = 8550$$

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» дорівнюватимуть нулю, оскільки уся розробка виконується штатом працівників.

До статті «Інші витрати» належать витрати, які не знайшли відображення у вже згаданих пунктах витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = 42750{,}00 * \frac{110\%}{100\%} = 47025 \text{ (грн.)}.$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-

технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 42750{,}00 * \frac{120\%}{100\%} = 51300 \text{ (грн.)}.$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 42750{,}00 + 5557{,}50 + 10627{,}65 + 3330 + 4738{,}02 + 564{,}3 + 8550 + 47025 + 51300 = 174442{,}47 \text{ грн.}$$

Розрахунок загальних витрат на науково-технічну роботу та оформлення її результатів.

Загальні витрати на завершення науково-технічної роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

А саме: якщо науково-технічна розробка знаходиться на стадії науково-дослідних робіт, то η =0,1; технічного проектування, то η =0,2;

розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо $\eta = 0,7$, оскільки розробка на час розрахунків перебуває на стадії промислового зразка:

$$ЗВ = \frac{174442,47}{0,7} = 249203,52 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Підсумковим позитивним результатом, що його може отримати потенційний інвестор від впровадження результатів певної науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дасть змогу покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину наявного та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

– абсолютного економічного ефекту (чистого дисконтованого доходу);

– внутрішньої економічної дохідності (внутрішньої норми дохідності);

– терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного продукту для використання масовим споживачем

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 * N + Ц_0 * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\vartheta}{100}\right), \qquad (4.10)$$

де $\pm \Delta Ц_0$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу; N – кількість споживачів які використовували аналогічний продукт у

році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$; $Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки; ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик; λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333; p – коефіцієнт, який враховує рентабельність продукту; ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 15000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти його ціну на *1000* грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 100 шт., протягом другого року – на 150 шт., протягом третього року на 200 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0 * 1000 + (15000 + 1000) * 100) * 0{,}8333 * 0{,}5) * (1 - 0{,}18) = 512499{,}980 \text{ грн.}$$

$$\Delta П_2 = (0 * 1000 + (15000 + 1000) * (100 + 150) * 0{,}8333 * 0{,}5) * (1 - 0{,}18) = 1366666{,}612 \text{ грн.}$$

$$\Delta П_3 = (0 * 1000 + (15000 + 1000) * (100 + 150 + 200) * 0{,}8333 * 0{,}5) * (1 - 0{,}18) = 2459999{,}902 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 4339166,49 грн.

Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$\text{ПП} = \sum_1^{\text{T}} \frac{\Delta \text{П}_\text{i}}{(1+\tau)^t}, \quad (5.11)$$

де $\Delta П_i$ –збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн; $T -$ період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки; $\tau -$ ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,2; t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

$$\text{ПП} = \left(\frac{512499{,}980}{(1+0{,}1)^1}\right) + \left(\frac{1366666{,}612}{(1+0{,}1)^2}\right) + \left(\frac{2459999{,}902}{(1+0{,}1)^3}\right) = 465909{,}07 + 1129476{,}539 + 1848234{,}336 = 3443619{,}947 \text{ грн.}$$

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} * \text{ЗВ}, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2…5, але може бути і більшим; *ЗВ*

– загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 249203{,}52 = 498407{,}04 \text{ грн.}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід $(NPV, Net\ Present\ Value)$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{абс} = \text{ПП} - PV, \quad (4.13)$$

$$E_{абс} = 3443619{,}947 - 498407{,}04 = 2945212{,}91 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\text{Е}_в$. Для цього використаємо формулу:

$$\text{Е}_в = \sqrt[Т_ж]{1 + \frac{\text{Е}_{абс}}{PV}} - 1, \quad (4.14)$$

де $T_{ж}$ –життєвий цикл наукової розробки, роки.

$$\text{Ев} = \sqrt[3]{1 + \frac{2945212,91}{498407,04}} - 1 = 0.905$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau_{\text{мін}} = d + f, \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,9…0,12); f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\text{мін}} = 0,5 + 0,3 = 0,8$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\text{Т}_{\text{ок}} = \frac{1}{\text{Е}_{\text{в}}}, \qquad (4.16)$$

$$T_{\text{ок}} = \frac{1}{0.95} = 1,10 \text{ р.}$$

Оскільки $T_{ок} < 3$, а саме термін окупності рівний 1,10 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновок

Економічна частина роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає – 249203,52 гривень. Було

прогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є високо конкурентоспроможним. Період окупності складе близько 1,10 роки.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи проведено аналіз сучасного стану розвитку систем багатокористувацького ранжування зображень на основі аналізу предметної області багатокористувацького ранжування зображень, аналізу відомих технічних рішень та порівняльного аналізу характеристик програмних систем багатокористувацького ранжування зображень.

Здійснено проектування моделі інформаційної технології багатокористувацького ранжування зображень, розроблено математичну модель багатокористувацького ранжування зображень, проведено проектування інтелектуальної моделі системи багатокористувацького ранжування зображень;

Програмно реалізовано систему багатокористувацького ранжування зображень. Обґрунтовано вибір інструментарію при розробці системи багатокористувацького ранжування зображень, проведено тестування системи. У результаті тестування визначенні характеристики розробленої програми, порівняно з аналогом.

У результаті розробки досягнено поставленої мети за рахунок реалізації функцій, що відсутні в програмах-аналогах. Розроблений програмний засіб має покращений функціонал, щонайменше на 2 можливості у порівнянні з програмами-аналогами, що забезпечить вищу продуктивність користувача.

Розраховано витрати на розробку нового програмного продукту, сума складає 229911,94 гривень. Спрогнозовано оцінку витрат по кожному пункту витрат. Розраховано можливий чистий прибуток, розраховано період окупності витрат для інвестора та економічний ефект від використання розробки. Як результат аналізу розрахунків можна зробити висновок, що програмний продукт дешевший за аналог і є конкурентоспроможним. Період окупності складе близько 1,04 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. В. Колодний, В. О. Дикун Інформаційна технологія багатокористувацького ранжування зображень / у матеріалах конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 2022.

2. Колодний В. В., Основи теорії прийняття рішень. Навчальний посібник. − Вінниця: ВДТУ, 2003. — 70 с.

3. Індивідуальне і групове прийняття рішень [Електронний ресурс]. Режим доступу – http://reci.pp.ua/1413-individualnoe-gruppovoe-prinyatie.html

4. Методи обробки експертної інформації [Електронний ресурс]. Режим доступу – https://uk.wikipedia.org/wiki/Методи_обробки_експертної_інформації

5. Колодний В. В. Застосування гештальт-порівнянь для виявлення переваг ОПР / В. В. Колодний, В. В. Зубко // «ІНТЕРНЕТ-ОСВІТА-НАУКА-2014»: Збірник матеріалів конференції. − Вінниця: ВНТУ, 2014. — с. 13-14.

6. Group decision-making [Електронний ресурс]. Режим доступу – https://en.wikipedia.org/wiki/Group_decision-making

7. Free Ranked-Choice Voting for Everyone [Електронний ресурс]. Режим доступу – https://www.rcv123.org

8. Ranker | SourceForge [Електронний ресурс] – Режим доступу: https://sourceforge.net/projects/ranker/

9. Ranked voting [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Ranked_voting

10. Система оцінки при багатокритеріальному виборі: пропозиція найбільш оптимального методу [Електронний ресурс] – Режим доступу: https://doippo.dp.ua/scholarship-in-ukraine/systema-oczinky-pry-bagatokryterialnomu-vybori-propozycziya-najbilsh-optymalnogo-metodu/

11. Вебсайт [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Вебсайт

12. HTML [Електронний ресурс] – Режим доступу: https://uk. wikipedia.org/wiki/HTML

13. CSS [Електронний ресурс] – Режим доступу: https://uk. wikipedia.org/wiki/CSS

14. Atom [Електронний ресурс] – Режим доступу: https://atom.io

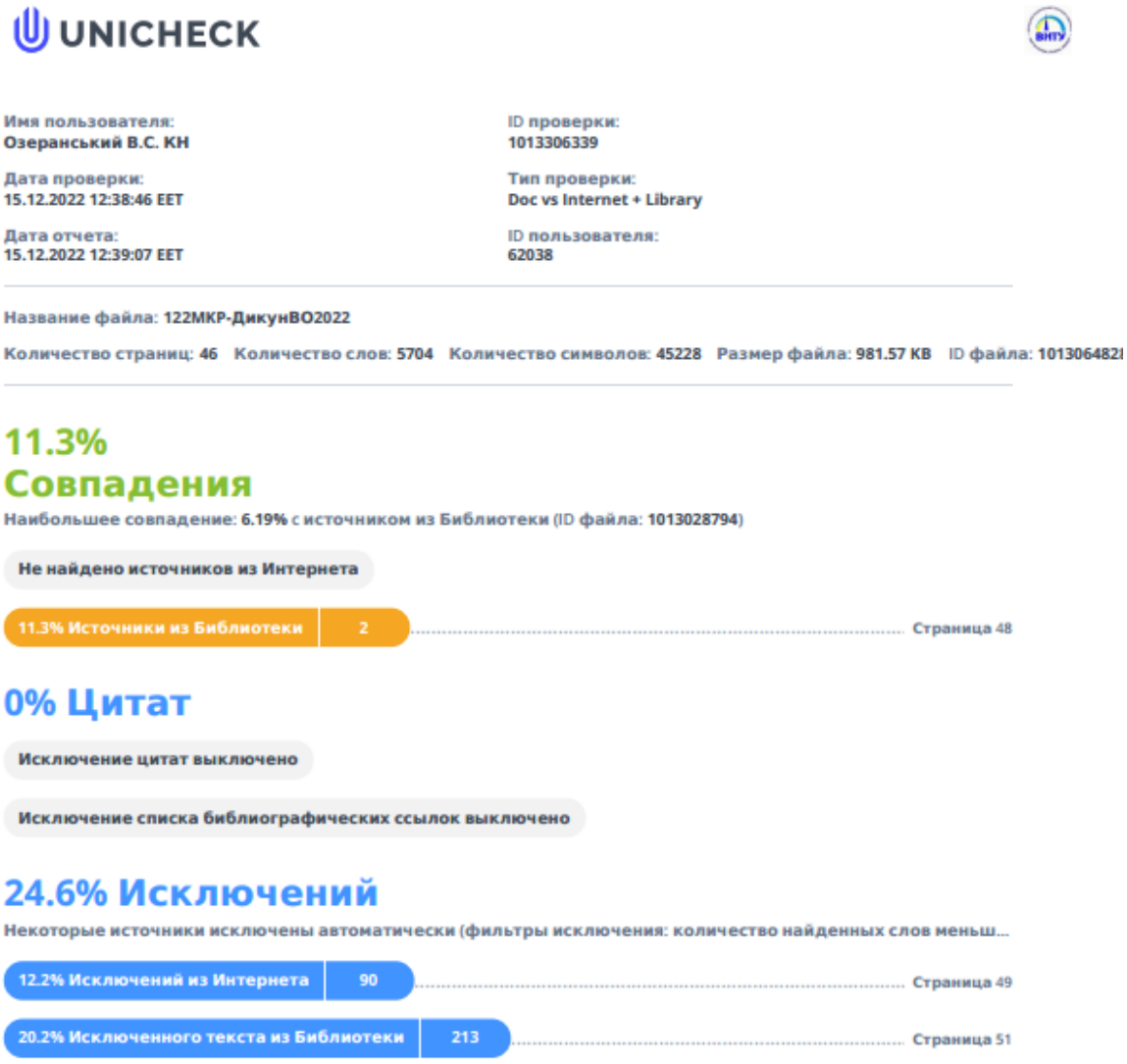
15. React [Електронний ресурс] – Режим доступу: https://uk.reactjs.org

16. Односторінковий застосунок [Електронний ресурс] – Режим доступу: https://uk. wikipedia.org/wiki/Односторінковий_застосунок

Додаток А

(обов’язковий).

Результат перевірки на антиплагіат у системі «UNICHECK»



UNICHECK

Имя пользователя:
Озеранський В.С. КН

ID проверки:
1013306339

Дата проверки:
15.12.2022 12:38:46 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
15.12.2022 12:39:07 EET

ID пользователя:
62038

Название файла: 122МКР-ДикунВО2022

Количество страниц: 46 Количество слов: 5704 Количество символов: 45228 Размер файла: 981.57 KB ID файла: 1013064828

11.3%
Совпадения

Наибольшее совпадение: 6.19% с источником из Библиотеки (ID файла: 1013028794)

Не найдено источников из Интернета

11.3% Источники из Библиотеки 2 ..... Страница 48

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

24.6% Исключений

Некоторые источники исключены автоматически (фильтры исключения: количество найденных слов меньш...

12.2% Исключений из Интернета 90 ..... Страница 49

20.2% Исключенного текста из Библиотеки 213 ..... Страница 51

Додаток Б.

(обов’язковий)

Інструкція користувача

Для початку роботи треба запустити застосунок в локальній мережі.

Через деякий час проект повністю завантажиться, та автоматично відкриється посилання у локальній мережі (рис. Б.1).

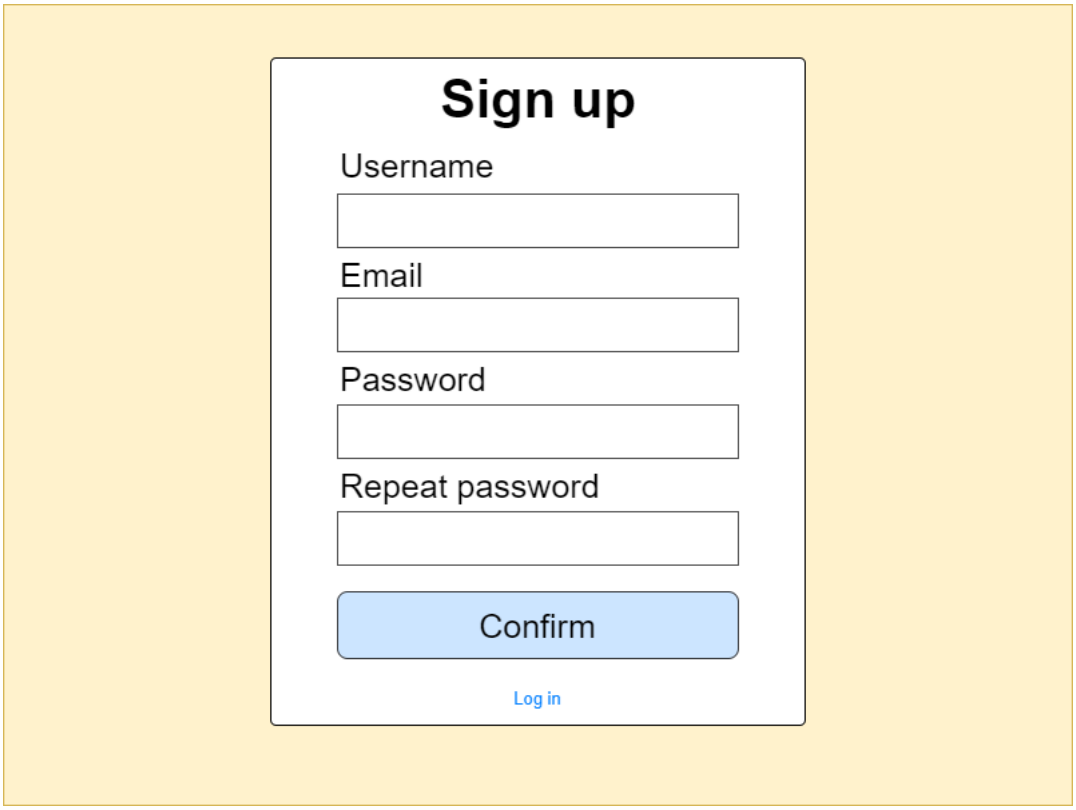


Рисунок Б.1 – Сторінка авторизації

Далі потрібно зареєструвати користувача на сайті, заповнивши форму реєстрації (рис. Б.2).

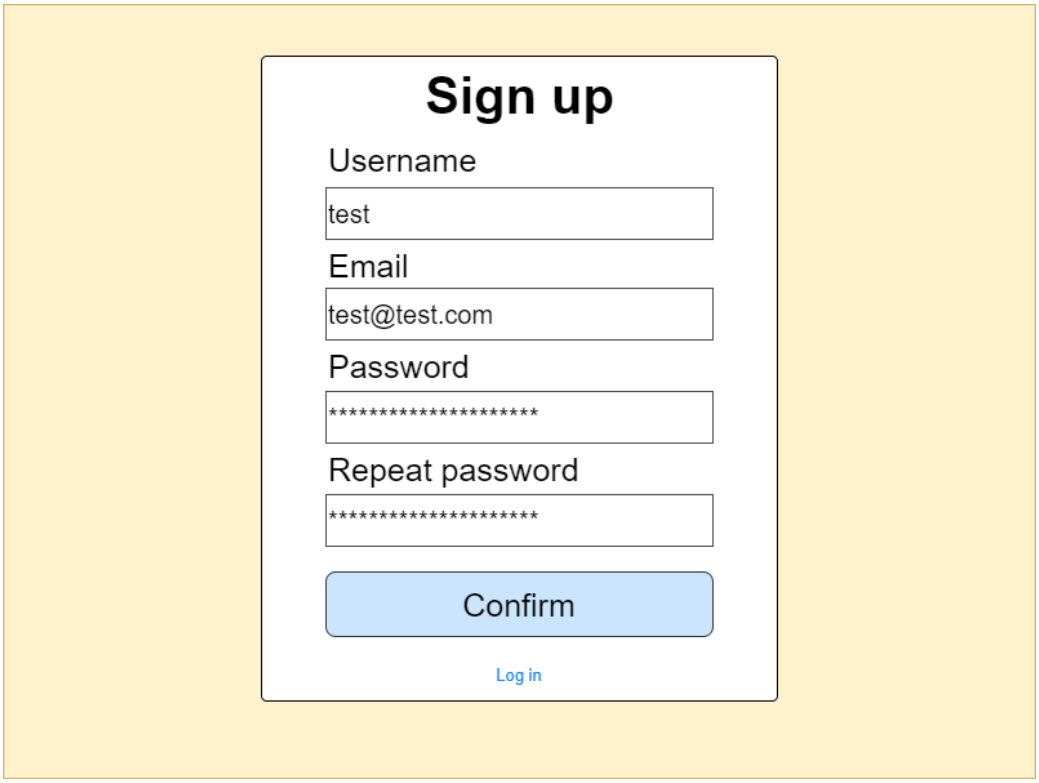


Рисунок Б.2 – Реєстрація користувача

Після реєстрації та входу відбудеться перенаправлення на головну сторінку. У нового користувача вона буде порожньою з підказкою для створення ранжування, але для користувача з наявними ранжуваннями головна сторінка буде відображати це.

Створити набір зображень можна, натиснувши кнопку «Створити» та обравши потрібні файли в провіднику (рисунок Б.3).

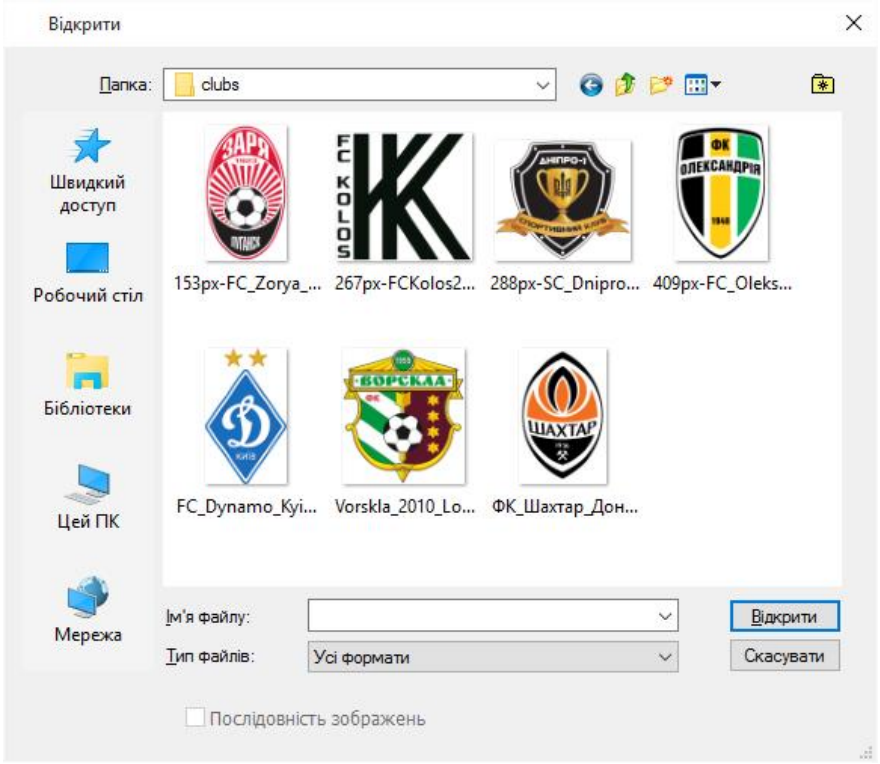


Рисунок Б.3 – Провідник

Процес ранжування зображено на рисунку Б.4. Зображення виводяться парами, користувач має вказати свої вподобання. Крім того, видно прогрес набору.

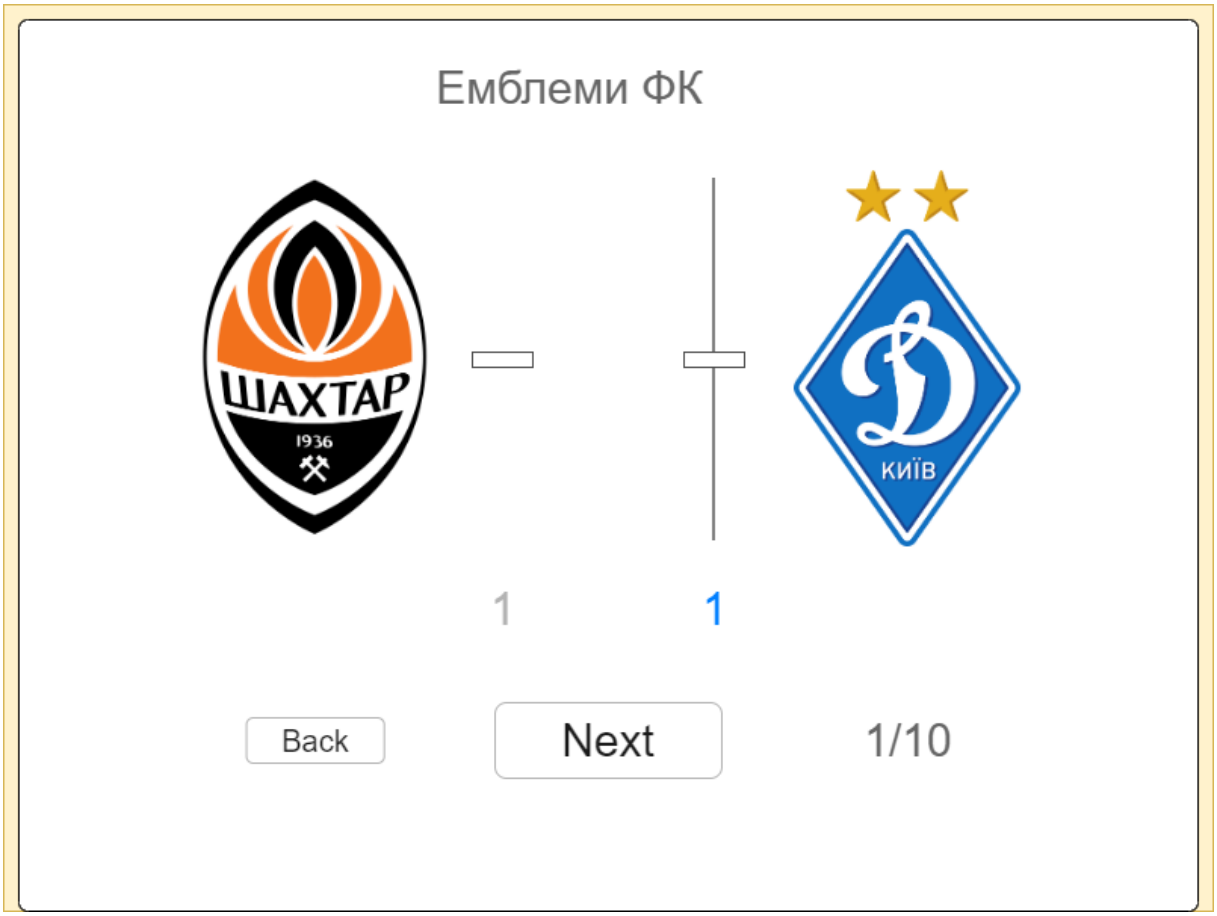


Рисунок Б.4 – Процес ранжування

Особисті та спільні ранжування можна переглянути на сторінці з результати, приклад наведено на рисунку Б.5.

Рисунок Б.5 – Результати ранжування

Додаток В.

(обов'язковий)

Лістинг програми

```
export const Authorization_Path = '/';

export const Authorized_Start_Path = '/main';

export type AppRouteProps = {
    accessType?: AppRouteAccessType;
    roles?: UserRole[];
} & RouteProps;

export const App_Routes: AppRouteProps[] = [
    {
        path: '/',
        component: AuthView,
        accessType: AppRouteAccessType.Anon,
        exact: true
    },
    {
        path: '/ranking',
        component: RankingPage,
        accessType: AppRouteAccessType.Authorized,
        exact: true
    },
    {
        path: '/main',
        component: MainView,
        accessType: AppRouteAccessType.Authorized,
    },
    {
```

```
        path: '/users',
        component: UsersView,
        accessType: AppRouteAccessType.Authorized,
        roles: [UserRole.Admin, UserRole.Owner]
    },
    {
        path: '/collections',
        component: CollectionsView,
        accessType: AppRouteAccessType.Authorized
    },
    {
        path: '/settings',
        component: SettingsView,
        accessType: AppRouteAccessType.Authorized
    }
];


const useStyles = makeStyles(theme => ({
    root: {
        display: 'flex'
    },
    content: {
        flex: 1,
        padding: theme.spacing(4)
    },
    toolbar: theme.mixins.toolbar
}));
```

```
export const DefaultLayout: React.FC = ({children}) => {
    const {state} = useUserState();
    const classes = useStyles();

    if (!state.currentUser) {
        return <> {children} </>;
    }

    return <div className={classes.root}>
        <Navbar />
        <AppDrawer />

        <div className={classes.content}>
            <div className={classes.toolbar} />

            {children}
        </div>
    </div>
}

export const RankingChart: React.FC<Props> = (props) => {
    const { data } = props;
    const theme = useTheme();

    return (
        <>
            <ResponsiveContainer width="100%" height={300}>
                <BarChart
                    data={data}
                    margin={{
                        top: 5,
                        right: 30,
```

```
                            left: 20,
                            bottom: 5,
                        }}
                        reverseStackOrder={false}
                    >
                        <CartesianGrid strokeDasharray="3 3" />
                        <XAxis dataKey="name" />
                        <YAxis />
                        <Tooltip />
                        <Legend />
                    </BarChart>
                </ResponsiveContainer>
            </>
        );
    };

    export const Navbar = () => {
        const { state } = useUserState();
        const { state: usersState, dispatch } =
useUsersState();

        const classes = useStyles();

        return <AppBar position={'fixed'}
className={classes.root}>
            <Toolbar>
                <div className={classes.selectWrapper}>
                    /> }
                </div>
                { state.currentUser && <NavbarProfile
                    currentUser={state.currentUser}
                    currentRole={state.currentRole}
```

```
                /> }
            </Toolbar>
        </AppBar>
    }

    export const RankingView = (props: Props) => {
        const {
            title = '',
            value = '',
            variant = '',
            description = '',
            icon
        } = props;

        const classes = useStyles({
            variant: variant
        });

        return <Paper>
            <Box p={2}>
                <Grid container>
                    <Grid item xs={9}>
                        <Box m={1} />
                        <Typography variant={'h4'}
classNameclasses.title}>
                            { title }
                        </Typography>
                        <Box m={1} />
                        <Typography variant={'h2'}>
                            { value }
                        </Typography>
                    </Grid>
```

```
                    <Grid item xs={3}>
                        <Box display={'flex'}
justifyContent={'center'} alignItems={'center'}
className={classes.icon}>
                            {icon && icon()}
                        </Box>
                    </Grid>
                </Grid>
                <Box m={3} />
                <Typography variant={'caption'}
className={classes.description}>
                    { description }
                </Typography>
            </Box>
        </Paper>
    }

    export const AuthLoginForm = () => {
        const { dispatch } = useUserState();

        const form = useForm({
            resolver: yupResolver(schema)
        });

        const mutation = useMutation('loginUser', (dto:
LoginUserRequestDto) => loginUserRequest(dto));

        const submitHandler = async (dto: LoginUserRequestDto)
=> {
            try {
                const response = await
mutation.mutateAsync(dto);

                if (response.status !== 200) {
```

```
                    for (let key in response.errors) {
                        form.setError(response.errors[key].path
as FieldPath<LoginUserRequestDto>, {
                            message:
response.errors[key].messages[0],
                        });
                    }
                } else {
                    Cookies.set('token', response.token);
                    dispatch(new
SetUserAction(mapUserToState(response.user as UserDto)));
                }
            } catch (error) {}
        }

        return <>
            <Typography align={'center'} variant={'h2'}>
                Login to Account
            </Typography>

            <Box m={1} />

            <Typography align={'center'}>
                Please enter your email and password to
continue
            </Typography>

            <Box m={4} />

            <FormProvider {...form}>
                <form autoComplete="off"
onSubmit={form.handleSubmit(submitHandler)}>
                    <Input name={'email'} label={'Email
address:'} />
```

```
                    <Box m={2} />

                    <Input name={'password'} type={'password'}
label={'Password:'} />

                    <Box m={5} />

                    <Box paddingX={2}>
                        <Button type={'submit'} fullWidth
variant={'contained'} color={'primary'}>
                            Sign In
                        </Button>
                    </Box>
                </form>
            </FormProvider>
        </>
    }

    export const AuthRegisterForm = () => {
        const { dispatch } = useUserState();

        const form = useForm<RegisterFormValues>({
            resolver: yupResolver(schema)
        });

        const mutation = useMutation('loginUser', (dto:
CreateUserRequestDto) => createUserRequest(dto));

        const submitHandler = async ({confirmPassword, ...dto}:
RegisterFormValues) => {
            try {
```

```
                const response = await
mutation.mutateAsync(dto);


                if (response.status !== 200) {
                    for (let key in response.errors) {
                        form.setError(response.errors[key].path
as FieldPath<CreateUserRequestDto>, {
                            message:
response.errors[key].messages[0],
                        });
                    }
                } else {
                    Cookies.set('token', response.token);
                    dispatch(new
SetUserAction(mapUserToState(response.user as UserDto)));
                }
            } catch (error) {}
        }


        return <>

            <FormProvider {...form}>
                <Typography align={'center'} variant={'h2'}>
                    Create an Account
                </Typography>

                <Box m={1} />

                <Typography align={'center'}>
                    Create an account to continue
                </Typography>

                <Box m={4} />
```

```
                <form autoComplete="off"
onSubmit={form.handleSubmit(submitHandler)}>
                    <Input name={'email'} label={'Email
address:'} />

                    <Box m={2} />

                    <Input name={'name'} label={'Username:'} />

                    <Box m={2} />

                    <Input name={'password'} type={'password'}
label={'Password:'} />

                    <Box m={2} />

                    <Input name={'confirmPassword'}
type={'password'} label={'Confirm Password:'} />

                    <Box m={5} />

                    <Box paddingX={2}>
                        <Button type={'submit'} fullWidth
variant={'contained'} color={'primary'}>
                            Sign Up
                        </Button>
                    </Box>
                </form>
            </FormProvider>
        </>
    }
```

```
    export const RankingCollectionView = () => {
        const { id } = useParams<{id: string}>();
        const { state } = useUsersState();

        const {
            data,
            isLoading
        } = useQuery(['collection', id], () =>
getRankingCollectionRequest(id));



        const {
            data: results,
            isLoading: resultsLoading
        } = useQuery(['results', data?.collection.id], () =>
getRankingRequest({
            take: 100,
            UserId: state.currentUser?.id as string,
            skip: 0,
            collectionId: data?.collection.id as string
        }))

        if (!data || isLoading) return <></>;

        console.log(data)

        return <>
            <Typography variant={'h1'}>
                Collection: {data.collection.createdAt}
            </Typography>

            <Box m={3} />
```

```
            <Grid container spacing={3}>
                <Grid item xs={8}>
                    <Paper>
                        <Box p={3}>
                            <Typography variant={'h3'}>
                                Results
                            </Typography>

                            <Box m={2} />

                            <RankingResultsList
data={results?.results ?? []} />
                        </Box>
                    </Paper>
                </Grid>
                <Grid item xs={4}>
                    <RankingCollectionCard
data={data.collection} UserData={state.Users.find(i => i.id ==
data.collection.UserId).endpoint} />
                </Grid>
            </Grid>
        </>
    }


    export const RankingCollectionCard = ({ data, UserData }:
Props) => {
        return (
            <Paper>
                <Box p={2}>
                    <Typography variant={'h4'}>
                        { data.createdAt }
                    </Typography>
```

```
                    <Typography>
                        { UserData }
                    </Typography>

                    <Box m={2} />

                    <Grid container spacing={1}>
                        <Grid item xs={6}>
                            <Box display={'flex'}
flexDirection={'column'} alignItems={'center'}>
                                <ResultScoreProgress
value={data.averagePerformanceScore.desktop} />

                                <Box m={1} />

                                <Typography>
                                    <DesktopWindowsOutlinedIcon
/>
                                </Typography>
                            </Box>
                        </Grid>
                        <Grid item xs={6}>
                            <Box display={'flex'}
flexDirection={'column'} alignItems={'center'}>
                                <ResultScoreProgress
value={data.averagePerformanceScore.mobile} />

                                <Box m={1} />

                                <Typography>
                                    <PhoneAndroidOutlinedIcon
/>
                                </Typography>
```

```
                            </Box>
                        </Grid>
                    </Grid>

                    <Box m={2} />

                    <Typography>
                        Total Results:
<strong>{data.totalResults}</strong>
                    </Typography>
                </Box>
            </Paper>
        )
    }

    export const RankingDashboard = () => {
        const {state: UsersState} = useUsersState();
        const {
            data: resultsResponse,
            isLoading: resultsLoading
        } = useQuery<RankingResultsResponseDto>(['results',
UsersState.currentUser?.id], () =>
            getRankingResultsRequest({
                skip: 0,
                take: 100,
                UserId: UsersState.currentUser?.id as string
            })
        );

        const {
            data: resultResponse,
            isLoading: resultLoading
```

```
        } = useQuery(
            ['lastResult', UsersState.currentUser?.id],
            () => getRankingResultRequest((resultsResponse as
RankingResultsResponseDto).results[0].id),
            {
                enabled: !!resultsResponse &&
resultsResponse.results.length > 0,
                refetchInterval: false,
                refetchIntervalInBackground: false
            }
        );


        const {
            data: infoResponse,
            isLoading: infoLoading
        } = useQuery(['info', UsersState.currentUser?.id],
            () =>
getRankingInfoRequest(UsersState.currentUser?.id as string)
        );
        const [results, setResults] = useState(5);
        const changeNumber = (id: string) => {
            setResults(parseInt(id))
            // amount = parseInt(id);
            console.log(id);
        }


        return !resultsResponse ||
!resultsResponse.results.length ?
            (
                <Alert severity={'warning'}>
                    You have no generated results yet. Please,
wait for finishing User jobs.
                </Alert>
```

```
            ) :
            (
            <>
                <Grid container spacing={3}>
                    <Grid item xs={12}>
                        { infoResponse?.info &&
<RankingGeneralInfo data={infoResponse.info} /> }
                    </Grid>

                    <Grid item xs={12}>
                        <RankingMinimizedResultInfo
                            data={resultResponse?.result as
RankingResultDto}
                            isLoading={resultLoading}
                            type={'desktop'}
                        />
                    </Grid>

                    <Grid item xs={12}>
                        <RankingMinimizedResultInfo
                            data={resultResponse?.result as
RankingResultDto}
                            isLoading={resultLoading}
                            type={'mobile'}
                        />
                    </Grid>

                    <Grid item xs={12}>
                        <Grid container spacing={3}>
                            <Grid item xs={6}>
                                <Paper>
                                    <Box p={3}>
```

```
                                        <Typography
variant={'h3'}>
                                            Last <DropMenu
                                                currentNumber =
{results}
                                                recordsLength =
{resultsResponse?.results.length}

onChange={changeNumber}
                                            /> results
                                        </Typography>
                                        <Box m={2} />

                                        <RankingResultsList
data={resultsResponse?.results.slice(0,results) ?? []} />
                                    </Box>
                                </Paper>
                            </Grid>

                            <Grid item xs={6}>
                                <Paper>
                                    <Box p={3}>
                                        <Typography
variant={'h3'}>
                                            Results Dynamic
                                        </Typography>
                                        <Typography
variant={'caption'}>
                                            Based on the last
10 results
                                        </Typography>

                                        <Box m={2} />
```

```
                                        <RankingResultsChart
data={mapResultsToChartData(resultsResponse?.results.slice(0,r
esults) ?? [])} />
                                    </Box>
                                </Paper>
                            </Grid>
                        </Grid>
                    </Grid>
                </Grid>
            </>
        )
    }
```

Додаток Г.
(обов'язковий)
Графічна частина

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ БАГАТОКОРИСТУВАЦЬКОГО РАНЖУВАННЯ ЗОБРАЖЕНЬ

Виконав: студент 2-го курсу, групи 2КН-21м
спеціальності 122 – Комп'ютерні науки
Дикун В. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент
Колодний В. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
«_____»_____2022 р.

Вінниця ВНТУ - 2022 рік

Рисунок Г.1 – Схема проведення багатокористувацького ранжування

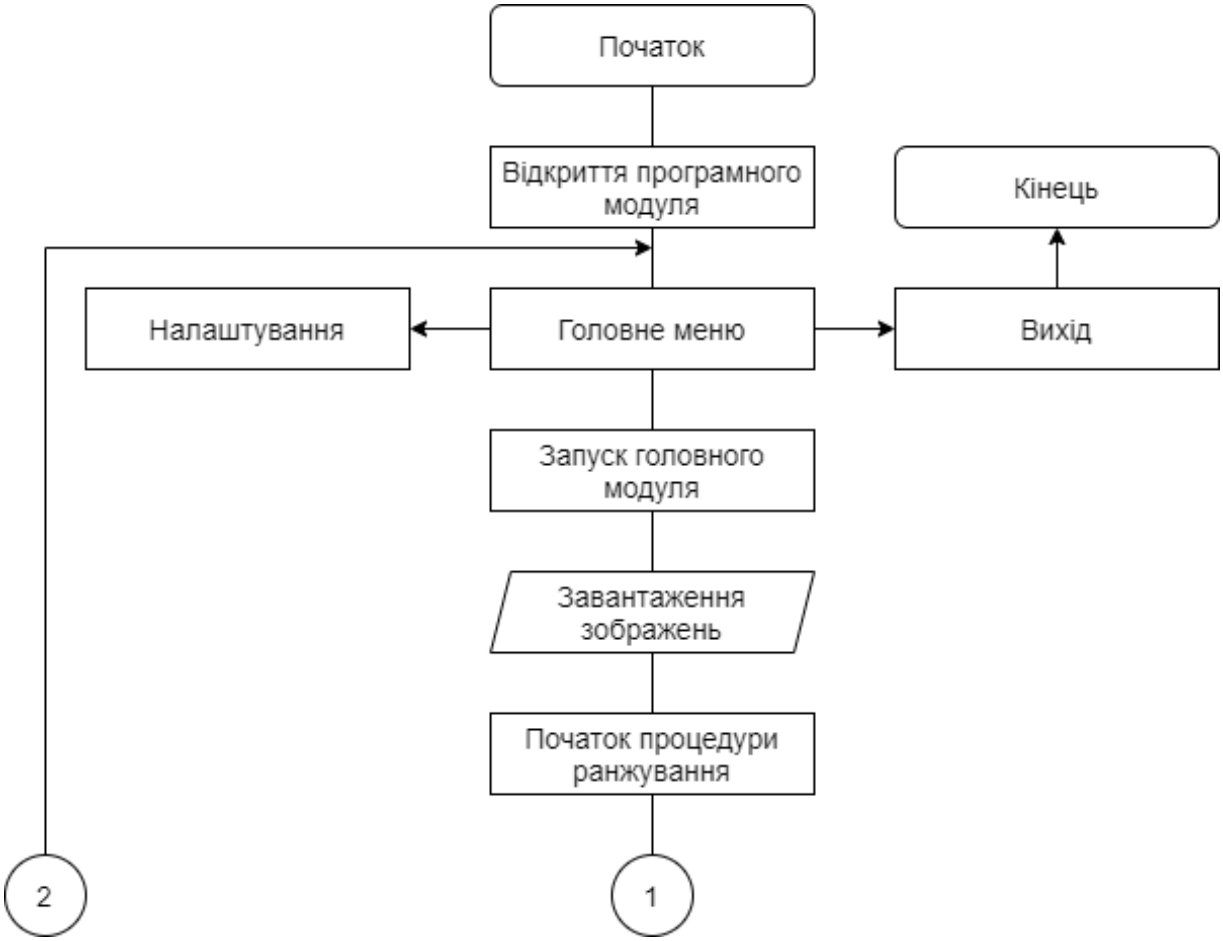


Рисунок Г.2 – Схема алгоритму роботи програмного модуля процедури ранжування

Продовження рисунку Г.2

Рисунок Г.3 – Структурна схема архітектури системи багатокористувацького ранжування зображень

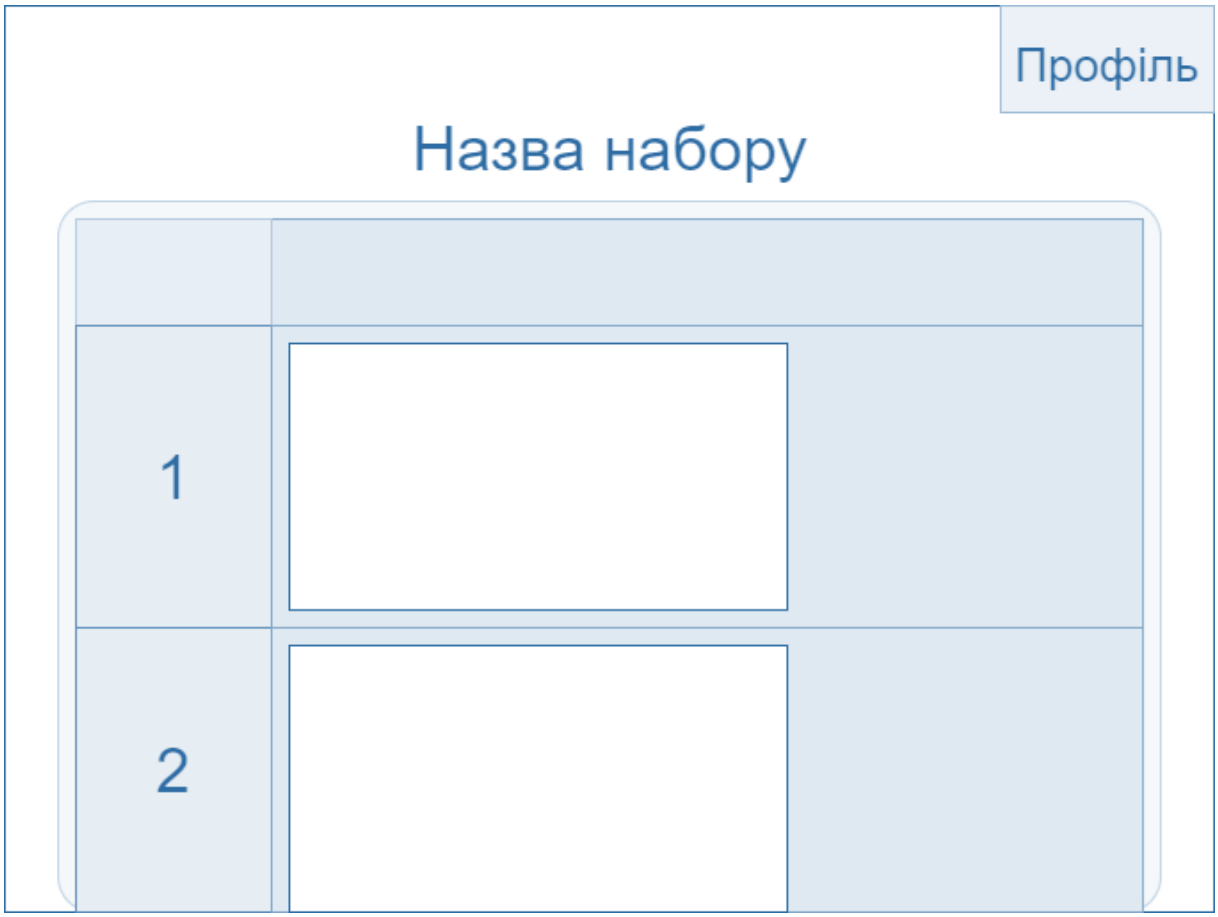


Рисунок Г.4 – Схематичне зображення результатів ранжування

Рисунок Г.5 – Результати роботи програмного забезпечення багатокористувацького ранжування зображень