Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Програмна система оцінювання якості графічних зображень

Виконав: студент 4 курсу
групи ЗПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Рябоконь А. М.
(прізвище та ініціали)

Керівник: д.т.н., професор каф. ПЗ Романюк О.Н.
(прізвище та ініціали)

Рецензент: д.т.н., проф. каф. ЗІ Лужецький В.А.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ О.Н. Романюк
«06» червня 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
« 21 » березня 2025 р.

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

Рябоконю Артему Миколайовичу

1. Тема роботи – «Програмна система оцінювання якості графічних зображень»

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: роздільна здатність зображень – до 1920×1080 пікселів; кольоровий режим – RGB (TrueColor); формати вхідних зображень – .png, .jpg; середовище розробки – Visual Studio Code; мова розробки – TypeScript з використанням фреймворку React (TSX); операційна система – Windows 10.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз існуючих об'єктивних та суб'єктивних методів оцінювання якості зображень, огляд аналогічних програмних систем, постановка задач дослідження, розробка методів для оцінювання якості зображень з використанням, розробка методів суб'єктивної оцінки якості (рейтинг користувача), реалізація функціоналу завантаження, перегляду й оцінки зображень, розробка графічного інтерфейсу

користувача, інтеграція бібліотек візуалізації результатів оцінки, тестування функціоналу, інструкція користувача, висновки, список використаних джерел, додатки.

5. Перелік графічного матеріалу: демонстрація роботи аналогічних рішень, блок-схема функціонування програмної системи, діаграма компонентів React-застосунку, ER-діаграма моделі даних, UI-макети інтерфейсу користувача, візуалізація метрик якості (графіки, гістограми, порівняння оригінал/результат).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О.Н., д.т.н., професор кафедри ПЗ	24.03.2025	01.06.2025

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів і засобів для оцінювання якості графічних зображень з використанням об'єктивних та суб'єктивних метрик	25.03.25 – 31.03.25	Вик.
2	Розробка методів оцінювання якості зображень з використанням об'єктивних і суб'єктивних метрик	03.04.25 – 17.04.25	Вик.
3	Розробка програмних модулів системи оцінювання якості графічних зображень	19.04.25 – 03.05.25	Вик.
4	Тестування програмного забезпечення	05.05.25 – 18.05.25	Вик.
5	Оформлення матеріалів до захисту БКР	20.05.25 – 30.05.25	Вик.

Студент _____ (підпис) Рябоконь А.М. (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) Романюк О.Н. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.932.2:004.93

Рябоконь А. М. Програмна система оцінювання якості графічних зображень : бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 75 с.

На укр. мові. Бібліогр.: 21 назв ; рис.: 21 ; табл.: 2.

У роботі розглянуто питання оцінювання якості графічних зображень на основі об'єктивних метрик (MSE, PSNR, SSIM, FSIM) і суб'єктивних методів. Проаналізовано існуючі підходи до оцінки візуальної якості та обґрунтовано доцільність створення програмної системи, що поєднує аналітичні та експертні засоби.

Запропоновано адаптивний метод із динамічним визначенням вагових коефіцієнтів для класичних метрик залежно від характеру спотворень. Також реалізовано гібридний підхід, що поєднує об’єктивні показники з оцінками користувачів, забезпечуючи узгодженість з візуальним сприйняттям.

Розроблено застосунок для комплексного оцінювання якості зображень із підтримкою об'єктивних метрик і збором суб'єктивних оцінок користувачів. Систему протестовано на різних наборах даних, що підтвердило її ефективність.

Результати дослідження можуть бути використані для автоматизації процесів контролю якості у сферах, пов’язаних з аналізом і обробкою зображень.

Ключові слова: оцінка якості зображень, об'єктивні метрики, MSE, PSNR, SSIM, FSIM, суб'єктивні методи, аналіз зображень, програмна система.

ABSTRACT

UDC 004.932.2:004.93

Ryabokon A. M. Software system for assessing the quality of graphic images: bachelor's thesis in specialty 121 – Software Engineering, educational program – Software Engineering. Vinnytsia: VNTU, 2025. 75 c.

In Ukrainian. Bibliogr: 21 titles ; fig.: 21 ; tbl.: 2.

The paper considers methods of assessing the quality of graphic images based on objective metrics (MSE, PSNR, SSIM, FSIM) and subjective methods. Existing approaches to visual quality assessment are analyzed and the feasibility of creating a software system that combines analytical and expert tools is substantiated.

An adaptive method with dynamic determination of weighting coefficients for classical metrics depending on the nature of the distortion is proposed. A hybrid approach combining objective metrics with user ratings is also implemented, ensuring consistency with visual perception.

An application for comprehensive image quality assessment with support for objective metrics and collection of subjective user ratings is developed. The system has been tested on various datasets, which has confirmed its effectiveness.

The results of the study can be used to automate quality control processes in areas related to image analysis and processing.

Keywords: image quality assessment, objective metrics, MSE, PSNR, SSIM, FSIM, subjective methods, image analysis, software system.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний розвиток комп’ютерних систем демонструє швидке збільшення обсягу та складності візуальної інформації, що потребує обробки. Комп’ютерна графіка міцно увійшла до багатьох сфер діяльності, включаючи наукові дослідження, інженерне проектування, медичну візуалізацію, індустрію розваг та системи віртуальної реальності. Зростають вимоги до візуальної якості, зокрема до реалістичності та деталізації графічних зображень.

У контексті забезпечення високої якості графічних зображень, значну увагу приділяють процесам оцінювання. На сьогодні існує багато метрик і методів оцінки якості, проте більшість з них орієнтовані на вже сформовані зображення та не завжди враховують особливості їх створення. Крім того, суб'єктивне людське сприйняття відіграє важливу роль у формуванні враження про якість, що обумовлює необхідність поєднання об’єктивних і суб’єктивних підходів до оцінювання.

У зв'язку з вищесказаним, розробка програмної системи оцінювання якості графічних зображень, яка комбінуватиме аналітичні та експертні методи, є актуальною та своєчасною.

Аналітичне оцінювання, ґрунтуючись на об'єктивних метриках та алгоритмах, дозволить кількісно визначити різні аспекти якості зображень: рівень деталізації, наявність артефактів, точність передачі кольорів та освітлення. Це забезпечить автоматичний контроль якості на різних етапах створення графічного контенту та можливість порівняння ефективності різних методів візуалізації.

Разом з тим, експертне оцінювання, залучаючи кваліфікованих фахівців, дозволить врахувати суб'єктивні аспекти сприйняття якості, такі як естетична привабливість, реалістичність та відповідність поставленим завданням. Поєднання експертних суджень з результатами аналітичного оцінювання забезпечить повну та всебічну оцінку якості графічних зображень.

Отже, розробка програмної системи, яка інтегрує аналітичні та експертні методи оцінювання якості графічних зображень, сприятиме:

– підвищенню об'єктивності та комплексності оцінки якості графічного контенту;

– оптимізації процесів створення графічних зображень, виявляючи та усуваючи недоліки на ранніх етапах;

– порівняльному аналізу ефективності різних методів і алгоритмів комп'ютерної графіки;

– формуванню єдиних стандартів якості в різних сферах застосування комп'ютерної графіки;

– покращенню взаємодії між розробниками та користувачами графічного контенту на основі об’єктивних даних про його якість.

Враховуючи зростаючу роль комп'ютерної графіки та відсутність комплексних інструментів оцінки її якості, розробка програмної системи, яка поєднує аналітичний та експертний підходи, є важливим напрямком для підвищення ефективності та якості візуалізації.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення ефективності оцінювання якості графічних зображень шляхом розробки програмних модулів, які поєднують експертні та аналітичні методи для автоматизованого аналізу та порівняння зображень.

Для досягнення мети необхідно виконати такі завдання:

– розробити алгоритм оцінювання якості зображень на основі аналітичних методів (метрики SSIM, PSNR, MSE тощо);

– розробити програмний модуль для суб'єктивної оцінки якості зображень, що враховуватиме експертні параметри та зворотний зв’язок користувачів;

– реалізувати програмний модуль порівняння зображень для аналізу відмінностей між оригінальним та обробленим зображенням за допомогою методів адаптивного та гібридного оцінювання;

– розробити графічний інтерфейс для взаємодії користувача з системою оцінювання;

– протестувати програмні модулі на різних графічних зображеннях для перевірки їх ефективності та коректності роботи.

Об’єкт дослідження – процес оцінювання якості графічних зображень.

Предмет дослідження – методи та засоби аналізу й порівняння графічних зображень.

Методи дослідження. У процесі дослідження використовувались: теорія чисел та чисельні методи, теорія диференціально-інтегрального числення, лінійна алгебра, методи аналітичної геометрії для розробки моделей та методів аналітичної та експертної оцінки якості графічних зображень; комп’ютерне моделювання для аналізу, імітації спотворень та перевірки достовірності отриманих результатів щодо якості візуального сприйняття зображень.

Наукова новизна отриманих результатів:

1. Вперше запропоновано адаптивний метод оцінювання якості зображень, особливість якого полягає в том, що враховуються характеристики зображень для динамічного визначення вагових коефіцієнтів класичних об'єктивних метрик: PSNR, SSIM, MSE, FSSIM, що дає можливість більш достовірно оцінити реалістичність сформованого зображення.

2. Розроблено гібридний метод оцінювання якості зображень, який поєднує адаптивне комбінування об'єктивних метрик: PSNR, SSIM, FSSIM, MSE із суб'єктивними оцінками сприйняття якості, що забезпечує підвищення узгодженості об’єктивної оцінки з візуальним сприйняттям людини. Метод передбачає аналіз характеру спотворень зображення з подальшим динамічним визначенням вагових коефіцієнтів для кожної метрики.

Практичне значення отриманих результатів полягає в тому, що на основі проведеного аналізу існуючих методів оцінювання якості зображень розроблено алгоритми та програмну систему для оцінювання якості зображень.

Особистий внесок здобувача. Усі наукові результати, викладені у БКР, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: аналітичні методи оцінки якості зображень [1], використання метрики psnr для оцінки якості зображень [2], методи математичного моделювання для об'єктивної та експертної оцінки якості цифрових зображень [3], оцінка якості зображень на основі аналітичних та експертних методів [4], оцінювання якості зображень у комп’ютерному моделюванні технологічних процесів на основі показників psnr та mse [5], комбіноване використання суб’єктивних і об’єктивних метрик оцінювання якості цифрових зображень [6].

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення БКРї доповідалися та обговорювалися на Міжнародних і Всеукраїнських конференціях: LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025), Всеукраїнська науково-практична Інтернет-конференція «Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку», XXV Всеукраїнська науково-технічна конференція молодих вчених, та здобувачів вищої освіти «Стан, досягнення і перспективи інформаційних систем і технологій», XXXIII Міжнародна науково-практична конференція «Інформаційні технології: наука, техніка, технологія, освіта, здоров’я» (MicroCAD–2025), X Всеукраїнська науково-методична конференція, XII Всеукраїнська науково-практична конференція молодих учених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ – 2025» (ІТ-2025).

Структура та обсяг БКР. БКР складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить 21 найменування та 4 додатки.

1 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ГРАФІЧНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ ОБ'ЄКТИВНИХ ТА СУБ'ЄКТИВНИХ МЕТРИК

1.1 Аналіз стану технологій оцінювання якості графічних зображень з використанням об'єктивних та суб'єктивних метрик

Оцінка якості графічних зображень є важливим питанням у багатьох галузях, таких як комп'ютерна графіка, медична візуалізація, відеоспостереження, автоматичне управління та штучний інтелект. Зі збільшенням використання цифрових технологій зростає потреба у високоякісних зображеннях, що вимагає розробки ефективних методів аналізу та оцінки якості візуального контенту. На сьогодні існує два основних підходи до оцінки якості графічних зображень: суб'єктивні та об'єктивні методи.

Суб'єктивні методи ґрунтуються на людському сприйнятті і передбачають залучення експертів або груп користувачів, які оцінюють якість зображень відповідно до власного візуального сприйняття. Одним з найпоширеніших суб'єктивних методів є MOS – Mean Opinion Score [7], який виставляє середній бал якості на основі оцінок кількох людей. Незважаючи на високу точність таких методів, їхніми основними недоліками є залежність від людського фактору, тривалий час тестування та висока вартість досліджень.

Об'єктивні методи оцінки якості зображення використовують математичні моделі та алгоритми для автоматизації процесу аналізу [8]. Ці методи можна розділити на три основні групи залежно від наявності або відсутності еталонних зображень. Повні еталонні методи використовують оригінальне зображення як еталон для порівняння. До них відносяться: PSNR [9] – пікове відношення сигнал/шум, SSIM [10] – індекс структурної подібності і FSIM [11] – індекс функціональної подібності. Ці методи дозволяють оцінити ступінь відхилення зображення від оригіналу і є корисними для стиснення, згладжування та інших аналізів обробки графічної інформації. Інша категорія включає методи зі

скороченим еталоном, які використовують лише деякі статистичні характеристики оригінального зображення. Це зменшує вимоги до обсягу вихідних даних, зберігаючи при цьому достатню точність оцінювання. Такі методи використовуються, коли повні еталонні зображення недоступні або їх зберігання є ресурсномістким.

Безеталонні метрики [12] працюють без порівняння з оригінальним зображенням і оцінюють якість на основі аналізу артефактів, таких як шум, розмиття, блокування та інші дефекти. Ці методи є найскладнішими, оскільки вимагають визначення якості зображення без доступу до еталона. Для цього використовуються алгоритми статистичного аналізу та методи машинного навчання для пошуку закономірностей візуальних дефектів. В останні роки значного розвитку набули методи оцінки якості зображень на основі штучного інтелекту та глибокого навчання.

Нейронні мережі, особливо згорткові нейронні мережі – CNN, можуть ефективно аналізувати зображення та оцінювати їхню якість на рівні, близькому до людського сприйняття. Ці моделі навчаються на великих масивах даних і можуть враховувати складні фактори, що впливають на якість сприйняття, такі як колірний баланс, деталізація та локальні спотворення. Розробка ефективних програмних модулів для оцінки якості графічних зображень є значним викликом, що вимагає врахування сучасних алгоритмів, продуктивності та адаптивності до різних типів візуального контенту. Впровадження таких модулів підвищує ефективність цифрової обробки зображень, оптимізує роботу систем комп'ютерного зору та покращує якість мультимедійного контенту.

1.2 Аналіз методів розв’язання задачі

У ході розробки програмного забезпечення для оцінки якості графічних зображень було виконано детальне дослідження існуючих об'єктивних метрик, що дають змогу з певною точністю вимірювати ступінь спотворення зображень після їх обробки або передачі. Головний акцент робився на тих методах, які не

тільки забезпечують достовірні числові показники якості, але й володіють достатньою обчислювальною продуктивністю для застосування у прикладних програмних компонентах.

Найбільш розповсюджені метрики, що були взяті за основу аналітичного оцінювання якості зображень в розробленій системі: MSE, PSNR, SSIM та FSIM.

Метрика MSE обчислює середнє квадратичне відхилення між пікселями оригінального та обробленого зображень. Попри простоту впровадження, цей підхід не враховує особливостей сприйняття зображення людиною, що обмежує його застосування у задачах, де важливе суб'єктивне враження від якості.

Метрика PSNR базується на значенні MSE та показує співвідношення максимальної можливої потужності сигналу до потужності шуму, яка представлена як помилка. Високе значення PSNR, зазвичай, вказує на високу якість зображення, проте, подібно до MSE, ця метрика чутлива до глобальних змін, але не завжди корелює з візуальним сприйняттям.

Більш точну та сучасну оцінку якості надає метрика SSIM, що аналізує локальні характеристики яскравості, контрасту та структури. SSIM вважається більш наближеною до людського візуального сприйняття та дозволяє краще оцінити локальні спотворення в зображеннях.

Метрика FSIM орієнтується на аналіз фрагментів зображень з урахуванням фазових характеристик та виявляє подібність зображень за ознаками, що найбільше впливають на візуальну якість. FSIM забезпечує високу точність у випадках, коли важливо зберегти структурну цілісність і локальні особливості зображення.

Всі згадані метрики є об'єктивними, тобто вони не залежать від оцінки людини та дозволяють отримати числовий показник якості. Проте, для досягнення більшої достовірності оцінювання в системі також було впроваджено компонент експертного аналізу – збір суб'єктивних оцінок від користувачів.

Поєднання об'єктивних і суб'єктивних методів дає змогу реалізувати гібридний підхід, що поєднує точність машинного аналізу з гнучкістю

людського сприйняття. Це особливо важливо у випадках, коли об'єктивні показники не збігаються з візуальним враженням.

Таким чином, модуль оцінки якості зображень спирається на комплексний підхід, що містить як класичні математичні метрики: MSE, PSNR, так і більш сучасні індекси, орієнтовані на збереження структури та особливостей зображень: SSIM, FSIM, а також інтеграцію суб'єктивних оцінок для забезпечення максимальної достовірності результатів. Це забезпечує високу гнучкість і універсальність запропонованого рішення в різних сферах використання – від побутових додатків до промислових систем контролю якості.

1.3 Аналіз аналогів

При розробці програмного модуля для оцінки якості графічних зображень необхідно було проаналізувати існуючі рішення, які виконують схожі функції. Це дозволило виявити основні переваги та недоліки вже розроблених систем і сформулювати вимоги до унікального модуля. Аналіз було проведено на прикладі трьох програмних рішень, що використовуються для оцінки якості зображень на основі експертних та аналітичних методів.

Першим подібним продуктом є OpenCV – Open Source Computer Vision Library [13]. Це потужна бібліотека комп'ютерного зору (див. рисунок 1.1), яка включає в себе широкий спектр інструментів для аналізу зображень, в тому числі можливість оцінювати якість зображення з точки зору шуму, розмиття, контрасту, артефактів тощо.

Рисунок 1.1 – Бібліотека OpenCV

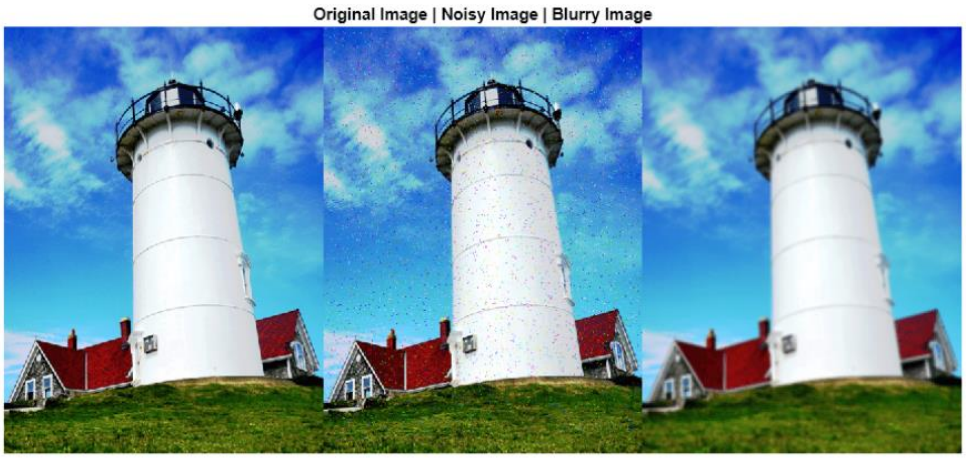
Основними перевагами OpenCV є її гнучкість, підтримка багатьох мов програмування (Python, C++, Java) та велика спільнота розробників. Однак OpenCV є низькорівневим інструментом і вимагає глибокого розуміння алгоритмів, що ускладнює його використання в React-додатках без додаткових API.

Іншим схожим продуктом є програмний продукт Natural Image Quality Evaluator або NIQE [14]. Це алгоритм, який не потребує еталонного зображення для оцінки якості зображення (див. рисунок 1.2); NIQE аналізує статистичні властивості зображення і прогнозує якість на основі навченої моделі; головна перевага NIQE полягає в тому, що він не залежить від еталонного зображення і підходить для автоматичної обробки контенту.

```
I = imread("lighthouse.png");
Inoise = imnoise(I,"salt & pepper",0.02);
Iblur = imgaussfilt(I,2);
```

Display the images.

```
montage({I,Inoise,Iblur},Size=[1 3])
title("Original Image | Noisy Image | Blurry Image")
```



Calculate the NIQE score for each image using the default model. Display the score.

```
niqeI = niqe(I);
disp("NIQE score for the original image is: "+niqeI)
```

```
NIQE score for the original image is: 2.5455
```

```
niqeInoise = niqe(Inoise);
disp("NIQE score for the noisy image is: "+niqeInoise)
```

Рисунок 1.2 – Приклад використання технології NIQE

Однак точність NIQE може бути нижчою, ніж у методів, що використовують повне порівняння з еталонним зображенням, особливо для певних типів зображень, таких як медичні або технічні зображення.

Третім аналогом є вебсервіс Google Cloud Vision API [15], який надає інструменти для аналізу зображень, включаючи розпізнавання об'єктів, оцінку якості та виявлення дефектів (див. рисунок 1.3). Він використовує алгоритми машинного навчання для визначення якості зображень та аналізу різних візуальних параметрів.

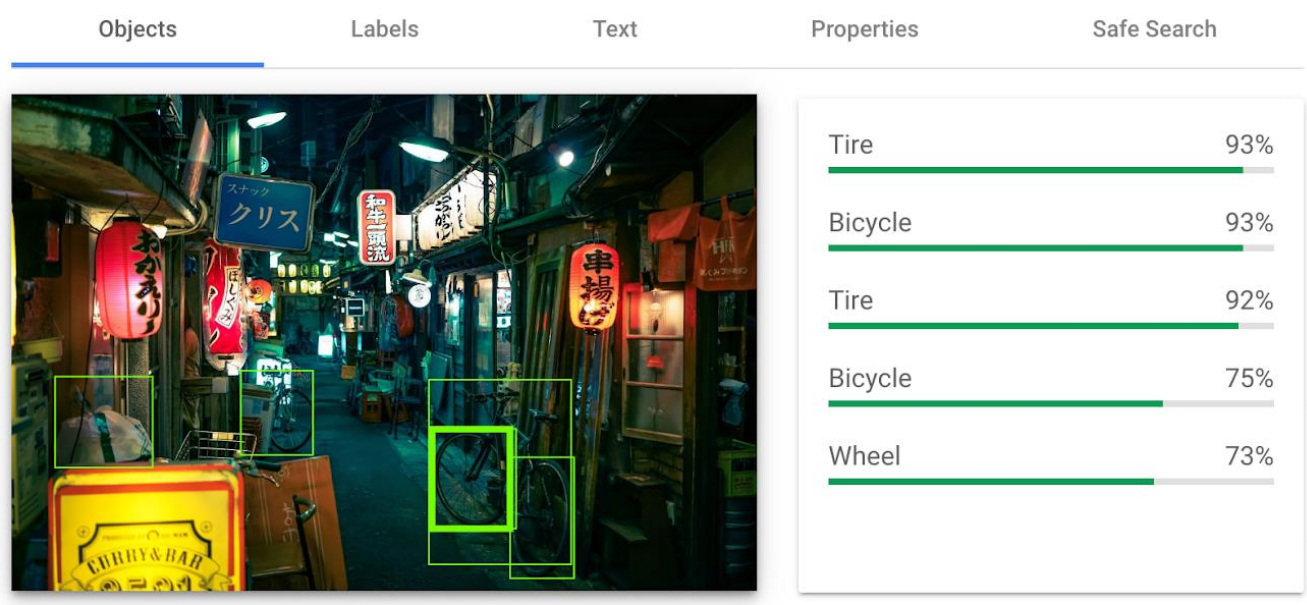


Рисунок 1.3 – Використання автоматичного підписування фотографії за допомогою Vision API

Основною перевагою Google Cloud Vision API є його легка інтеграція у вебзастосунки, в тому числі React-проекти. Недоліками є модель оплати за використання, залежність від підключення до інтернету та обмежений контроль над алгоритмами оцінювання.

Після аналізу аналога було створено таблицю порівняльних характеристик для оцінки за ключовими критеріями:

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	OpenCV	NIQE	Google Cloud Vision API	Власний застосунок
1	2	3	4	5
Об’єктивний метод оцінки	+	-	+	+
Аналітичний метод оцінки	-	+	+	+

Продовження таблиці 1.1

1	2	3	4	5
Потрібність еталонного зображення	+	-	-	±
Простота інтеграції у React	-	±	+	+
Гнучкість налаштувань	+	-	-	+
Необхідність підключення до інтернету	-	-	+	+
Вартість (безкоштовність)	+	+	-	+
Загальна оцінка (%)	57%	43%	57%	71%

Таким чином, аналіз аналогічних продуктів дозволив визначити, що існуючі рішення мають як переваги, так і недоліки. Власний модуль поєднує експертні та аналітичні методи оцінки для підвищення гнучкості та точності оцінки. Модуль було розроблено на React з використанням новітніх технологій для інтеграції у вебзастосунки. У результаті було створено інтуїтивно зрозумілий інструмент аналізу зображень, який можна використовувати в рідному середовищі без необхідності підключення до сторонніх сервісів.

1.4 Постановка задач дослідження

На основі аналізу існуючих аналогічних продуктів та виявлення їх сильних та слабких сторін були сформульовані основні завдання, які необхідно виконати для успішної розробки програмного модуля для системи оцінки якості графічних зображень. Програмне забезпечення буде розроблено з використанням технології React у середовищі розробки VS Code, із застосуванням експертних та аналітичних методів оцінювання. Основними завданнями є:

– розробка алгоритму оцінки якості зображення, що поєднує аналітичні методи (об'єктивні показники, такі як PSNR, SSIM) та експертні методи (суб'єктивна оцінка користувача);

– розробка модуля завантаження та попередньої обробки зображень, який підтримує різні формати файлів і виконує нормалізацію, масштабування та конвертацію для коректної обробки;

– розробка модуля порівняння зображень. Це дозволяє аналізувати відмінності між оригінальними та обробленими зображеннями за допомогою запропонованих метрик якості;

– реалізація користувацького інтерфейсу, що дозволяє зручно взаємодіяти з системою та візуалізувати результати оцінки та порівняння зображень у реальному часі;

– створення гнучкої системи конфігурації для вибору методів оцінки, параметрів та відповідного порівняння результатів;

– інтеграція модуля зберігання та експорту результатів, що дозволяє зберігати оброблені дані у вигляді звітів і таблиць;

– автоматизація процесу аналізу. Якість зображень може бути оцінена без ручного втручання за допомогою запрограмованих алгоритмів та нейромережевих підходів;

– тестування програмних продуктів для перевірки точності застосованих методів, працездатності алгоритмів і зручності використання для кінцевого користувача.

Виконання цих завдань дозволяє створювати ефективні програмні продукти для оцінки якості графічних зображень, які відповідають сучасним вимогам до точності, швидкості та зручності використання.

1.5 Висновки

У проведеному аналізі розглядаються сучасні методи оцінки якості графічних зображень. Суб'єктивні методи, такі як MOS, використовують людину для оцінки якості, але є дорогими і залежними від людського фактору. Об'єктивні методи використовують математичні алгоритми і поділяються на повністю референтні (PSNR, SSIM, FSIM), частково референтні (RR) і нереферентні (NR) та аналізують артефакти без порівняння з оригіналом. Крім того, розглядаються підходи глибокого навчання, зокрема згорткові нейронні мережі, які забезпечують високу точність оцінювання.

У порівняльному аналізі розглядаються OpenCV, NIQE та Google Cloud Vision API: OpenCV є потужним, але його важко інтегрувати у вебзастосунки; NIQE не потребує еталонних зображень, але може бути менш точним; Google Cloud Vision API легко інтегрується у вебзастосунки, але має платну модель і вимагає підключення до Інтернету. Була підготовлена порівняльна таблиця, яка показує, що власний модуль перевершує аналогічні продукти з точки зору гнучкості, простоти інтеграції та незалежності від сторонніх сервісів.

На основі проведеного аналізу за допомогою комбінації експертних оцінок та методів аналізу було сформульовано вимоги до кастомного модуля, який забезпечує точність та гнучкість оцінювання, легко інтегрується в React-застосунок та є незалежним від зовнішніх сервісів

2 РОЗРОБКА НОВИХ МЕТОДІВ ОЦІНЮВАННЯ ЯКОСТІ ЗОБРАЖЕНЬ

2.1 Розробка методу адаптивного вибору критерію оцінювання зображень

Незважаючи на широке застосування окремих об'єктивних метрик оцінювання якості зображень, кожна з них має свої сильні та слабкі сторони. Наприклад, PSNR чутлива до пікових помилок, але може не відображати структурні спотворення, які добре сприймаються людським зором. SSIM краще враховує структурну схожість, але може бути менш чутливою до незначних змін пікселів. FSSIM фокусується на градієнтних ознаках, що є важливим для візуального сприйняття, але може не повністю відображати інші типи спотворень.

Для подолання обмежень окремих метрик пропонується метод адаптивного вибору критерію оцінювання зображень Adaptive Evaluation Metric Combination. Основна ідея полягає у динамічному виборі найбільш релевантної метрики або комбінації метрик залежно від характеристик спотворень зображення. Це дозволить отримати більш об'єктивну та узгоджену з людським сприйняттям оцінку якості.

Розроблений метод AEMC включає такі етапи:

1. Аналіз характеристик спотворень – на цьому етапі проводиться аналіз тестового зображення для виявлення домінуючих типів спотворень. Це може включати аналіз розподілу піксельних значень, частотних характеристик, наявності блокових артефактів, розмиття тощо. Для прикладу, можна використовувати статистичні характеристики (середнє значення, дисперсія, асиметрія, ексцес), аналіз спектра потужності (за допомогою перетворення Фур'є) або методи виявлення конкретних типів артефактів (наприклад, блочності).

2. Визначення вагових коефіцієнтів для метрик – на основі виявлених характеристик спотворень визначаються вагові коефіцієнти для кожної з

розглянутих об'єктивних метрик (PSNR, MSE, SSIM, FSSIM). Метрикам, які є більш чутливими до виявлених типів спотворень, присвоюються вищі вагові коефіцієнти. Наприклад:

- якщо виявлено значну кількість випадкового шуму, більша вага може бути надана PSNR та MSE, оскільки вони добре відображають пікові помилки;
- при наявності структурних спотворень (розмиття, зсув) більшу вагу слід надати SSIM та FSSIM;
- у випадку переважання артефактів стиснення (блочність) можна розробити спеціалізовані метрики або адаптувати ваги існуючих метрик для кращого їх виявлення.

3. Комбінування метрик: Після визначення вагових коефіцієнтів, загальна оцінка якості зображення Q_{AEMC} обчислюється як зважена сума індивідуальних значень метрик:

$$Q_{AEMC} = \omega_{PSNR} \cdot Q_{PSNR} + \omega'_{MSE} \cdot Q'_{MSE} + \omega_{SSIM} \cdot Q_{SSIM} + \omega_{FSSIM} \cdot Q_{FSSIM} \quad (2.1)$$

де:

- Q_{PSNR}, Q_{SSIM}, Q_{FSSIM} – нормалізовані значення відповідних метрик (наприклад, приведені до діапазону [0, 1]).
- Q'_{MSE} – обернене нормалізоване значення MSE (оскільки менші значення MSE кращі). Наприклад, $Q'_{MSE} = 1 - Q_{MSE}$, де Q_{MSE} – нормалізоване значення MSE.
- ω_{PSNR}, ω'_{MSE}, ω_{SSIM}, ω_{FSSIM} – вагові коефіцієнти для кожної метрики, що задовольняють умові: $\omega_{PSNR} + \omega'_{MSE} + \omega_{SSIM} + \omega_{FSSIM} = 1$, $\omega_I \geq 0$. Визначення вагових коефіцієнтів базується на попередньому аналізі великої кількості зображень з різними типами спотворень та їх суб'єктивних оцінок. Можливе використання методів машинного навчання для автоматичного визначення оптимальних вагових коефіцієнтів.

4. Виведення адаптованої оцінки – отримана комбінована оцінка Q_{AEMC} представляє собою адаптивну міру якості зображення, яка враховує характеристики спотворень та відносну важливість різних об'єктивних метрик.

Розглянемо приклад, де тестове зображення містить як розмиття, так і невелику кількість шуму.

1. Аналіз характеристик спотворень – аналіз зображення може показати наявність високочастотних компонентів, що свідчить про шум, а також зниження амплітуди високих частот, що вказує на розмиття.

2. Визначення вагових коефіцієнтів – на основі аналізу, метриці SSIM та FSSIM, які добре реагують на структурні зміни, присвоюються вищі вагові коефіцієнти (наприклад, $\omega_{SSIM} = 0.4$, $\omega_{FSSIM} = 0.3$). Метрикам PSNR та MSE, чутливим до пікових помилок (шуму), присвоюються менші ваги (наприклад, $\omega_{PSNR} = 0.2$, $\omega'_{MSE} = 0.1$).

3. Комбінування метрик – припустимо, що після обчислення та нормалізації отримано наступні значення метрик: $Q_{PSNR} = 0.85$, $Q'_{MSE} = 0.92$, $Q_{SSIM} = 0.78$, $Q_{FSSIM} = 0.82$. Тоді адаптована оцінка якості буде:

$$Q_{AEMC} = 0.2 \cdot 0.85 + 0.1 \cdot 0.92 + 0.4 \cdot 0.78 + 0.3 \cdot 0.82 = 0.82 \quad (2.2)$$

4. Виведення адаптованої оцінки – отримана оцінка якості $Q_{AEMC} = 0.82$ є компромісом між оцінками окремих метрик, враховуючи наявність як шуму, так і розмиття.

На етапі комбінування метрик використовується формула зваженої суми:

$$Q_{AEMC} = \sum_i \omega_i \cdot Q'_i \quad (2.3)$$

де:

- Q_i – нормалізоване значення і-ої метрики (або обернене нормалізоване значення для MSE)$
- ω_i – ваговий коефіцієнт для і-ої метрики, що залежить від характеристик спотворень$
- $\sum_i \omega_I = 1$.

Процес нормалізації метрик до діапазону [0, 1] може бути виконаний за допомогою лінійного масштабування:

$$Q_{norm} = \frac{Q - Q_{min}}{Q_{max} - Q_{min}} \quad (2.4)$$

де Q_{min} та Q_{max} – мінімальне та максимальне очікувані значення метрики відповідно (які можуть бути визначені емпірично або теоретично).

Для MSE використовується обернене значення після нормалізації:

$$Q'_{MSE} = 1 - \frac{MSE - MSE_{min}}{MSE_{max} - MSE_{min}} \quad (2.5)$$

Переваги методу AEMC:

- адаптивність – метод динамічно враховує характеристики спотворень зображення при оцінюванні якості;
- гнучкість – можливість комбінування різних об'єктивних метрик та налаштування вагових коефіцієнтів;
- потенційно вища узгодженість з людським сприйняттям – завдяки врахуванню різних типів спотворень та їхньої візуальної значущості.

Напрямки подальшого розвитку:

- розробка більш ефективних методів аналізу характеристик спотворень;
- автоматичне визначення оптимальних вагових коефіцієнтів з використанням машинного навчання та баз даних суб'єктивних оцінок якості;
- розширення набору використовуваних об'єктивних метрик, включаючи метрики, орієнтовані на конкретні типи спотворень;
- дослідження можливості застосування нелінійних методів комбінування метрик.

Розроблений метод AEMC є перспективним підходом до об'єктивної оцінки якості зображень, що дозволяє враховувати різноманітні характеристики спотворень та підвищити узгодженість автоматизованих оцінок з суб'єктивним сприйняттям людини.

2.2 Розробка гібридного методу оцінювання якості зображень на основі комбінування об’єктивних і суб’єктивних метрик

Гібридний метод оцінювання якості зображень поєднує об’єктивні та суб’єктивні метрики з метою досягнення більш точної та репрезентативної

оцінки, яка відображає як технічні характеристики, так і візуальне сприйняття людиною. Розробка такого методу передбачає інтеграцію результатів, отриманих за допомогою об'єктивних метрик – SSIM, FSIM, PSNR, MSE – та суб'єктивних оцінок – MOS, шкала Лайкерта, метод Кендала – в єдиний комбінований показник якості.

Об'єктивні метрики розраховуються автоматично за допомогою алгоритмів, що оцінюють ступінь схожості між оригінальним та спотвореним зображенням:

- PSNR обчислюється як:

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right) \quad (2.6)$$

де MAX – максимальне можливе значення пікселя (наприклад, 255 для 8-бітного зображення), а MSE – середньоквадратична похибка між двома зображеннями, яка визначається як:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{i=0}^{N-1}[I(i,j) - K(i,j)]^2 \quad (2.7)$$

де I(i,j) та K(i,j) - значення пікселів у і-му рядку та j-му стовпці еталонного та тестового зображень відповідно.

- SSIM розраховується за формулою:

$$SSIM = \frac{(2\mu_X\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2.8)$$

де μ_x,μ_y – середні значення, σ_x^2,σ_y^2 – дисперсії, σ_{xy} – коваріація локальних вікон x та y, а c_1,c_2 – невеликі константи для стабілізації.

- FSIM базується на порівнянні фазових характеристик зображення і оцінює схожість важливих візуальних характеристик, таких як границі та контраст.

Суб'єктивні метрики, як MOS та Лайкерт, обчислюються за результатами опитування експертів. Кожен експерт надає оцінку якості зображення за шкалою, після чого проводиться усереднення результатів. Наприклад, середнє значення MOS для зображення і обчислюється як:

$$MOS(i) = \left(\frac{1}{K}\right) \cdot \sum_{j=1}^{K} S_{ij} \quad (2.9)$$

де K – кількість експертів, s_{ij} – оцінка, надана j-м експертом для зображення i.

Метод Кендала використовується для оцінки узгодженості оцінок між експертами. Коефіцієнт Кендала τ_b розраховується для кожної пари експертів, після чого обчислюється середнє значення, що дозволяє виявити ступінь консенсусу. Підвищена узгодженість підвищує довіру до суб'єктивних результатів.

Для поєднання результатів суб'єктивних та об'єктивних метрик у гібридному методі запропоновано використовувати нормалізоване середнє зважене значення. Кожна метрика нормалізується до діапазону [0,1], після чого застосовується вагова формула:

$$Q_{hybrid} = \omega_1 * SSIM_{norm} + \omega_2 * FSIM_{norm} + \omega_3 * PSNR_{norm} + \omega_4 * (1 - MSE_{norm}) + \omega_5 * MOS_{norm} + \omega_6 * Likert_{norm} + \omega_7 * \tau_{norm}$$

де $\omega_1 ... \omega_7$ — вагові коефіцієнти, що визначають вклад кожної метрики у фінальну оцінку. Значення ваг можна підібрати емпірично або за допомогою методів оптимізації, таких як градієнтний спуск або регресійний аналіз, з урахуванням кореляції кожної метрики із загальним враженням експертів.

Нормалізація метрик здійснюється, наприклад, за формулою:

$$M_{norm} = \frac{M - M_{min}}{M_{max} - M_{min}} \quad (2.11)$$

де M – значення метрики, M_{min} і M_{max} – відповідно мінімальне і максимальне значення метрики у вибірці.

Гібридний метод має низку переваг:

– поєднує об'єктивність автоматизованих метрик з точністю людського сприйняття;

– підвищує надійність оцінки завдяки врахуванню кількох джерел інформації;

– забезпечує гнучкість, дозволяючи змінювати ваги в залежності від контексту застосування (медична візуалізація, відеонагляд, тощо);

– зменшує вплив випадкових похибок або суб’єктивної необ’єктивності окремих експертів.

Результатом розрахунку є агреговане значення Qh_{ybrid}, що дозволяє ранжувати зображення за якістю або порівнювати ефективність різних алгоритмів обробки зображень.

Таким чином, гібридний метод поєднує кращі риси суб’єктивних і об’єктивних оцінок і може бути використаний як науково обґрунтований інструмент в задачах автоматизованого аналізу якості зображень.

2.3 Висновки

У другому розділі було представлено комплексний підхід до вдосконалення процесу оцінювання якості зображень шляхом розробки двох нових методів: адаптивного та гібридного оцінювання. Проаналізовано обмеження існуючих об’єктивних метрик і обґрунтовано необхідність їх комбінування для отримання оцінок, більш узгоджених із суб’єктивним сприйняттям людини.

Розроблений метод AEMC дозволяє динамічно адаптувати ваги різних метрик залежно від типу спотворень зображення. Це забезпечує гнучкість, підвищену точність і більшу відповідність оцінок людському зору. Метод включає попередній аналіз зображення, визначення вагових коефіцієнтів, нормалізацію метрик та розрахунок зваженої суми.

Також у розділі запропоновано гібридний підхід, який поєднує об'єктивні метрики з суб'єктивними оцінками, такий підхід дозволяє покращити точність автоматизованих оцінок навіть у випадках складних або неочевидних спотворень.

Загалом, результати демонструють перспективність використання адаптивних і гібридних методів для об’єктивного оцінювання якості зображень, що дозволяє підвищити ефективність систем автоматичного аналізу зображень і зробити їх оцінки ближчими до реального сприйняття людини.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ ОЦІНЮВАННЯ ЯКОСТІ ГРАФІЧНИХ ЗОБРАЖЕНЬ

3.1 Розробка алгоритмів оцінки якості графічних зображень

Розроблений програмний застосунок для порівняння та аналізу зображень використовує сучасні вебтехнології [16] для забезпечення ефективної та зручної роботи користувача. Головним середовищем розробки обрано Visual Studio Code [17], що забезпечує зручну організацію проєкту та інтеграцію інструментів для розробки та налагодження. Клієнтська частина застосунку побудована на основі фреймворку React [18, 19, 20], що гарантує високу продуктивність рендерингу компонентів, їх повторне використання та динамічне оновлення інтерфейсу. Управління станом застосунку здійснюється за допомогою вбудованих механізмів React, що забезпечує узгодженість даних під час обробки зображень.

Загальний алгоритм роботи програми починається із завантаження користувачем двох графічних зображень через інтерактивний інтерфейс (див. рисунок 3.1). Користувач може вибрати файли за допомогою стандартного діалогового вікна або перетягнути їх у відповідні області інтерфейсу. Після завантаження кожного зображення програма перевіряє його формат на відповідність підтримуваним типам (JPEG, PNG, WEBP, BMP). У разі непідтримуваного формату, система відображає повідомлення про помилку. Для коректно завантажених зображень програма проводить попередню обробку, включаючи отримання метаданих (назва, розмір, тип) та аналіз основних характеристик, таких як ширина, висота, а також обчислення яскравості, контрасту та кількості мегапікселів.

Наступним етапом є аналіз якості кожного завантаженого зображення окремо. Програма обчислює числові значення контрасту, яскравості та мегапікселів, а також визначає розмір файлу. На основі цих значень, використовуючи заздалегідь визначені критерії, програма надає якісну оцінку кожного параметра, яка візуалізується за допомогою кольорового кодування

(зелений для добрих значень, жовтий для середніх та червоний для низьких). Додатково, для кожного параметра надається текстове пояснення, що допомагає користувачеві інтерпретувати отримані значення. Також розраховується загальна оцінка якості зображення на основі співвідношення розміру файлу до кількості мегапікселів.

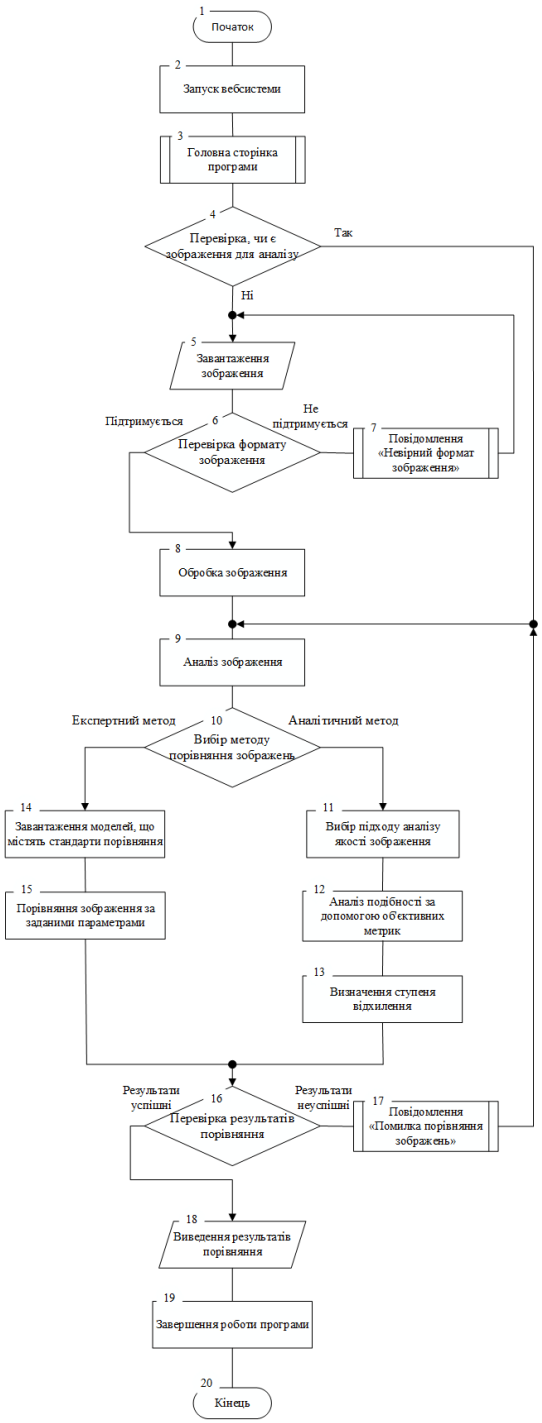


Рисунок 3.1 – Алгоритм роботи програми

У випадку, якщо користувач завантажує два зображення, програма автоматично переходить до етапу їх порівняння. Для цього застосовуються аналітичні методи оцінки подібності та відмінностей між зображеннями. Обчислюються такі об’єктивні метрики, як середньоквадратична похибка, яка відображає середню різницю між пікселями, пікове відношення сигнал/шум, що характеризує рівень спотворень, та індекс структурної подібності, який оцінює подібність структурних елементів зображень. Математичні формули для розрахунку цих метрик відображаються за допомогою LaTeX для забезпечення їх точного та зрозумілого представлення. Отримані значення MSE, PSNR та SSIM відображаються користувачеві разом з їх інтерпретацією, що дозволяє оцінити ступінь схожості та якість одного зображення відносно іншого.

Окрім об'єктивних способів оцінки зображень, програмне забезпечення було спроектовано з урахуванням можливості суб'єктивного аналізу. Ці методи базуються на особистому сприйнятті якості зображення безпосередньо людиною-експертом чи групою опитуваних. Для реалізації суб'єктивної оцінки було впроваджено декілька технік, зокрема метод середньої оцінки думок – MOS, метод рангової кореляції Кендалла та шкала Лайкерта.

Застосування методу MOS передбачало залучення групи фахівців, яким демонструвалися досліджувані зображення. Кожен експерт оцінював якість кожного зображення за визначеною числовою шкалою, наприклад, від 1 (найгірша якість) до 5 (найкраща якість). Після того, як усі експерти надали свої оцінки, для кожного зображення розраховувалося середнє арифметичне всіх поставлених балів. Це середнє значення й складало оцінку MOS для конкретного зображення, віддзеркалюючи узагальнену думку групи експертів щодо його якості.

Метод рангової кореляції Кендалла використовувався для оцінки співпадіння думок різних експертів. У цьому випадку кожен експерт не призначав абсолютну оцінку якості, а ранжував набір досліджуваних зображень від найкращого до найгіршого. Після отримання рангів від усіх експертів розраховувався коефіцієнт рангової кореляції Кендалла між парами експертів

або між кожним експертом та усередненим рангом. Високий коефіцієнт кореляції свідчив про велику узгодженість думок експертів щодо відносного порядку якості зображень.

Для отримання більш деталізованої інформації про сприйняття якості зображень також застосовувалася шкала Лайкерта. У цьому випадку експертам або респондентам пропонувалося оцінити різноманітні аспекти якості зображення (наприклад, чіткість, колірність, наявність артефактів) за допомогою шкали з кількома варіантами відповідей, які демонстрували ступінь згоди або незгоди з конкретним твердженням (наприклад, від «цілком не згоден» до «повністю згоден»). Аналіз відповідей за шкалою Лайкерта дозволяв зрозуміти, які саме аспекти якості є найбільш важливими для оцінювачів та як різні зображення сприймаються за цими критеріями.

Всі результати, отримані за допомогою суб'єктивних методів (значення MOS, коефіцієнти кореляції Кендалла, розподіл відповідей за шкалою Лайкерта), відображалися у звіті разом із результатами об'єктивного аналізу. Це надавало можливість користувачеві отримати комплексну оцінку якості зображень, поєднуючи кількісні метрики з якісними судженнями експертів.

Фінальним етапом роботи програми є відображення отриманих результатів аналізу та порівняння. Усі обчислені параметри, якісні оцінки та значення метрик порівняння відображаються в структурованому вигляді. Пояснення до параметрів та інтерпретація значень метрик допомагають користувачеві зрозуміти результати аналізу.

Для кращого розуміння роботи модулів програмних засобів оцінки якості графічних зображень було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (див. рисунок 3.2)

Рисунок 3.2 – Діаграма діяльності системи

Важливо зазначити, що програма надає можливість використовувати весь свій функціонал без попередньої авторизації. Однак, у такому випадку, користувач не зможе зберегти отримані результати оцінки. Для реалізації функції збереження результатів знадобиться розробка додаткової системи авторизації та механізмів зберігання даних користувача.

3.2 Розробка модуля визначення характеристик завантаженого зображення

Модуль для отримання інформації про зображення, представлений функцією «getImageInfo» (див. рисунки 3.3, 3.4), відповідає за асинхронне зчитування файлу зображення, аналіз його основних характеристик та повернення об'єкта «ImageInfo», що містить ці відомості.

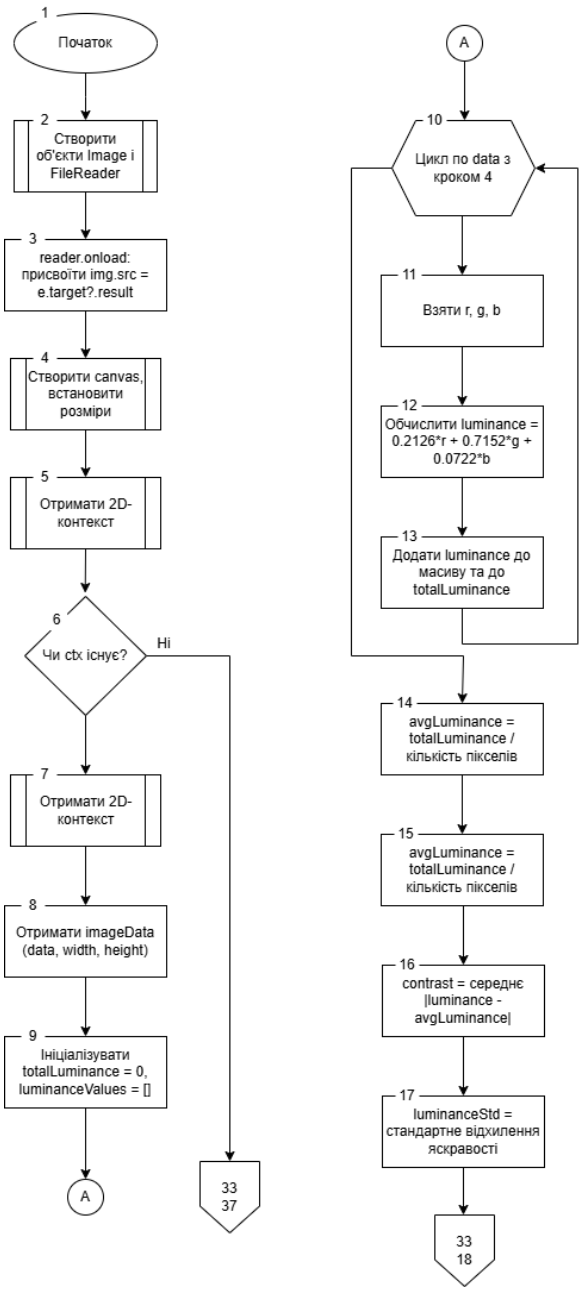


Рисунок 3.3 – Блок-схема алгоритму функції визначення характеристик зображення

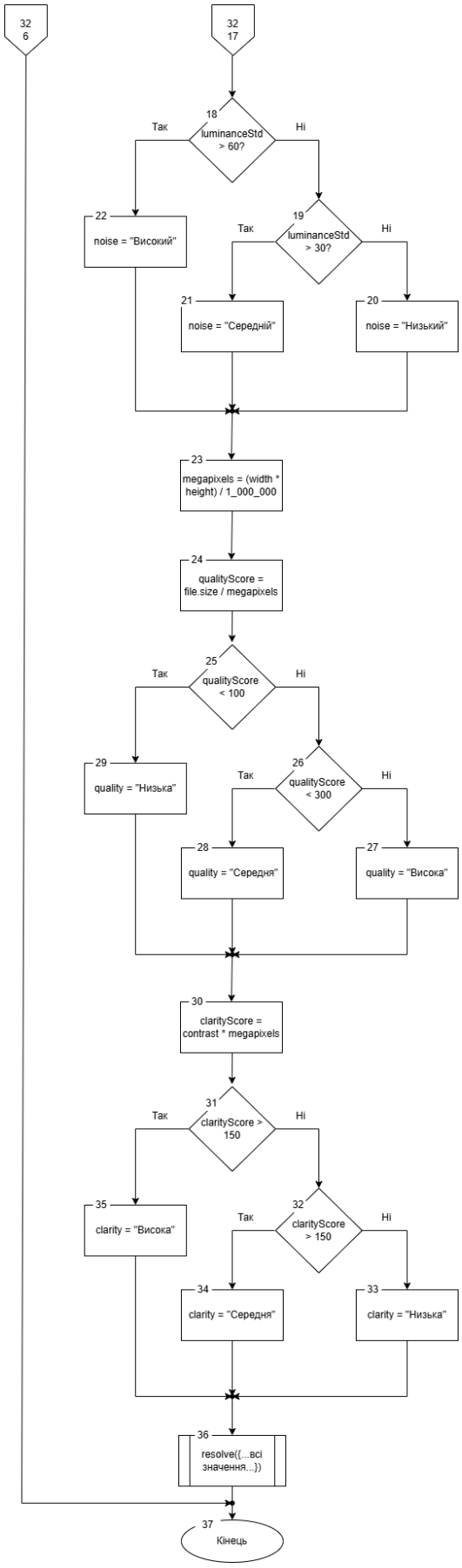


Рисунок 3.4 – Продовження блок-схеми алгоритму функції визначення характеристик зображення

Процес починається зі створення нового проміса (Promise), яка буде виконана (resolve) після успішного отримання та обробки інформації про зображення. Всередині промісу формуються два об'єкти: «img» типу «Image» для завантаження зображення та «reader» типу «FileReader» для зчитування вмісту файлу.

Функція «reader.onload» визначає обробник події завершення завантаження файлу. Після успішного зчитування вмісту файлу у вигляді Data URL, цей URL присвоюється властивості «src» об'єкта «img», що запускає завантаження зображення в браузері.

Наступним кроком, функція «img.onload» визначає обробник події завантаження зображення. Після успішного завантаження зображення, створюється елемент «canvas» з розмірами, відповідними розмірам завантаженого зображення.

Отримується 2D контекст цього канвасу за допомогою «canvas.getContext('2d')». Якщо контекст не вдається отримати (наприклад, через помилку браузера), виконання функції негайно припиняється.

За допомогою методу «ctx.drawImage(img, 0, 0)» завантажене зображення малюється на канвасі, починаючи з верхнього лівого кута (координати 0, 0). Після цього витягуються піксельні дані зображення з канвасу за допомогою «ctx.getImageData(0, 0, canvas.width, canvas.height)», які зберігаються в об'єкті «imageData». З цього об'єкта видобувається масив «data», що містить RGBA значення кожного пікселя зображення у вигляді лінійного масиву.

Для розрахунку яскравості та контрасту зображення, спочатку ініціалізуються змінні «totalLuminance» та порожній масив «luminanceValues». Далі відбувається ітерація по масиву «data» з кроком 4 (оскільки кожен піксель містить чотири значення: червоний, зелений, синій, альфа-канал). Для кожного пікселя обчислюється значення яскравості (luminance) за допомогою формули, що враховує вагові коефіцієнти для кожного кольорового каналу. Обчислене значення яскравості додається до масиву «luminanceValues» та акумулюється в змінній «totalLuminance».

Після опрацювання всіх пікселів обчислюється середня яскравість («avgLuminance») шляхом ділення «totalLuminance» на загальну кількість пікселів (довжина «luminanceValues»). Контраст («contrast») оцінюється як середнє абсолютне відхилення яскравості кожного пікселя від середньої яскравості.

Для наближеної оцінки рівня шуму («noise»), обчислюється стандартне відхилення значень яскравості. На основі цього стандартного відхилення визначається якісна оцінка шуму ("Високий", "Середній" або "Низький").

Потім обчислюється кількість мегапікселів («megapixels») шляхом множення ширини на висоту зображення та ділення на мільйон. Оцінка якості («qualityRating») визначається на основі співвідношення розміру файлу в кілобайтах до кількості мегапікселів. Чим менше це співвідношення, тим вищою вважається якість.

Для оцінки чіткості («clarity»), обчислюється скорингове значення як добуток контрасту на кількість мегапікселів, на основі якого присвоюється якісна оцінка («Висока», «Середня» або «Низька»).

На завершення, проміс виконується з об'єктом «ImageInfo», що містить всі зібрані та обчислені дані про зображення, зокрема ім'я файлу, розмір, тип, контраст, яскравість, мегапікселі, оцінку якості, рівень шуму, чіткість, а також масив піксельних даних («data»), ширину («width») та висоту («height») зображення.

Функція «reader.readAsDataURL(file)» ініціює процес зчитування вмісту переданого файлу («file»). Після завершення цього процесу, буде активований обробник «reader.onload».

Таким чином, модуль «getImageInfo» забезпечує асинхронне отримання та аналіз основних візуальних характеристик зображення, що є важливим етапом для подальшої оцінки його якості.

3.3 Розробка модуля обчислення середньоквадратичної похибки та пікового відношення сигнал/шум

Модуль розрахунку середньоквадратичної помилки та пікового співвідношення сигнал/шум містить дві основні функції: «calculateMSE» та «calculatePSNR».

Функція «calculateMSE» (див. рисунок 3.5) отримує на вхід два необов'язкові об'єкти «ImageInfo» (допустимо «null»), що містять інформацію про порівнювані зображення.

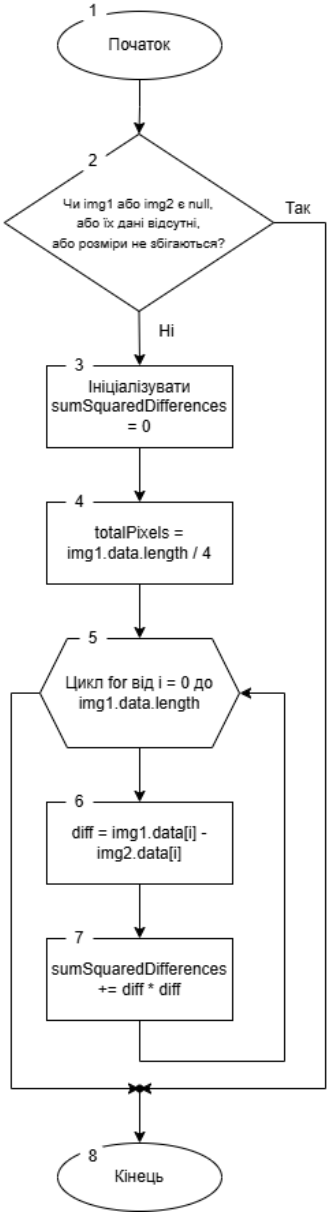


Рисунок 3.5 – Блок-схема алгоритму обчислення MSE

Спочатку відбувається перевірка, чи обидва об'єкти зображень існують («img1», «img2»), чи мають вони піксельні дані («data»), а також чи збігаються їх розміри (ширина «width» та висота «height»).

Якщо хоч одна з цих умов не виконується, функція повертає «null», бо обчислення MSE неможливе без правильних даних та ідентичних розмірів зображень.

Якщо перевірка пройдена успішно, ініціалізується змінна «sumSquaredDifferences» для нагромадження суми квадратів різниць між пікселями, що відповідають один одному, а також розраховується загальна кількість пікселів («totalPixels») в одному зображенні, шляхом ділення довжини масиву піксельних даних на 4 (адже кожен піксель представлений чотирма компонентами RGBA).

Наступним кроком є ітерація по масиву пікселів першого зображення («img1.data»). У кожній ітерації визначається різниця між значеннями кольорових каналів (червоний, зелений, синій, альфа) для відповідних пікселів обох зображень. Ця різниця підноситься до квадрату і додається до «sumSquaredDifferences».

Після завершення обробки всіх пікселів обчислюється MSE через ділення накопиченої «sumSquaredDifferences» на «totalPixels». Отриманий результат MSE повертається функцією.

Функція «calculatePSNR» (див. рисунок 3.6) приймає на вхід значення MSE (може бути «null») та необов'язкове максимальне значення пікселя («maxPixelValue», за замовчуванням 255 для 8-бітових зображень).

На початку перевіряється, чи MSE дорівнює «null» або нулю. У таких випадках функція повертає «null», оскільки PSNR не може бути розрахований (ділення на нуль або відсутність похибки).

В разі коректного значення MSE, PSNR обчислюється за формулою, де обчислене значення PSNR повертається функцією.

Обидві функції є детермінованими, тобто при однакових вхідних даних завжди повертають ідентичний результат. «calculateMSE» вимагає піксельних

даних обох зображень та їх відповідності за розміром, а «calculatePSNR» залежить від коректного значення MSE.

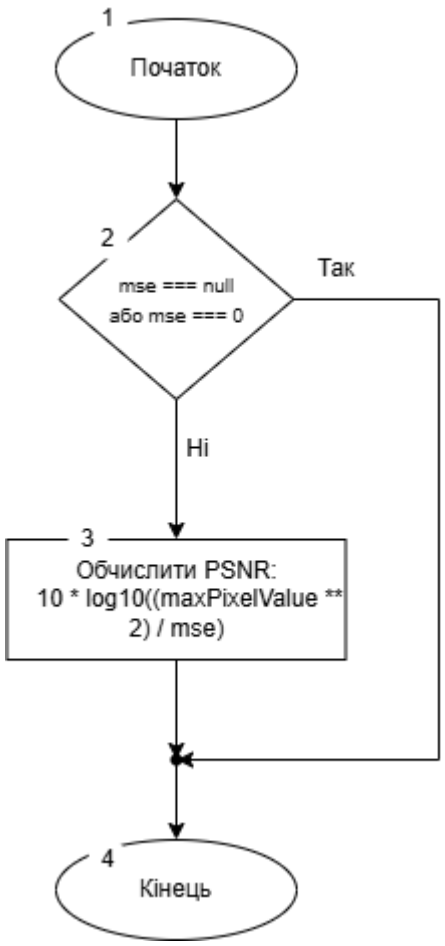


Рисунок 3.6 – Блок-схема алгоритму обчислення PSNR

У сукупності ці функції дають змогу кількісно оцінити різницю між двома зображеннями, використовуючи середньоквадратичну похибку та відношення сигнал/шум.

3.4 Розробка модуля обчислення структурної схожості

Модуль розрахунку структурної схожості містить функцію calculateSSIM.

Функція calculateSSIM отримує на вхід два необов'язкові об'єкти ImageInfo (допустимо null), що містять інформацію про порівнювані зображення. На початковому етапі відбувається ретельна перевірка наявності обох об'єктів

зображень (img1, img2), наявності у них піксельних даних (data), а також ідентичності їхніх розмірів (ширини width та висоти height).

Якщо хоча б одна з перевірок не проходить, функція негайно повертає null, сигналізуючи про неможливість обчислення SSIM через відсутність необхідних даних або розбіжність у розмірах зображень.

У випадку успішного проходження всіх перевірок, ініціюється процес обчислення структурної схожості. Спочатку визначається функція luminance, яка відповідає за розрахунок яскравості пікселя на основі значень його червоного, зеленого та синього каналів з використанням вагових коефіцієнтів, ця функція зображена на рисунку.

Далі обчислюється загальна кількість пікселів (pixels) в зображенні. Потім відбувається ітерація по піксельних даних обох зображень. У кожній ітерації для відповідних пікселів обчислюється значення яскравості за допомогою функції luminance, і ці значення додаються до масивів l1 та l2 відповідно.

Наступним кроком є розрахунок середніх значень яскравості (μ_1, μ_2), дисперсій яскравості (σ_1^2, σ_2^2) та коваріації яскравості (σ_{12}) для обох зображень.

Після цього визначаються дві константи (c_1, c_2) для стабілізації знаменника при малих значеннях середніх та дисперсій. Зазвичай вони обчислюються на основі динамічного діапазону пікселів (в даному випадку 255 для 8-бітових зображень).

На завершення обчислюється значення SSIM за формулою, що враховує середні значення, дисперсії та коваріацію яскравості обох зображень, а також стабілізуючі константи.

Отримане значення SSIM, що лежить в діапазоні від -1 до 1 (де 1 означає повну структурну схожість), повертається функцією.

Функція calculateSSIM є детермінованою та вимагає наявності піксельних даних обох зображень і їхньої повної відповідності за розмірами для коректного обчислення структурної схожості. Вона надає кількісну оцінку структурної подібності між двома зображеннями, що є важливим у багатьох задачах обробки зображень.

3.5 Розробка інтерфейсу програмного застосунку

При розробці користувацького інтерфейсу програмного забезпечення особливу увагу було приділено створенню інтуїтивно зрозумілого, візуально привабливого та технологічно сучасного середовища для користувача.

Для реалізації інтерактивних елементів та динамічної поведінки інтерфейсу було використано бібліотеку React, яка забезпечує ефективне оновлення компонентів та зручну обробку подій користувача. Стилізація елементів інтерфейсу здійснювалася за допомогою фреймворку Tailwind CSS [21], що дозволило швидко створювати адаптивний та візуально узгоджений дизайн завдяки використанню готових класів стилів.

Початковий етап взаємодії користувача із системою починається зі сторінки, що пропонує обрати спосіб ідентифікації (див. рисунок 3.7).

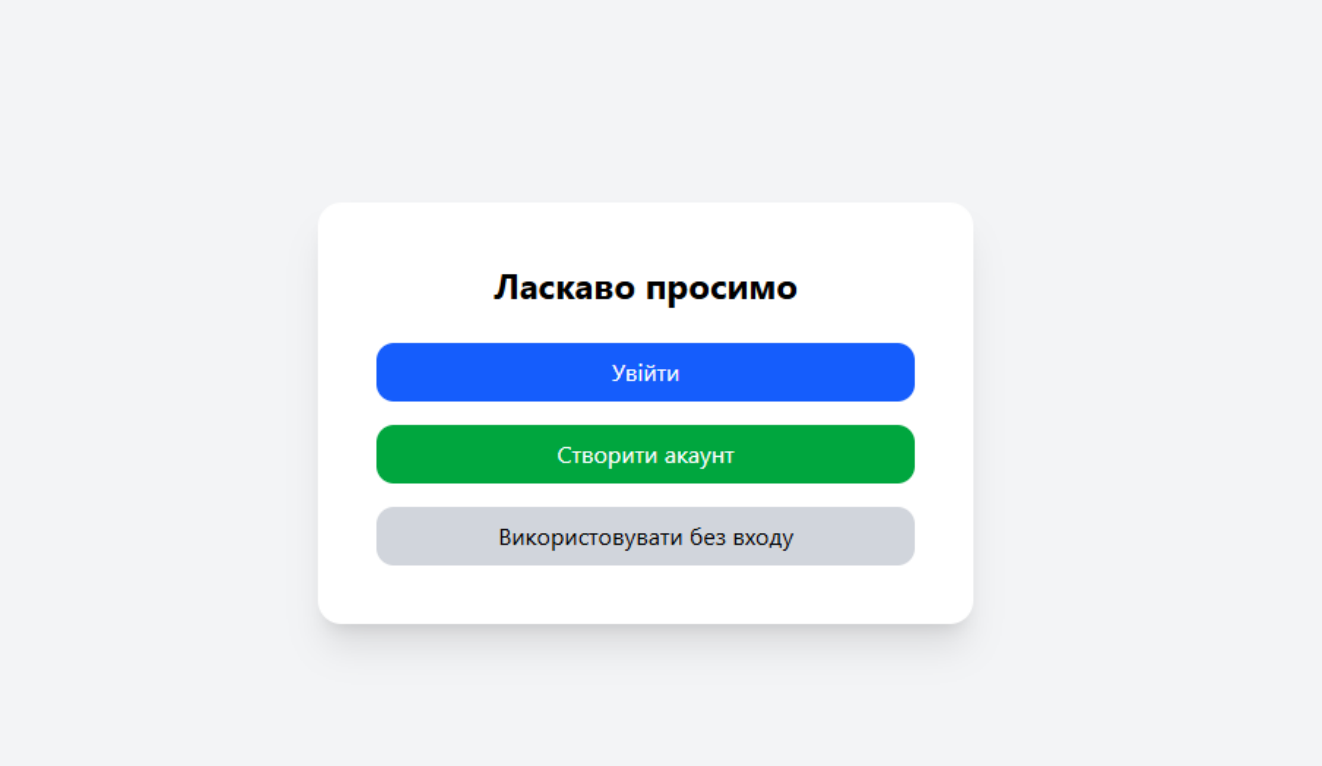


Рисунок 3.7 – Сторінка вибору способу авторизації

Дизайн цієї сторінки, створеної з використанням React та Tailwind CSS, виконано у світлих тонах, що створює відчуття відкритості та простоти. На

сторінці розміщено чітко відокремлені кнопки, що пропонують увійти до вже існуючого облікового запису або створити новий. Також передбачено можливість скористатися функціоналом програми без попередньої авторизації, що забезпечує швидкий доступ до основних можливостей.

Наступним кроком для зареєстрованих користувачів є сторінка входу або реєстрації, також розроблені з використанням React та стилізовані за допомогою Tailwind CSS. Сторінка входу виконана у спокійних синіх відтінках, що асоціюється з надійністю та стабільністю (див. рисунок 3.8). Форма для введення облікових даних розташована по центру, забезпечуючи зручне фокусування уваги.

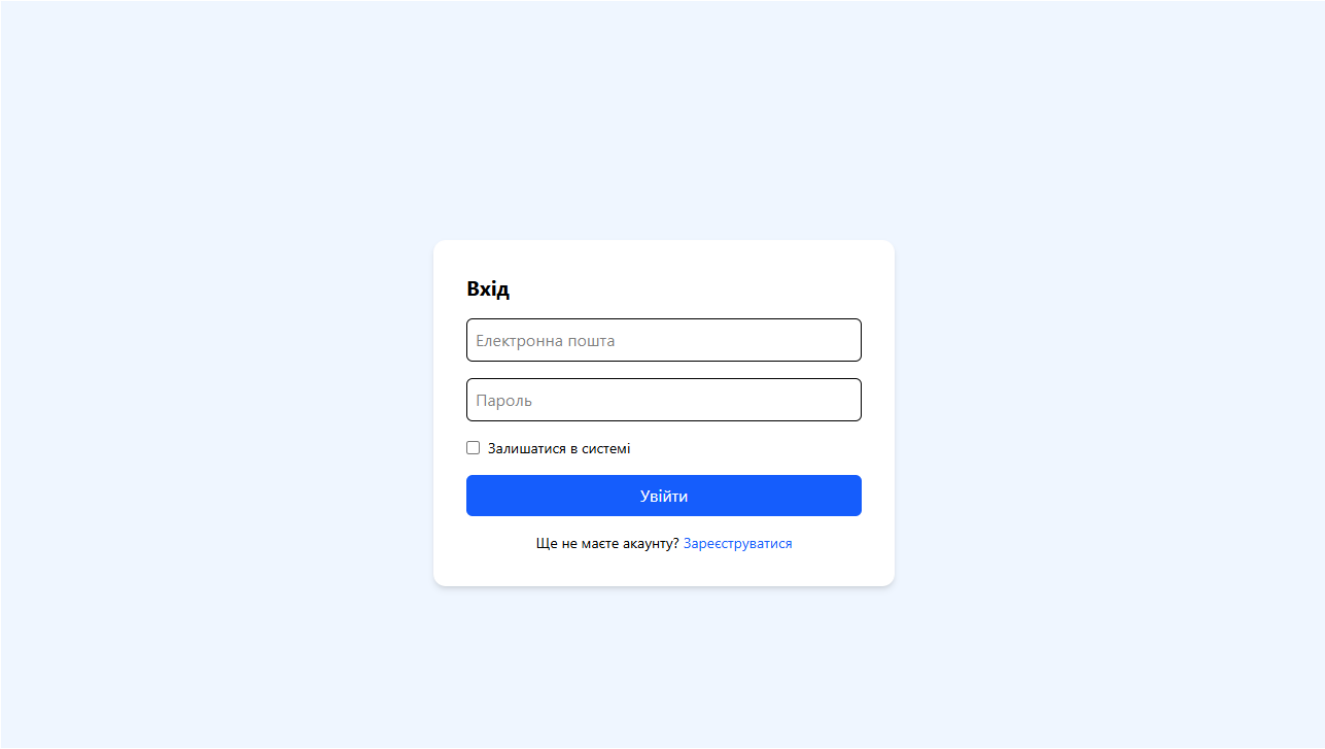


Рисунок 3.8 – Сторінка авторизації

Сторінка реєстрації, навпаки, використовує зелену кольорову гаму, що символізує створення нового облікового запису та позитивний початок роботи з програмою (див. рисунок 3.9). Обидві сторінки мають лаконічний дизайн з мінімальною кількістю елементів, що полегшує процес введення даних.

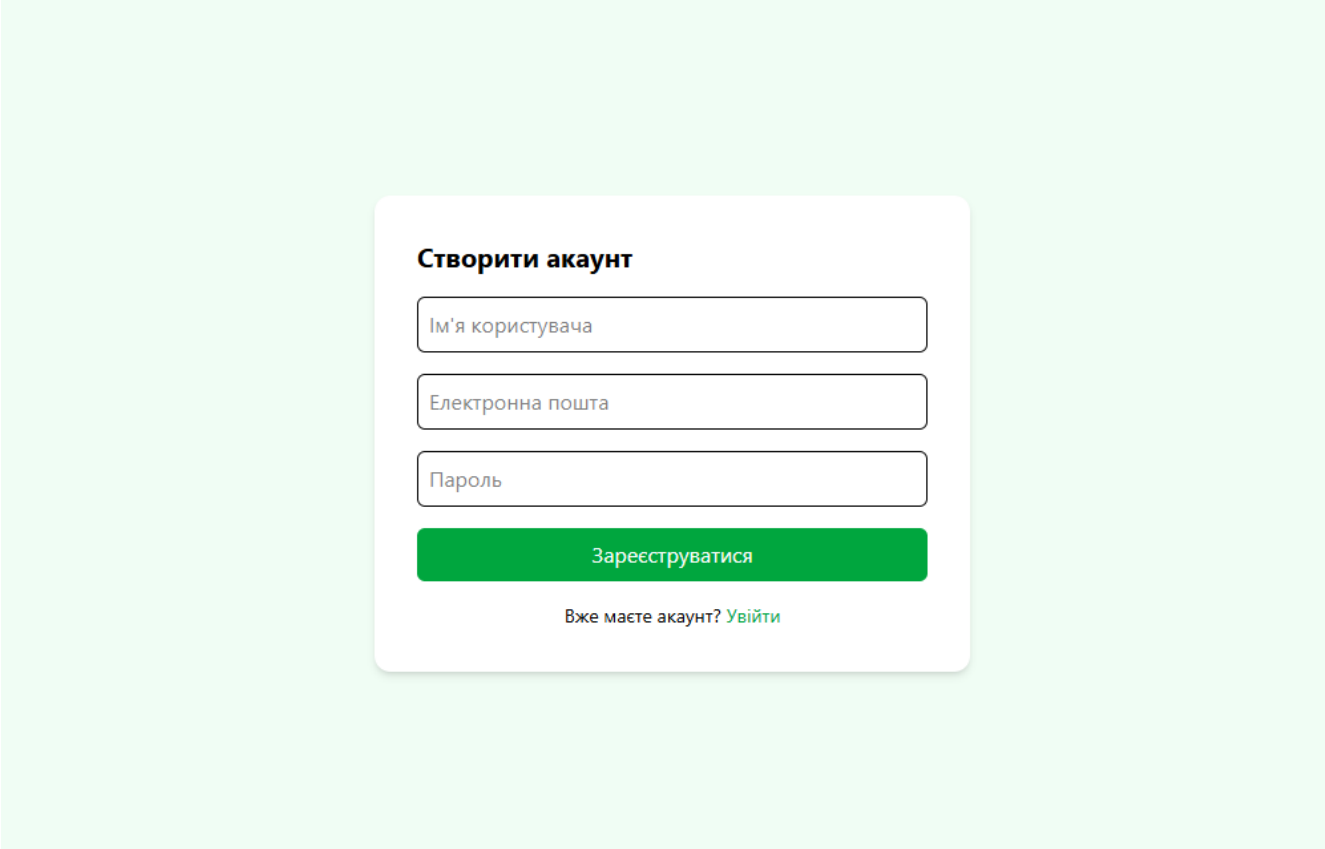


Рисунок 3.9 – Сторінка реєстрації

Основна робоча область програми, важливий елемент інтерфейсу, також побудована на основі React з використанням стилів Tailwind CSS. Вона має світлий та просторий дизайн, що сприяє комфортній роботі з функціоналом порівняння та аналізу зображень. Центральну частину екрана займають блоки для завантаження двох зображень, візуально відокремлені один від одного (див. рисунок 3.10). Кожен блок має чіткі індикатори для завантаження, реалізованого за допомогою стандартного елементу <input type="file"> з обробкою подій React, та відображення обраного файлу через елемент <img>. У процесі обробки даних, для відображення очікування, використовується анімація, реалізована засобами CSS.

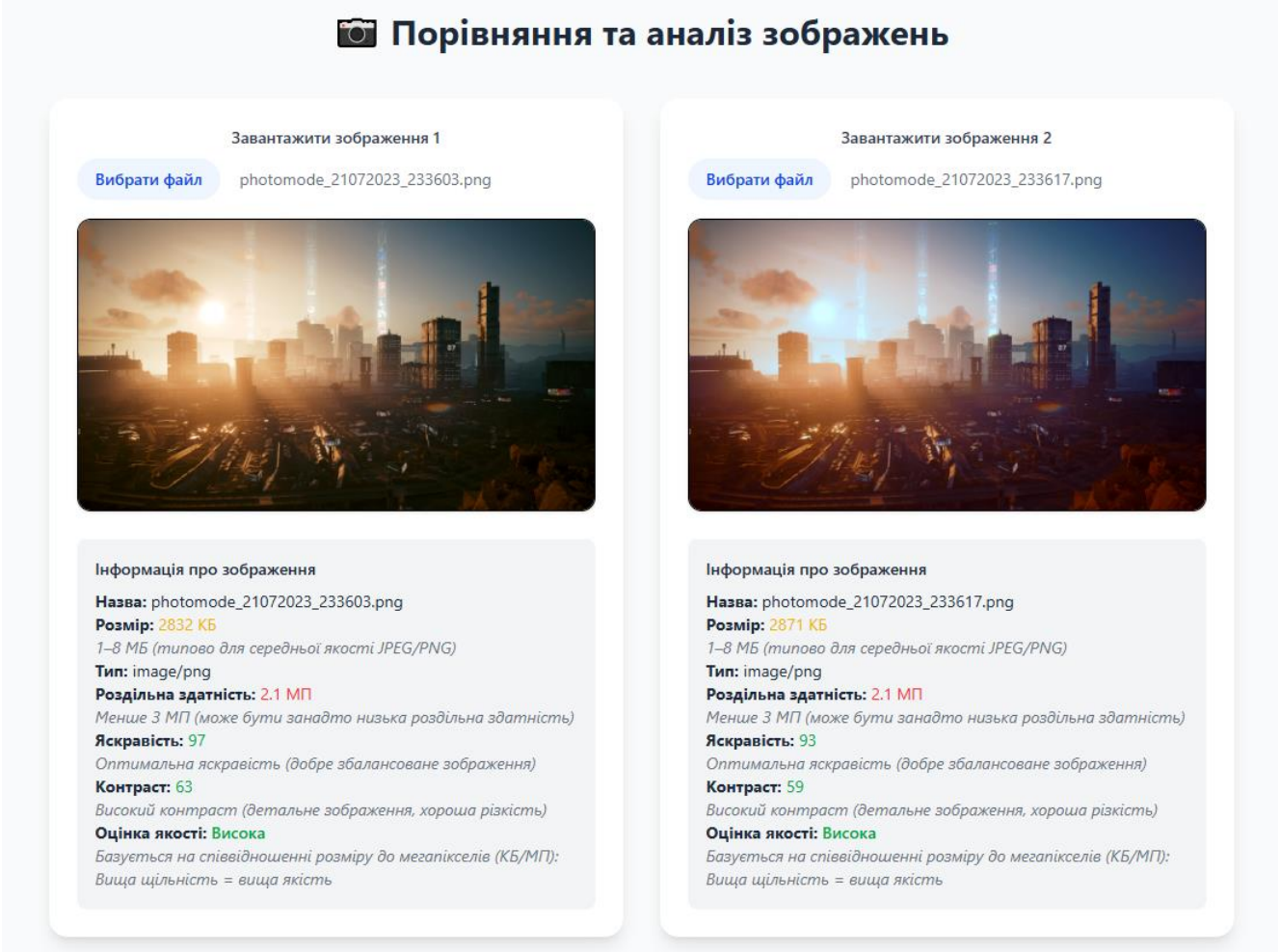


Рисунок 3.10 – Головна сторінка програмного застосунку

Після завершення аналізу, результати, включаючи математичні формули, відображаються у зрозумілому та структурованому вигляді (див. рисунок 3.11). Для коректного відображення математичних виразів використовувалася бібліотека KaTeX [22], що дозволяє форматувати формули у LaTeX-подібному синтаксисі та відображати їх безпосередньо у вебсторінці.

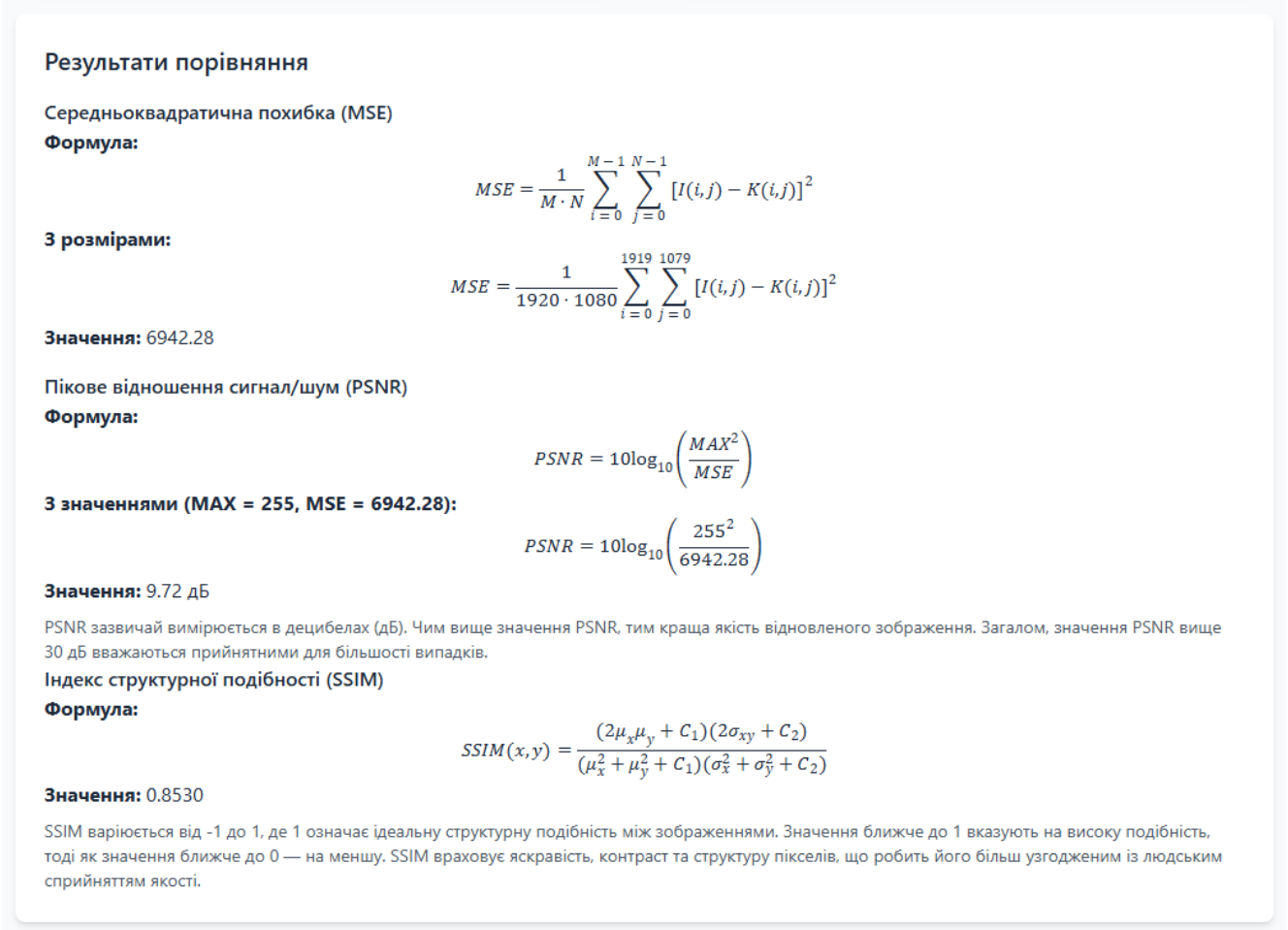


Рисунок 3.11 – Відображення результатів порівняння якості зображень

Таким чином, розроблений користувацький інтерфейс, завдяки використанню сучасних вебтехнологій, таких як React для інтерактивності та керування станом, Tailwind CSS для швидкої та адаптивної стилізації, та KaTeX для відображення математичних виразів, прагне бути інтуїтивно зрозумілим, візуально привабливим, технологічно сучасним та максимально зручним для виконання основних завдань програмного забезпечення.

3.5 Висновки

У третьому розділі описано повний цикл розробки програмних модулів, призначених для системи оцінювання якості графічних зображень, що отримала назву ImageRate. У відповідних підрозділах розглядаються теоретичні підґрунтя

та практичні аспекти впровадження функціоналу, потрібного для аналізу та порівняння зображень.

У сукупності, всі підрозділи розділу 3 спрямовані на створення повноцінної, зручної та наочної системи для оцінки якості графічних зображень. Розроблені модулі забезпечують функціональність як базового рівня (завантаження зображення, перегляд характеристик), так і експертну оцінку якості, що здійснюється через формалізовані метрики. Це робить систему придатною для використання як у навчальних, так і у прикладних задачах аналізу цифрових зображень.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Тестування системи

Системне тестування [23] – це всеосяжна процедура перевірки програмного забезпечення, розробленого як цілісний організм. Його головна мета – підтвердити, що розроблений продукт відповідає зафіксованим вимогам – як функціональним, так і нефункціональним, а також визначити, наскільки він відповідає сподіванням кінцевих користувачів. На відміну від модульного тестування, що зосереджується на ізольованій перевірці окремих складових коду, системне тестування досліджує взаємодію між цими складовими, комунікаційні протоколи та загальну поведінку системи в різних робочих середовищах.

Стосовно програмної системи, призначеної для кількісного оцінювання якості візуальних даних, системне тестування охоплює увесь технологічний ланцюжок обробки інформації. Воно включає етапи запуску та завантаження графічних файлів різних форматів та розмірів, їх подальшу обробку згідно з визначеними алгоритмами, процедури порівняльного аналізу, кількісне визначення відповідних показників якості та візуалізацію отриманих результатів. Ключовим є підтвердження правильності використаних алгоритмів оцінки, точності проведених обчислень, адекватності представлення вихідних даних, а також стійкості системи до виникнення потенційних помилок, таких як некоректні вхідні дані або обмежені обчислювальні ресурси.

Крім того, системне тестування передбачає моделювання реальних сценаріїв експлуатації програмного забезпечення. Це може включати аналіз продуктивності системи в умовах високого навантаження, оцінку стабільності при тривалій безперервній роботі під значним навантаженням, ергономічну оцінку інтерфейсу користувача (за його наявності) та аудит безпеки даних, що обробляються. Для забезпечення всебічної оцінки якісних характеристик програмної системи оцінювання зображень використовується різноманітний

набір технік тестування, що включає функціональне тестування (верифікація відповідності заявленим функціям), нефункціональне тестування (оцінка продуктивності, стійкості до навантажень, стресостійкості, безпеки та зручності використання) та регресійне тестування (контроль відсутності негативного впливу нових змін на існуючу функціональність). Успішне завершення циклу системного тестування є вкрай важливим етапом, який гарантує випуск надійного та ефективного програмного продукту, здатного задовольнити потреби цільової аудиторії.

Під час роботи оцінювання якості зображення система може перебувати в наступних станах:

1. Початковий стан – система очікує завантаження двох зображень для порівняння. Змінні images та imageInfos містять початкові значення [null, null], метрики mse та psnr мають значення null, прапорець isCalculating встановлено в false, а повідомлення про помилку error також має значення null. Інтерфейс відображає дві області для завантаження зображень (див. рисунок 4.1).

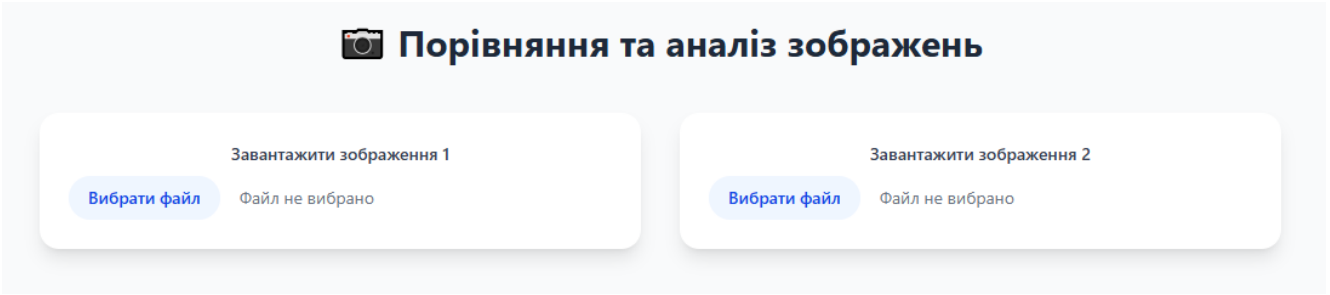


Рисунок 4.1 – Початковий стан системи

2. Завантаження першого зображення – користувач завантажує перше зображення. Відбувається оновлення стану images[0] (URL завантаженого зображення) та imageInfos[0] (інформація про зображення, включаючи дані, ширину та висоту). Перевіряється тип файлу. Якщо тип не підтримується, система переходить у стан Помилка завантаження. В іншому випадку, error залишається null (див. рисунок 4.2).

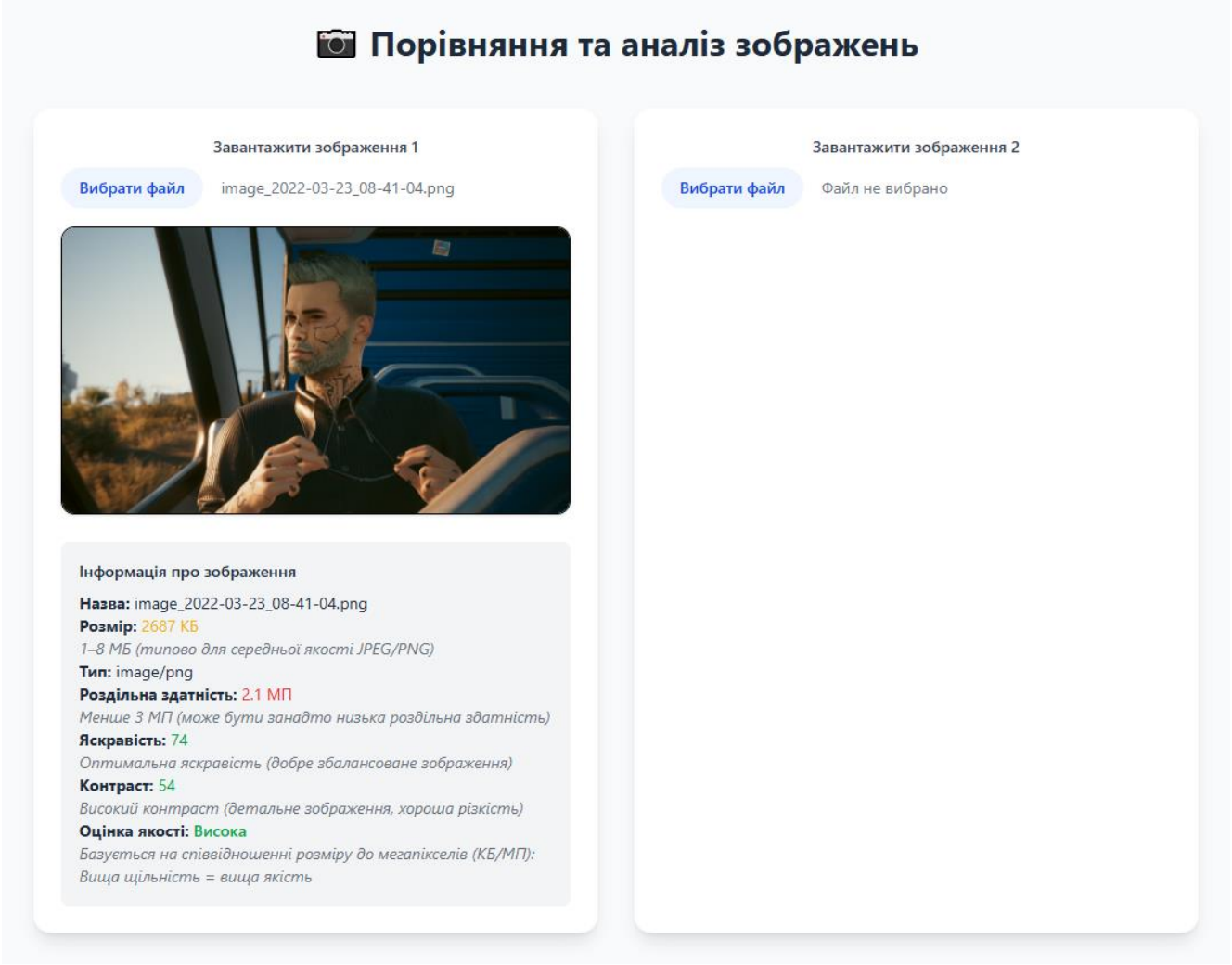


Рисунок 4.2 – Система з завантаженим лише одним зображенням

3. Завантаження другого зображення – користувач завантажує друге зображення. Аналогічно до попереднього кроку, оновлюються images[1] та imageInfos[1], проводиться перевірка типу файлу (див. рисунок 4.3). У разі помилки типу файлу, система переходить у стан Помилка завантаження.

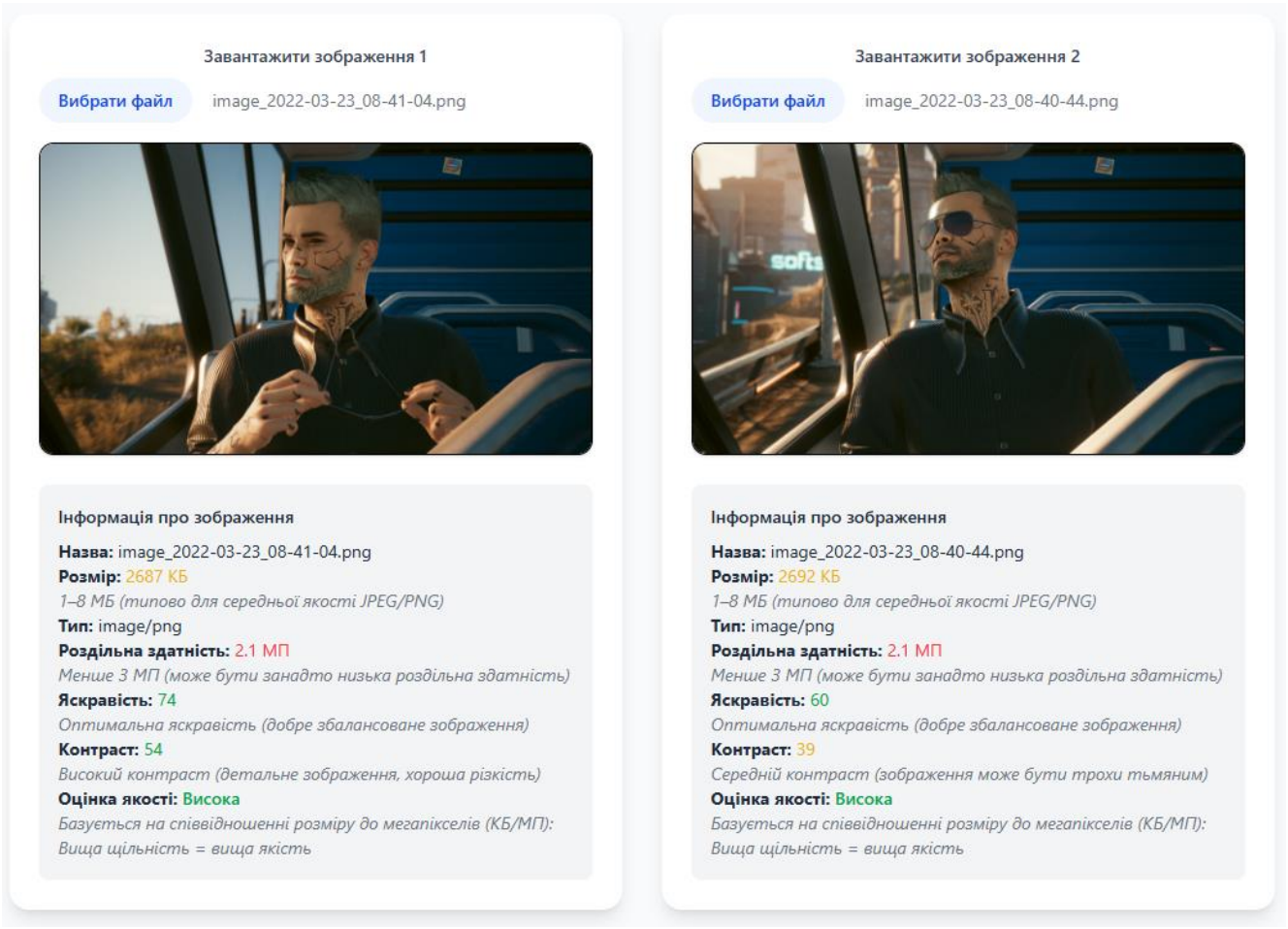


Рисунок 4.3 – Система з двома завантаженими зображеннями

4. Очікування розрахунку – після успішного завантаження обох зображень, якщо їхні розміри (ширина та висота) співпадають і дані обох зображень доступні, система автоматично переходить у стан Розрахунок. Прапорець isCalculating встановлюється в true, а значення mse та psnr скидаються до null. Відображається індикатор процесу розрахунку (див. рисунок 4.4).

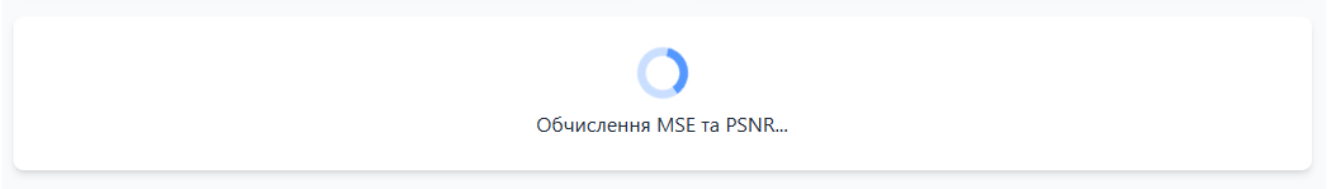


Рисунок 4.4 – Індикатор процесу розрахунку

5. Розрахунок – виконується асинхронний розрахунок середньоквадратичної похибки (MSE) між даними пікселів двох зображень за допомогою функції calculateMSE. Після завершення розрахунку, отримане значення MSE зберігається у стані mse. Далі, на основі розрахованого MSE, обчислюється пікове відношення сигнал/шум (PSNR) за допомогою функції calculatePSNR, і результат зберігається у стані psnr. Після завершення обох розрахунків, прапорець isCalculating знову встановлюється в false. Система переходить у стан Відображення результатів.

6. Відображення результатів – після успішного розрахунку MSE, PSNR та SSIM, їхні значення відображаються на екрані разом з відповідними формулами у форматі LaTeX. Також відображається інформаційне повідомлення про інтерпретацію значення PSNR та SSIM (див. рисунок 4.5).

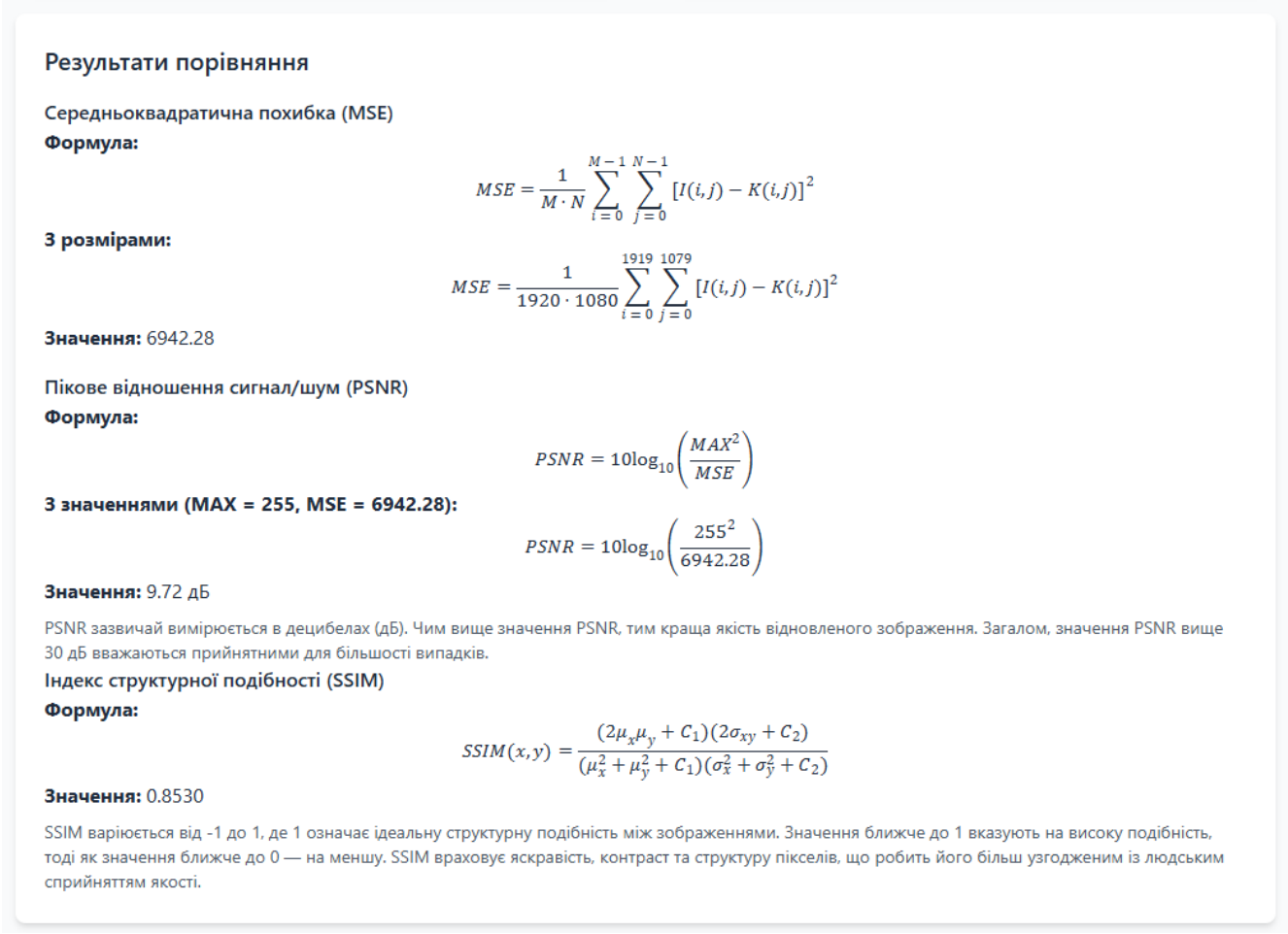


Рисунок 4.5 – Відображення результатів порівняння

7. Помилка завантаження – цей стан виникає, якщо користувач намагається завантажити файл непідтримуваного типу. У цьому стані змінна error містить повідомлення про помилку, яке відображається користувачеві. Стани images та imageInfos можуть містити URL та інформацію про вже завантажені (можливо, некоректні) зображення. Розрахунок MSE та PSNR не відбувається.

8. Некоректні вхідні дані – цей стан є станом компонента, який описує ситуацію, коли завантажено два зображення, але їхні розміри не співпадають або дані одного з зображень відсутні. У цьому випадку, функція calculate не виконує розрахунок, а значення mse та psnr встановлюються в null. Система відображає повідомлення про невідповідні (див. рисунок 4.6) дані та повертається до стану очікування нових (коректних) зображень.

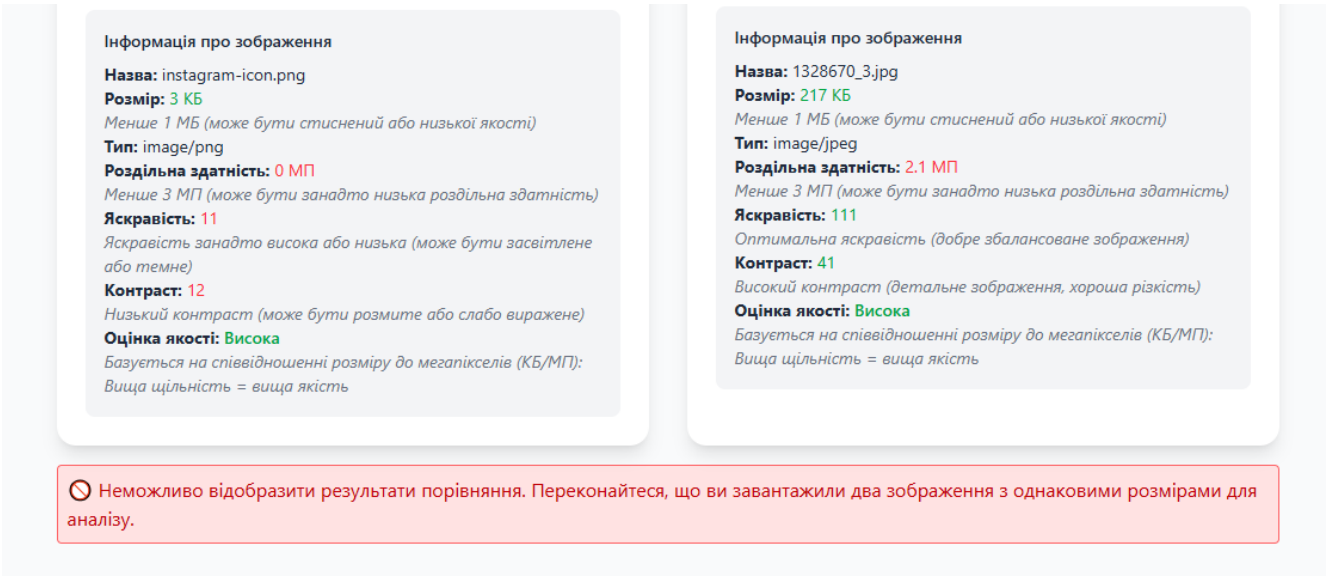


Рисунок 4.6 – Попередження про завантаження файлів з різними розмірами

Під час роботи програмного застосунку можуть виникати наступні помилки:

1. Непідтримуваний тип файлу – користувач намагається завантажити файл, тип якого не включено до списку SUPPORTED_IMAGE_TYPES. У цьому випадку, встановлюється стан error з відповідним повідомленням, наприклад, «Непідтримуваний тип файлу: image/svg+xml». Це попереджає користувача про

необхідність завантажити зображення у підтримуваному форматі (наприклад, JPEG, PNG) (див. рисунок 4.7).

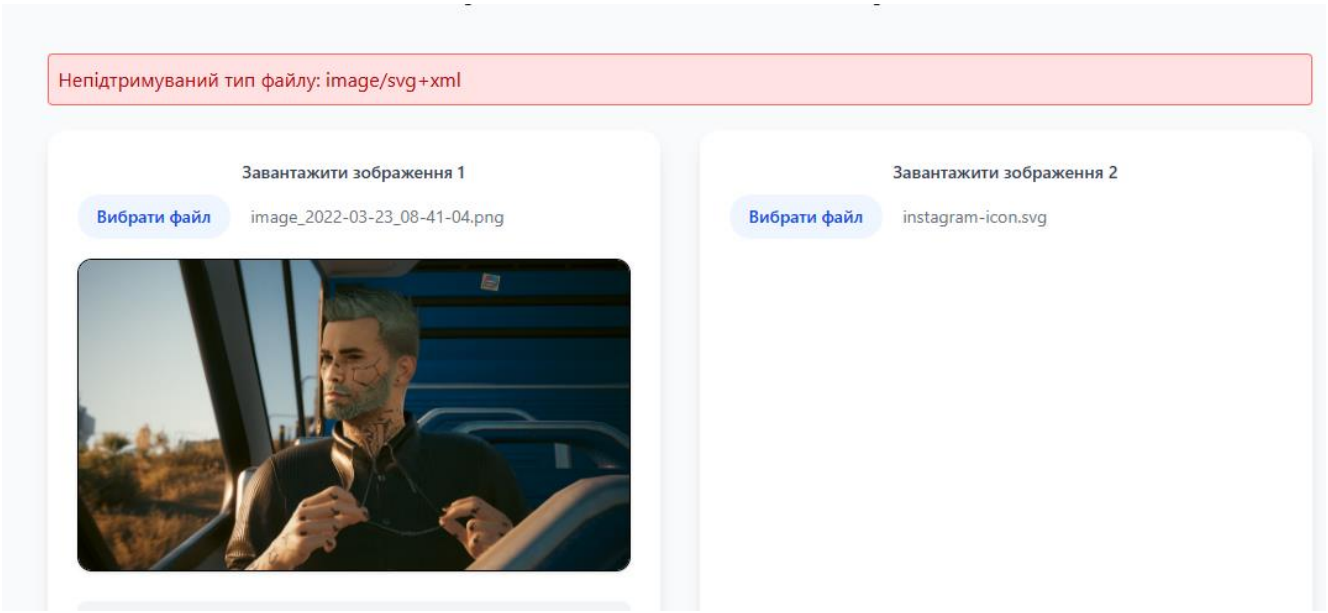


Рисунок 4.7 – Помилка при завантаженні другого зображення у невідповідному форматі

2. Помилка рендерингу LaTeX – під час відображення формул MSE та PSNR може виникнути помилка при спробі рендерингу LaTeX за допомогою бібліотеки katex. У цьому випадку, помилка реєструється в консолі, але для користувача відображається оригінальний LaTeX вираз замість відрендереної формули. Це є скоріше попередженням про проблему з відображенням, ніж критичною помилкою функціональності.

Хоча явних інформаційних індикаторів у коді не передбачено, можна виділити наступні ситуації, які можна розглядати як індикатори для користувача:

1. Різні розміри зображень – якщо користувач завантажує два зображення з різними значеннями ширини або висоти, розрахунок MSE та PSNR не виконується. Хоча це не є помилкою в обробці (система працює відповідно до логіки), це є важливою інформацією для користувача, оскільки порівняння зображень різних розмірів є некоректним. У поточному коді ця ситуація призводить до скидання mse та psnr до null без явного повідомлення

користувачеві. Доцільно було б додати візуальне попередження про необхідність завантаження зображень однакового розміру.

2. Відсутність одного з зображень – якщо завантажено лише одне зображення, розрахунок також не виконується. Знову ж таки, це не є помилкою, але користувач повинен розуміти, що для порівняння необхідно завантажити два зображення.

Таким чином, підрозділ оцінювання якості зображення проходить через різні стани в залежності від дій користувача та результатів обробки. Система обробляє помилки, пов'язані з непідтримуваними типами файлів, та потенційні проблеми з відображенням формул. Для покращення досвіду користувача, доцільно було б додати більш явні попередження про некоректні вхідні дані, такі як зображення різних розмірів або відсутність одного з необхідних зображень.

4.2 Розробка інструкції користувача

Дана інструкція описує процес використання підсистеми оцінювання якості зображення, включаючи технічні вимоги для її запуску та покрокову інструкцію з використання основного функціоналу.

Для коректної роботи підсистеми оцінювання якості зображення необхідно забезпечити відповідність апаратного та програмного забезпечення комп'ютера наступним вимогам (див. таблицю 4.1):

Таблиця 4.1 – Вимоги до апаратного та програмного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
1	2	3
Процесор	Intel Core i5 або аналогічний	Intel Core i3 або аналогічний
Оперативна пам'ять	8 ГБ RAM	4 ГБ RAM

Продовження таблиці 4.1

1	2	3
Вільний простір на диску	500 МБ	100 МБ
Відеокарта	Будь-яка сучасна відеокарта з підтримкою WebGL (апаратне прискорення)	Будь-яка відеокарта, сумісна з сучасними браузерами
Операційна система	Windows 10 або новіша, macOS Catalina або новіша, сучасні дистрибутиви Linux	Windows 7 або новіша, macOS Mojave або новіша, сучасні дистрибутиви Linux
Браузер	Google Chrome (останньої версії), Mozilla Firefox (останньої версії), Safari (останньої версії), Microsoft Edge (останньої версії) з увімкненою підтримкою JavaScript. Рекомендується використовувати останні версії браузерів для забезпечення максимальної продуктивності.	Google Chrome (версія 70 або новіша), Mozilla Firefox (версія 60 або новіша), Safari (версія 12 або новіша), Microsoft Edge (версія 79 або новіша) з увімкненою підтримкою JavaScript.
Мережа Інтернет	Не потрібна для локального використання після завантаження сторінки.	Не потрібна для локального використання після завантаження сторінки.
Роздільна здатність екрана	HD (1920x1080) або вище	HD (1280x720)

Продовження таблиці 4.1

1	2	3
Додаткове обладнання	Миша, клавіатура	Миша або тачпад, клавіатура

Після успішного завантаження вебсторінки з програмною системою оцінювання якості зображення, користувач потрапляє на головний екран, де представлені дві області для завантаження зображень.

Крок 1. Завантаження зображень:

1. У першій області з підписом «Завантажити зображення 1» натисніть кнопку «Обрати файл» або відповідний елемент інтерфейсу для завантаження першого зображення, яке буде вважатися еталонним або оригінальним.

2. У діалоговому вікні виберіть необхідний файл зображення з підтримуваним форматом (наприклад, JPEG, PNG) та натисніть «Відкрити». Після успішного завантаження, в області відобразиться попередній перегляд зображення та інформація про нього (за наявності).

3. Аналогічно, у другій області з підписом «Завантажити зображення 2» завантажте друге зображення, якість якого необхідно оцінити шляхом порівняння з першим.

Переконайтеся, що обидва завантажені зображення мають однакові розміри (ширину та висоту) для коректного розрахунку метрик якості. У випадку завантаження зображень різних розмірів, розрахунок MSE та PSNR не буде виконано.

Повідомлення про помилку – якщо при завантаженні файлу з'являється повідомлення «Непідтримуваний тип файлу: [тип файлу]», це означає, що завантажений файл має формат, який не підтримується системою. Будь ласка, завантажте зображення у підтримуваному форматі.

Крок 2. Очікування та перегляд результатів:

1. Після успішного завантаження обох зображень однакового розміру, система автоматично розпочне процес розрахунку середньоквадратичної похибки, пікового відношення сигнал/шум та індексу структурної подібності.

2. Під час розрахунку на екрані може відображатися індикатор завантаження або повідомлення «Обчислення MSE та PSNR...».

3. Після завершення розрахунку, в нижній частині екрана з'явиться блок «Результати порівняння», який містить наступну інформацію:

- середньоквадратична похибка – відображається формула розрахунку MSE у форматі LaTeX, формула з підставленими розмірами зображень (за наявності інформації про розміри) та обчислене значення MSE з точністю до двох знаків після коми;
- пікове відношення сигнал/шум – відображається формула розрахунку PSNR у форматі LaTeX, формула з підставленим значенням MAX (255 для 8-бітових зображень) та розрахованим значенням MSE, а також кінцеве значення PSNR у децибелах (дБ) з точністю до двох знаків після коми. Також надається інформаційне повідомлення про загальну інтерпретацію значення PSNR.

Інтерпретація результатів:

- Mean Squared Error – чим нижче значення MSE, тим менша різниця між двома зображеннями, і отже, тим вища їх схожість. Значення MSE = 0 вказує на повну ідентичність зображень;
- Peak Signal-to-Noise Ratio – PSNR вимірюється в децибелах (дБ). Чим вище значення PSNR, тим краща якість відновленого або стиснутого зображення порівняно з оригінальним. Зазвичай, значення PSNR вище 30 дБ вважаються прийнятними для більшості випадків.

Крок 3. Повторне порівняння або завантаження нових зображень.

Для порівняння інших пар зображень, користувач може повторно виконати Крок 1, завантаживши нові файли зображень. Результати попереднього порівняння будуть замінені новими після успішного розрахунку.

Дана інструкція охоплює основний функціонал підсистеми оцінювання якості зображення.

4.3 Висновки

У розділ детально описується процес тестування підсистеми оцінювання якості зображень. Було визначено основні стани системи, можливі помилки (непідтримуваний тип файлу, помилка рендерингу LaTeX) та потенційні попередження (різні розміри зображень, відсутність одного з зображень). Розроблена інструкція користувача надає чітке керівництво щодо технічних вимог та порядку роботи з системою, включаючи завантаження зображень та інтерпретацію отриманих результатів MSE, PSNR та SSIM. Проведене тестування та розроблена інструкція є важливими кроками для забезпечення надійності, зручності використання та розуміння принципів роботи підсистеми оцінювання якості зображень кінцевими користувачами.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено методи та програмні модулі для оцінювання якості графічних зображень, що поєднують аналітичні та експертні підходи.

Проведено аналіз існуючих методів оцінювання якості графічних зображень, виявлено їхні переваги та недоліки, а також обґрунтовано необхідність розробки комплексної системи, яка б враховувала як об'єктивні метрики, так і суб'єктивне сприйняття.

Вперше запропоновано адаптивний метод оцінювання якості зображень, особливість якого полягає в динамічному визначенні вагових коефіцієнтів класичних об'єктивних метрик на основі характеристик спотворень зображень, що забезпечує більш гнучку та узгоджену з людським сприйняттям оцінку.

Також вперше розроблено гібридний метод оцінювання якості зображень, який інтегрує адаптивне комбінування об'єктивних метрик із суб'єктивними оцінками експертів. Аналіз характеру спотворень зображення використовується для динамічного визначення вагових коефіцієнтів кожної метрики, що підвищує узгодженість об'єктивної оцінки з візуальним сприйняттям людини.

Практичне значення отриманих результатів полягає у розробці алгоритмів та програмної системи, що поєднує аналітичні та експертні методи оцінювання якості графічних зображень. Розроблена система може бути використана для підвищення об'єктивності та комплексності оцінки графічного контенту, оптимізації процесів його створення, порівняльного аналізу ефективності різних методів комп'ютерної графіки, формування єдиних стандартів якості та покращення взаємодії між розробниками та користувачами.

Проведене тестування розроблених програмних модулів на різних графічних зображеннях підтвердило їхню ефективність та коректність роботи. Результати дослідження можуть бути використані в різних сферах застосування комп'ютерної графіки, де важливим є забезпечення високої якості візуального контенту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рябоконь А.М., Романюк О.Н. Аналітичні методи оцінки якості зображень. Матеріали XII Всеукраїнської науково-практичної конференції молодих учених (2025), Київ, 15 травня 2025 р. – Видавництво Київський столичний університет імені Бориса Грінченка, 2025р. – С. 246-248.

2. . Рябоконь А.М., Романюк О.Н. Використання метрики PSNR для оцінки якості зображень. Матеріали LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025), Вінниця, 24-27 вересня 2025 р. – Вінниця.

3. Рябоконь А.М., Романюк О.Н. Методи математичного моделювання для об'єктивної та експертної оцінки якості цифрових зображень. Матеріали X Всеукраїнської науково-методичної конференції «Освіта, наука та виробництво: розвиток та перспективи» (2025), Суми, 24 квітня 2025 р. – Видавництво Сумський державний університет, 2025р. – С. 192-194.

4. Рябоконь А.М., Романюк О.Н. Оцінка якості зображень на основі аналітичних та експертних методів. Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів (2025), Одеса, 17-18 квітня 2025 р. – Видавництво ОНТУ, 2025р. – С. 151-153.

5. Рябоконь А.М., Романюк О.Н. Оцінювання якості зображень у комп’ютерному моделюванні технологічних процесів на основі показників psnr та mse. Матеріали Всеукраїнської науково-практичної Internet-конференції (2025), Черкаси, 17-21 березня 2025 р. – С. 109-110.

6. Рябоконь А.М., Романюк О.Н. Комбіноване використання суб’єктивних і об’єктивних метрик оцінювання якості цифрових зображень. Матеріали XXXIII Міжнародної науково-практичної конференції MicroCAD-2025 (2025), Харків, 14-17 травня 2025 р – Видавництво НТУ «Харківський політехнічний інститут», 2025р. – С. 1528.

7. Mean Opinion Score (MOS). [Електронний ресурс] – Режим доступу до ресурсу: https://developer.salesforce.com/docs/atlas.en-

us.voice_pt_developer_guide.meta/voice_pt_developer_guide/voice_pt_mos.html (дата звернення: 10.03.2025).

8. A. C. Jiji, Y. R. A. Bessant, S. Absa, S. M. S. Sujitha. Digital Image Processing: Practical Implementation With MATLAB. Booksclinic Publishing, 2023. – 198 с.

9. Peak Signal-to-Noise Ratio as an Image Quality Metric [Електронний ресурс] – Режим доступу до ресурсу: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html?srsltid=AfmBOorpLN7-UW37RPRbi6Z6hwjGaIAZEiTlu9Cz99PhEzIhHAfJLVOE. (дата звернення: 10.03.2025).

10. SSIM: Structural Similarity Index – Imatest [Електронний ресурс] URL: https://www.imatest.com/docs/ssim/ (дата звернення: 10.03.2025).

11. FSIM: A Feature Similarity Index for Image Quality Assessment [Електронний ресурс] – Режим доступу до ресурсу: https://signalprocessingsociety.org/publications-resources/blog/fsim-feature-similarity-index-image-quality-assessment (дата звернення: 10.03.2025).

12. V. V. Estrela. Intelligent Healthcare Systems. CRC Press, 2023. – 398с.

13. OpenCV - Open Computer Vision Library [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/ (дата звернення: 03.04.2025).

14. Naturalness Image Quality Evaluator [Електронний ресурс] – Режим доступу до ресурсу: https://quality.nfdi4ing.de/en/latest/image_quality/NIQE.html (дата звернення: 03.04.2025).

15. Vision AI: Image and visual AI tools [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/vision?hl=en (дата звернення: 03.04.2025).

16. 30 Best Free Tools for Frontend Developers in 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/rowsanali/30-best-free-tools-for-frontend-developers-in-2025-1gdm (дата звернення: 03.04.2025).

17. Visual Studio Code - Code Editing. Redefined [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення: 03.04.2025).

18. A. Banks, E. Porcello. Learning React: Modern Patterns for Developing React Apps 2nd Edition. Sebastopol: O'Reilly, 2020. – 307 с.

19. D. Flanagan. JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language 7th Edition. Sebastopol: O'Reilly, 2020. – 706 с.

20. A. Jones. Mastering TypeScript Programming: An In-Depth Exploration of Essential Concepts. Walzone Press, 2025. – 259 с.

21. Tailwind CSS – Rapidly build modern websites without ever leaving your HTML [Електронний ресурс] – Режим доступу до ресурсу: https://katex.org/ (дата звернення: 03.04.2025).

22. KaTeX – The fastest math typesetting library for the web [Електронний ресурс] – Режим доступу до ресурсу: https://katex.org/ (дата звернення: 03.04.2025).

23. G. Blokdyk. System Testing A Complete Guide - 2020 Edition. 5STARCooks, 2021. – 304 с.

ДОДАТКИ

61

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«14» 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Програмна система оцінювання якості графічних зображень»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
д.т.н., проф. Романюк О.Н.
«14» 03 2025 року.
Виконав:
студент гр. ЗПІ-21б Рябоконь А.М.
«14» 03 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Програмна система оцінювання якості графічних зображень».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №18 від 18 лютого 2025 р.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності оцінювання якості графічних зображень шляхом розробки програмних модулів, які поєднують експертні та аналітичні методи для автоматизованого аналізу і порівняння зображень.

4. Вихідні дані для проведення НДР

1. A. C. Jiji, Y. R. A. Bessant, S. Absa, S. M. S. Sujitha. Digital Image Processing: Practical Implementation With MATLAB. Booksclinic Publishing, 2023. – 198 c.

2. V. V. Estrela. Intelligent Healthcare Systems. CRC Press, 2023. – 398c.

3. A. Banks, E. Porcello. Learning React: Modern Patterns for Developing React Apps 2nd Edition. Sebastopol: O'Reilly, 2020. – 307 c.

4. D. Flanagan. JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language 7th Edition. Sebastopol: O'Reilly, 2020. – 706 c.

5. A. Jones. Mastering TypeScript Programming: An In-Depth Exploration of Essential Concepts. Walzone Press, 2025. – 259 c.

5. Технічні вимоги

Вихідні дані до роботи: роздільна здатність зображень – до 1920×1080 пікселів; кольоровий режим – RGB (TrueColor); формати вхідних зображень – .png, .jpg.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської кваліфікаційної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів і засобів для оцінювання якості графічних зображень з використанням об'єктивних та суб'єктивних метрик	25.03.25 – 31.03.25
2	Розробка методів оцінювання якості зображень з використанням об'єктивних і суб'єктивних метрик	03.04.25 – 17.04.25
3	Розробка програмних модулів системи оцінювання якості графічних зображень	19.04.25 – 03.05.25
4	Тестування програмного забезпечення	05.05.25 – 18.05.25
5	Оформлення матеріалів до захисту БКР	20.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

64

Додаток Б – Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмна система оцінювання якості графічних зображень.

Тип роботи: БКР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: д.т.н., проф. каф. ПЗ Романюк О.Н.

Оригінальність	96,5%
Схожість	3,5%

Особа, відповідальна за перевірку  Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою StrikePlagiarism

Автор роботи  Рябоконь А.М.

Керівник роботи Романюк О.Н.

Додаток В – Лістинг програми

```
// main-app.page.tsx
import katex from 'katex';
import React, { useState, useEffect } from 'react';

interface ImageInfo {
  name: string;
  size: number;
  type: string;
  contrast: number;
  brightness: number;
  megapixels: number;
  quality: string;
  noise: string;
  clarity: string;
  data?: Uint8ClampedArray;
  width?: number;
  height?: number;
}

const getQualityColor = (value: number, type: 'contrast' | 'brightness' | 'mp' | 'size' | 'noise' | 'clarity')
=> {
  switch (type) {
    case 'contrast':
      return value > 40 ? 'text-green-600' : value > 20 ? 'text-yellow-500' : 'text-red-500';
    case 'brightness':
      return value > 180 || value < 60 ? 'text-red-500' : 'text-green-600';
    case 'mp':
      return value > 8 ? 'text-green-600' : value > 3 ? 'text-yellow-500' : 'text-red-500';
    case 'size':
      return value > 8000 ? 'text-red-500' : value > 1000 ? 'text-yellow-500' : 'text-green-600';
    case 'noise':
```

```
      return value > 30 ? 'text-red-500' : value > 10 ? 'text-yellow-500' : 'text-green-600';
    case 'clarity':
      return value > 100 ? 'text-green-600' : value > 50 ? 'text-yellow-500' : 'text-red-500';
    default:
      return '';
  }
};

const getImageInfo = (file: File): Promise<ImageInfo> => {
  return new Promise((resolve) => {
    const img = new Image();
    const reader = new FileReader();

    reader.onload = (e) => {
      img.src = e.target?.result as string;
    };

    img.onload = () => {
      const canvas = document.createElement('canvas');
      canvas.width = img.width;
      canvas.height = img.height;
      const ctx = canvas.getContext('2d');
      if (!ctx) return;

      ctx.drawImage(img, 0, 0);
      const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
      const { data } = imageData;

      let totalLuminance = 0;
      const luminanceValues: number[] = [];

      for (let i = 0; i < data.length; i += 4) {
        const [r, g, b] = [data[i], data[i + 1], data[i + 2]];
```

```
      const luminance = 0.2126 * r + 0.7152 * g + 0.0722 * b;
      luminanceValues.push(luminance);
      totalLuminance += luminance;
    }

    const avgLuminance = totalLuminance / luminanceValues.length;
    const contrast = luminanceValues.reduce((sum, l) => sum + Math.abs(l - avgLuminance), 0) / luminanceValues.length;

    const luminanceStd = Math.sqrt(
      luminanceValues.reduce((sum, l) => sum + (l - avgLuminance) ** 2, 0) / luminanceValues.length
    );

    const noise = luminanceStd > 60 ? 'Високий' : luminanceStd > 30 ? 'Середній' : 'Низький';

    const megapixels = (img.width * img.height) / 1_000_000;
    const qualityScore = file.size / megapixels;
    const qualityRating = qualityScore < 100 ? 'Низька' : qualityScore < 300 ? 'Середня' : 'Висока';

    const clarityScore = contrast * megapixels;
    const clarity = clarityScore > 300 ? 'Висока' : clarityScore > 150 ? 'Середня' : 'Низька';

    resolve({
      name: file.name,
      size: Math.round(file.size / 1024),
      type: file.type,
      contrast: Math.round(contrast),
      brightness: Math.round(avgLuminance),
      megapixels: Math.round(megapixels * 10) / 10,
      quality: qualityRating,
      noise,
      clarity,
      data,
```

```
        width: img.width,
        height: img.height,
      });
    };

    reader.readAsDataURL(file);
  });
};

const getParameterExplanation = (value: number, type: 'contrast' | 'brightness' | 'mp' | 'size') => {
  switch (type) {
    case 'contrast':
      return value > 40
        ? 'Високий контраст (детальне зображення, хороша різкість)'
        : value > 20
          ? 'Середній контраст (зображення може бути трохи тьмяним)'
          : 'Низький контраст (може бути розмите або слабо виражене)';
    case 'brightness':
      return value > 180 || value < 60
        ? 'Яскравість занадто висока або низька (може бути засвітлене або темне)'
        : 'Оптимальна яскравість (добре збалансоване зображення)';
    case 'mp':
      return value > 8
        ? 'Більше 8 МП (висока деталізація, DSLR або сучасні смартфони)'
        : value > 3
          ? '3–8 МП (стандартна якість для мобільних пристроїв)'
          : 'Менше 3 МП (може бути занадто низька роздільна здатність)';
    case 'size':
      return value > 8000
        ? 'Файл > 8 МБ (може бути надмірно великий або з високим шумом)'
        : value > 1000
          ? '1–8 МБ (типово для середньої якості JPEG/PNG)'
          : 'Менше 1 МБ (може бути стиснений або низької якості)';
```

```
    default:
      return '';
  }
};

const ImageInfoCard: React.FC<{ info: ImageInfo }> = ({ info }) => (
  <div className="w-full bg-gray-100 rounded-lg p-4 mt-2 text-sm text-gray-800">
    <h3 className="font-semibold mb-2">Інформація про зображення</h3>
    <p><strong>Назва:</strong> {info.name}</p>

    <p>
      <strong>Розмір:</strong>
      <span className={`ml-1 ${getQualityColor(info.size, 'size')}`}>
        {info.size} КБ
      </span>
      <br />
      <em className="text-gray-500">{getParameterExplanation(info.size, 'size')}</em>
    </p>

    <p><strong>Тип:</strong> {info.type}</p>

    <p>
      <strong>Роздільна здатність:</strong>
      <span className={`ml-1 ${getQualityColor(info.megapixels, 'mp')}`}>
        {info.megapixels} МП
      </span>
      <br />
      <em className="text-gray-500">{getParameterExplanation(info.megapixels, 'mp')}</em>
    </p>

    <p>
      <strong>Яскравість:</strong>
      <span className={`ml-1 ${getQualityColor(info.brightness, 'brightness')}`}>
```

```
        {info.brightness}
      </span>
      <br />
      <em className="text-gray-500">{getParameterExplanation(info.brightness,
'brightness')}</em>
    </p>

    <p>
      <strong>Контраст:</strong>
      <span className={`ml-1 ${getQualityColor(info.contrast, 'contrast')}`}>
        {info.contrast}
      </span>
      <br />
      <em className="text-gray-500">{getParameterExplanation(info.contrast, 'contrast')}</em>
    </p>

    <p>
      <strong>Оцінка якості:</strong>
      <span className={`ml-1 font-medium ${info.quality === 'Висока' ? 'text-green-600' :
info.quality === 'Середня' ? 'text-yellow-500' : 'text-red-500'
        }`}>
        {info.quality}
      </span>
      <br />
      <em className="text-gray-500">
        Базується на співвідношенні розміру до мегапікселів (КБ/МП): Вища щільність = вища
якість
      </em>
    </p>
  </div>
);

const calculateMSE = (img1: ImageInfo | null, img2: ImageInfo | null): number | null => {
```

```
  if (!img1 || !img2 || !img1.data || !img2.data || img1.width !== img2.width || img1.height !==
img2.height) {
    return null;
  }

  let sumSquaredDifferences = 0;
  const totalPixels = img1.data.length / 4;

  for (let i = 0; i < img1.data.length; i++) {
    const diff = img1.data[i] - img2.data[i];
    sumSquaredDifferences += diff * diff;
  }

  return sumSquaredDifferences / totalPixels;
};

const calculatePSNR = (mse: number | null, maxPixelValue: number = 255): number | null => {
  if (mse === null || mse === 0) {
    return null;
  }
  return 10 * Math.log10((maxPixelValue ** 2) / mse);
};

const calculateSSIM = (img1: ImageInfo | null, img2: ImageInfo | null): number | null => {
  if (
    !img1 || !img2 ||
    !img1.data || !img2.data ||
    img1.width !== img2.width ||
    img1.height !== img2.height
  ) return null;

  const luminance = (r: number, g: number, b: number) => 0.2126 * r + 0.7152 * g + 0.0722 * b;
  const pixels = img1.width! * img1.height!;
```

```
  const l1: number[] = [];
  const l2: number[] = [];

  for (let i = 0; i < img1.data.length; i += 4) {
    l1.push(luminance(img1.data[i], img1.data[i + 1], img1.data[i + 2]));
    l2.push(luminance(img2.data[i], img2.data[i + 1], img2.data[i + 2]));
  }

  const μ1 = l1.reduce((a, b) => a + b, 0) / pixels;
  const μ2 = l2.reduce((a, b) => a + b, 0) / pixels;

  const σ1Sq = l1.reduce((sum, val) => sum + (val - μ1) ** 2, 0) / pixels;
  const σ2Sq = l2.reduce((sum, val) => sum + (val - μ2) ** 2, 0) / pixels;
  const σ12 = l1.reduce((sum, _, i) => sum + (l1[i] - μ1) * (l2[i] - μ2), 0) / pixels;

  const c1 = (0.01 * 255) ** 2;
  const c2 = (0.03 * 255) ** 2;

  const numerator = (2 * μ1 * μ2 + c1) * (2 * σ12 + c2);
  const denominator = (μ1 ** 2 + μ2 ** 2 + c1) * (σ1Sq + σ2Sq + c2);

  return numerator / denominator;
};

const SUPPORTED_IMAGE_TYPES = ['image/jpeg', 'image/png', 'image/webp', 'image/bmp'];

export const MainApp: React.FC = () => {
  const [images, setImages] = useState<(string | null)[]>([null, null]);
  const [imageInfos, setImageInfos] = useState<(ImageInfo | null)[]>([null, null]);
  const [mse, setMse] = useState<number | null>(null);
  const [psnr, setPsnr] = useState<number | null>(null);
  const [ssim, setSsim] = useState<number | null>(null);
  const [isCalculating, setIsCalculating] = useState(false);
```

```
const [error, setError] = useState<string | null>(null);

useEffect(() => {
  const calculate = async () => {
    if (
      imageInfos[0] &&
      imageInfos[1] &&
      imageInfos[0].data &&
      imageInfos[1].data &&
      imageInfos[0].width === imageInfos[1].width &&
      imageInfos[0].height === imageInfos[1].height
    ) {
      setIsCalculating(true);
      setMse(null);
      setPsnr(null);
      setSsim(null);

      const calculatedMse = calculateMSE(imageInfos[0], imageInfos[1]);
      const calculatedPsnr = calculatePSNR(calculatedMse);
      const calculatedSsim = calculateSSIM(imageInfos[0], imageInfos[1]);

      setMse(calculatedMse);
      setPsnr(calculatedPsnr);
      setSsim(calculatedSsim);
      setIsCalculating(false);
    }
  };

  calculate();
}, [imageInfos]);

const handleImageUpload = async (
  e: React.ChangeEvent<HTMLInputElement>,
```

```
  index: number
) => {
  const file = e.target.files?.[0];
  if (file) {
    const url = URL.createObjectURL(file);
    const info = await getImageInfo(file);

    if (!SUPPORTED_IMAGE_TYPES.includes(file.type)) {
      setError(`Непідтримуваний тип файлу: ${file.type}`);
      return;
    }
    setError(null);
    const newImages = [...images];
    const newInfos = [...imageInfos];
    newImages[index] = url;
    newInfos[index] = info;

    setImages(newImages);
    setImageInfos(newInfos);
  }
};

const renderKatex = (expression: string) => {
  try {
    return katex.renderToString(expression, {
      throwOnError: false,
      displayMode: true,
    });
  } catch (error) {
    console.error("Error rendering LaTeX:", error);
    return expression;
  }
};
```

```
return (
  <div className="min-h-screen bg-gray-50 p-8">
    <div className="max-w-5xl mx-auto">
      <h1 className="text-3xl font-bold text-center mb-10 text-gray-800">
        📷 Порівняння та аналіз зображень
      </h1>
      {error && (
        <div className="mt-4 p-2 bg-red-100 text-red-700 border border-red-400 rounded mb-5">
          {error}
        </div>
      )}

      <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
        {[0, 1].map((i) => (
          <div
            key={i}
            className="bg-white rounded-2xl shadow-lg p-6 flex flex-col items-center gap-4"
          >
            <label className="w-full text-center">
              <span className="block text-sm font-medium text-gray-700 mb-2">
                Завантажити зображення {i + 1}
              </span>
              <input
                type="file"
                accept="image/*"
                onChange={(e) => handleImageUpload(e, i)}
                className="block w-full text-sm text-gray-500
                           file:mr-4 file:py-2 file:px-4
                           file:rounded-full file:border-0
                           file:text-sm file:font-semibold
                           file:bg-blue-50 file:text-blue-700
                           hover:file:bg-blue-100
```

```
                    "
                />
              </label>

              {images[i] && (
                <img
                  src={images[i] as string}
                  alt={`Зображення ${i + 1}`}
                  className="w-full max-h-64 object-contain rounded-xl border shadow"
                />
              )}

              {imageInfos[i] && <ImageInfoCard info={imageInfos[i]!} />}
            </div>
          ))}
        </div>
        {isCalculating && (
          <div className="mt-8 bg-white rounded-lg shadow-md p-6 text-gray-800 text-center">
            <svg className="animate-spin h-10 w-10 mx-auto text-blue-500" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
              <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>
              <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
            </svg>
            <p className="mt-2">Обчислення MSE та PSNR...</p>
          </div>
        )}
        {(!isCalculating && mse !== null && psnr !== null && ssim !== null) ? (
          <div className="mt-8 bg-white rounded-lg shadow-md p-6 text-gray-800">
            <h2 className="text-xl font-semibold mb-4">Результати порівняння</h2>

            <div className="mb-4">
              <h3 className="font-semibold">Середньоквадратична похибка (MSE)</h3>
```

```
<p><strong>Формула:</strong> <span dangerouslySetInnerHTML={{ __html:
renderKatex(`MSE = \\frac{1}{M \\cdot N} \\sum_{i=0}^{M-1} \\sum_{j=0}^{N-1} [I(i, j) - K(i,
j)]^2`) }} /></p>

{imageInfos[0]?.width && imageInfos[0]?.height && (<p><strong>З
розмірами:</strong> <span dangerouslySetInnerHTML={{ __html: renderKatex(`MSE =
\\frac{1}{${imageInfos[0].width} \\cdot ${imageInfos[0].height}}
\\sum_{i=0}^{${imageInfos[0].width - 1}} \\sum_{j=0}^{${imageInfos[0].height - 1}} [I(i, j) -
K(i, j)]^2`) }} /></p>)}

<p>

<strong>Значення:</strong> {mse.toFixed(2)}</p>

</div>


<div>

<h3 className="font-semibold">Пікове відношення сигнал/шум (PSNR)</h3>

<p><strong>Формула:</strong> <span dangerouslySetInnerHTML={{ __html:
renderKatex(`PSNR = 10 \\log_{10} \\left( \\frac{MAX^2}{MSE} \\right)`) }} /></p>

<p><strong>З значеннями (MAX = 255, MSE = {mse !== null ? mse.toFixed(2) :
'?'}):</strong> <span dangerouslySetInnerHTML={{ __html: renderKatex(`PSNR = 10 \\log_{10}
\\left( \\frac{255^2}{${mse !== null ? mse.toFixed(2) : '?'}} \\right)`) }} /></p>

<p><strong>Значення:</strong> {psnr !== null ? psnr.toFixed(2) + ' дБ' : '?'}</p>

{psnr !== null && (

<p className="mt-2 text-sm text-gray-600">

PSNR зазвичай вимірюється в децибелах (дБ). Чим вище значення PSNR, тим
краща якість відновленого

зображення. Загалом, значення PSNR вище 30 дБ вважаються прийнятними для
більшості випадків.

</p>

)}

</div>


<div>

<h3 className="font-semibold">Індекс структурної подібності (SSIM)</h3>

<p>

<strong>Формула:</strong>{' '}

<span

dangerouslySetInnerHTML={{
```

```
            __html: renderKatex(`SSIM(x, y) = \\frac{(2\\mu_x\\mu_y + C_1)(2\\sigma_{xy} + C_2)}{(\\mu_x^2 + \\mu_y^2 + C_1)(\\sigma_x^2 + \\sigma_y^2 + C_2)}`),
          }}
        />
      </p>
      {/* <p>
        <strong>З значеннями:</strong>{' '}
        <span
          dangerouslySetInnerHTML={{
            __html: renderKatex(
              `SSIM = \\frac{(2 \\cdot ${muX?.toFixed(2) ?? '?'} \\cdot ${muY?.toFixed(2) ?? '?'} + ${c1?.toExponential?.(2) ?? '?'})` +
              `(2 \\cdot ${covXY?.toFixed(2) ?? '?'} + ${c2?.toExponential?.(2) ?? '?'})}` +
              `{(${muX?.toFixed(2) ?? '?'}^2 + ${muY?.toFixed(2) ?? '?'}^2 + ${c1?.toExponential?.(2) ?? '?'})` +
              `(${sigmaX?.toFixed(2) ?? '?'}^2 + ${sigmaY?.toFixed(2) ?? '?'}^2 + ${c2?.toExponential?.(2) ?? '?'})}`
            ),
          }}
        />
      </p> */}
      <p>
        <strong>Значення:</strong> {ssim !== null ? ssim.toFixed(4) : '?'}
      </p>
      {ssim !== null && (
        <p className="mt-2 text-sm text-gray-600">
          SSIM варіюється від -1 до 1, де 1 означає ідеальну структурну подібність між зображеннями.
          Значення ближче до 1 вказують на високу подібність, тоді як значення ближче до 0 — на меншу.
          SSIM враховує яскравість, контраст та структуру пікселів, що робить його більш узгодженим із людським сприйняттям якості.
        </p>
      )}
    </div>
  </div>
```

```
      ) : images.filter(item => !!item).length > 2 && <div className="mt-4 p-2 bg-red-100 text-red-700 border border-red-400 rounded mb-5">
        🚫 Неможливо відобразити результати порівняння. Переконайтеся, що ви завантажили два зображення з однаковими розмірами для аналізу.
      </div>}
    </div>
  </div>
 );
};



// sign-up.page.tsx
import { FormEvent, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';

export const SignupPage = () => {
 const navigate = useNavigate();

 const [formData, setFormData] = useState({
  username: '',
  email: '',
  password: '',
 });

 const [error, setError] = useState('');

 const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
  setError('');
 };

 const handleSubmit = (e: FormEvent<HTMLFormElement>) => {
  e.preventDefault();
```

```
  const users = JSON.parse(localStorage.getItem('users') || '[]');

  const existingUser = users.find((user: any) =>
    user.email.toLowerCase() === formData.email.toLowerCase()
  );

  if (existingUser) {
    setError('Даний користувач вже зареєстрований');
  } else {
    const newUser = {
      username: formData.username,
      email: formData.email,
      password: formData.password,
    };

    localStorage.setItem('users', JSON.stringify([...users, newUser]));

    sessionStorage.setItem(
      'currentUser',
      JSON.stringify({
        email: formData.email,
        password: formData.password,
        stayLoggedIn: false, // or true if you add a checkbox
      })
    );

    navigate('/main-app');
  }
};

return (
  <div className="min-h-screen flex items-center justify-center bg-green-50 p-4">
    <div className="bg-white shadow-md rounded-xl p-8 max-w-md w-full">
```

```
<h2 className="text-xl font-bold mb-4">Створити акаунт</h2>
<form className="space-y-4" onSubmit={handleSubmit}>
 <input
  type="text"
  name="username"
  placeholder="Ім'я користувача"
  value={formData.username}
  onChange={handleChange}
  required
  className="w-full border p-2 rounded-md"
 />
 <input
  type="email"
  name="email"
  placeholder="Електронна пошта"
  value={formData.email}
  onChange={handleChange}
  required
  className="w-full border p-2 rounded-md"
 />
 <input
  type="password"
  name="password"
  placeholder="Пароль"
  value={formData.password}
  onChange={handleChange}
  required
  className="w-full border p-2 rounded-md"
 />
 {error && <p className="text-red-500 text-sm">{error}</p>}
 <button
  type="submit"
  className="w-full bg-green-600 text-white p-2 rounded-md hover:bg-green-700"
```

```
          >
            Зареєструватися
          </button>
        </form>
        <p className="text-sm text-center mt-4">
          Вже маєте акаунт?{' '}
          <Link to="/login" className="text-green-600 hover:underline">
            Увійти
          </Link>
        </p>
      </div>
    </div>
  );
};


// log-in.page.tsx
import { useState, FormEvent } from 'react';
import { Link, useNavigate } from 'react-router-dom';

export const LoginPage = () => {
  const navigate = useNavigate();
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [isStayLogged, setIsStayLogged] = useState(false);
  const [error, setError] = useState('');

  const saveUserSession = (user: { email: string; password: string; stayLoggedIn: boolean }) => {
    sessionStorage.setItem('currentUser', JSON.stringify(user));
    if (user.stayLoggedIn) {
      localStorage.setItem('currentUser', JSON.stringify(user));
    }
  };
```

```
const handleSubmit = (e: FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  const users = JSON.parse(localStorage.getItem('users') || '[]');

  const existingUser = users.find(
    (user: any) => user.email === email && user.password === password
  );

  if (!existingUser) {
    setError('Користувача не знайдено або пароль неправильний');
    return;
  }

  saveUserSession({ email, password, stayLoggedIn: isStayLogged });
  navigate('/main-app');
};

return (
  <div className="min-h-screen flex items-center justify-center bg-blue-50 p-4">
    <div className="bg-white shadow-md rounded-xl p-8 max-w-md w-full">
      <h2 className="text-xl font-bold mb-4">Вхід</h2>
      <form className="space-y-4" onSubmit={handleSubmit}>
        <input
          type="email"
          placeholder="Електронна пошта"
          className="w-full border p-2 rounded-md"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
        <input
          type="password"
```

```
          placeholder="Пароль"
          className="w-full border p-2 rounded-md"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
        <label className="flex items-center space-x-2 text-sm">
          <input
            type="checkbox"
            checked={isStayLogged}
            onChange={(e) => setIsStayLogged(e.target.checked)}
          />
          <span>Залишатися в системі</span>
        </label>
        {error && <p className="text-red-500 text-sm">{error}</p>}
        <button
          type="submit"
          className="w-full bg-blue-600 text-white p-2 rounded-md hover:bg-blue-700"
        >
          Увійти
        </button>
      </form>
      <p className="text-sm text-center mt-4">
        Ще не маєте акаунту?{' '}
        <Link to="/signup" className="text-blue-600 hover:underline">
          Зареєструватися
        </Link>
      </p>
    </div>
  </div>
 );
};
```

```
// welcome-page.page.tsx
import { useNavigate } from "react-router-dom";

export const WelcomePage = () => {
  const navigate = useNavigate();

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100 p-4">
      <div className="bg-white rounded-2xl shadow-xl p-10 max-w-md w-full text-center">
        <h1 className="text-2xl font-bold mb-6">Ласкаво просимо</h1>

        <div className="space-y-4">
          <button
            onClick={() => navigate("/login")}
            className="w-full bg-blue-600 hover:bg-blue-700 text-white py-2 px-4 rounded-xl"
          >
            Увійти
          </button>

          <button
            onClick={() => navigate("/signup")}
            className="w-full bg-green-600 hover:bg-green-700 text-white py-2 px-4 rounded-xl"
          >
            Створити акаунт
          </button>

          <div className="relative group w-full">
            <button
              className="w-full bg-gray-300 hover:bg-gray-400 text-black py-2 px-4 rounded-xl"
              onClick={() => navigate("/main-app")}
            >
              Використовувати без входу
```

```
            </button>

            <div className="absolute left-1/2 -translate-x-1/2 mt-2 w-max px-3 py-2 bg-yellow-100 text-yellow-800 text-sm rounded-md shadow-md opacity-0 group-hover:opacity-100 transition-opacity duration-200 z-10">
                ⚠ Оцінка якості зображення не буде збережена.
            </div>
          </div>
        </div>
      </div>
    </div>
  );
};
```

Додаток Г – Графічна частина

ГРАФІЧНА ЧАСТИНА

ПРОГРАМНА СИСТЕМА ОЦІНЮВАННЯ ЯКОСТІ ГРАФІЧНИХ ЗОБРАЖЕНЬ

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: Програмна система оцінювання якості графічних зображень

Виконав:

студент групи ЗПІ-21б Рябоконь А.М.

Науковий керівник:

д.т.н., проф. кафедри ПЗ Романюк О.Н.

Рисунок Г.1 – Титульний аркуш

Мета, предмет та об'єкт дослідження

Метою дослідження є підвищення ефективності оцінювання якості графічних зображень шляхом розробки програмних модулів, які поєднують експертні та аналітичні методи для автоматизованого аналізу і порівняння зображень.

Об'єктом дослідження є процес оцінювання якості графічних зображень.

Предметом дослідження є методи та засоби аналізу й порівняння графічних зображень.

Рисунок Г.2 – Мета, предмет та об'єкт дослідження

Наукова новизна розробки

1. Вперше запропоновано адаптивний метод оцінювання якості зображень, особливість якого полягає в том, що враховуються характеристики спотворень зображень для динамічного визначення вагових коефіцієнтів класичних об'єктивних метрик: PSNR, SSIM, MSE, FSSIM. Це дозволяє формувати гнучку та узгоджену з людським сприйняттям оцінку якості зображення.
2. Розроблено гібридний метод оцінювання якості зображень, який поєднує адаптивне комбінування об'єктивних метрик: PSNR, SSIM, FSSIM, MSE із суб'єктивними оцінками сприйняття якості. Метод передбачає аналіз характеру спотворень зображення з подальшим динамічним визначенням вагових коефіцієнтів для кожної метрики, що забезпечує підвищення узгодженості об’єктивної оцінки з візуальним сприйняттям людини.

Рисунок Г.3 – Наукова новизна розробки

Постановка задач

- Розробити алгоритм оцінювання якості зображень на основі аналітичних методів (метрики SSIM, PSNR, MSE тощо).
- Розробити програмний модуль для суб'єктивної оцінки якості зображень, що враховуватиме експертні параметри та зворотний зв’язок користувачів.
- Реалізувати програмний модуль порівняння зображень для аналізу відмінностей між оригінальним та обробленим зображенням за допомогою методів адаптивного та гібридного оцінювання.
- Розробити графічний інтерфейс для взаємодії користувача з системою оцінювання.
- Протестувати програмні модулі на різних графічних зображеннях для перевірки їх ефективності та коректності роботи.

Рисунок Г.4 – Постановка задач

Аналіз аналогів

Критерії	OpenCV	NIQE	Google Cloud Vision API	Власний застосунок
Об'єктивний метод оцінки	+	-	+	+
Аналітичний метод оцінки	-	+	+	+
Потрібність еталонного зображення	+	-	-	±
Простота інтеграції у React	-	±	+	+
Гнучкість налаштувань	+	-	-	+
Необхідність підключення до інтернет	-	-	+	+
Вартість (безкоштовність)	+	+	-	+
Загальна оцінка (%)	57%	43%	57%	71%

Рисунок Г.5 – Аналіз аналогів

Рисунок Г.6 – Варіантний аналіз розробки

Аналіз методу адаптивного вибору критерію оцінювання зображень

Для подолання обмежень окремих об'єктивних метрик оцінювання якості зображень, таких як: PSNR, SSIM та FSSIM, запропоновано метод адаптивного вибору критерію оцінювання AEMC, який динамічно визначає найбільш релевантну метрику або їх комбінацію на основі аналізу характеристик спотворень зображення (розподілу пікселів, частотних характеристик, артефактів тощо) та подальшого визначення вагових коефіцієнтів для кожної метрики, чутливої до виявлених спотворень, з метою отримання більш об'єктивної та узгодженої з людським сприйняттям комбінованої оцінки якості, що відкриває перспективи для подальшого розвитку в напрямку вдосконалення методів аналізу спотворень, автоматичного визначення вагових коефіцієнтів, розширення набору метрик та дослідження нелінійних методів їх комбінування.

Рисунок Г.7 – Аналіз методу адаптивного вибору критерію оцінювання зображень

Аналіз гібридного методу оцінювання якості зображень на основі комбінування об’єктивних і суб’єктивних метрик

Гібридний метод оцінювання якості зображень комбінує об’єктивні метрики: SSIM, FSIM, PSNR, MSE, які автоматично розраховують ступінь схожості зображень на основі математичних формул, та суб’єктивні оцінки такі як: MOS, шкала Лайкерта, метод Кендала, отримані в результаті опитування експертів для відображення людського сприйняття; для інтеграції цих різних типів даних використовується нормалізоване середнє зважене значення, де кожній метриці присвоюється ваговий коефіцієнт, що визначає її внесок у підсумкову оцінку, яка потім може бути використана для ранжування зображень або порівняння алгоритмів обробки, забезпечуючи таким чином більш точну, надійну та гнучку оцінку якості, що поєднує об'єктивність автоматизованих методів з чутливістю людського зору

Рисунок Г.8 – Аналіз гібридного методу оцінювання якості зображень на основі комбінування об’єктивних і суб’єктивних метрик

Рисунок Г.9 – Алгоритм роботи системи

Діаграма алгоритму функції визначення характеристик зображення

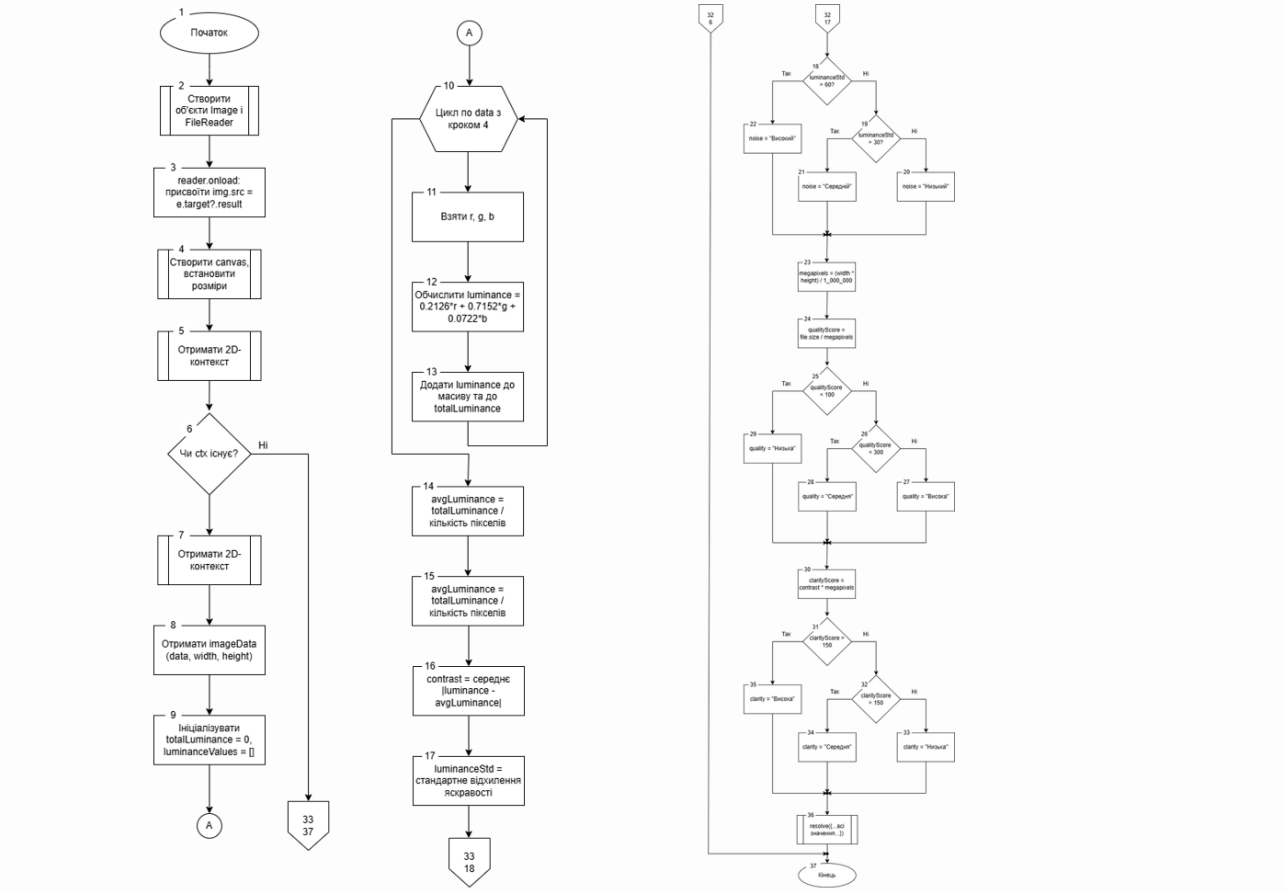


Рисунок Г.10 – Діаграма алгоритму функції визначення характеристик зображення

Рисунок Г.11 – Діаграма алгоритму обчислення MSE

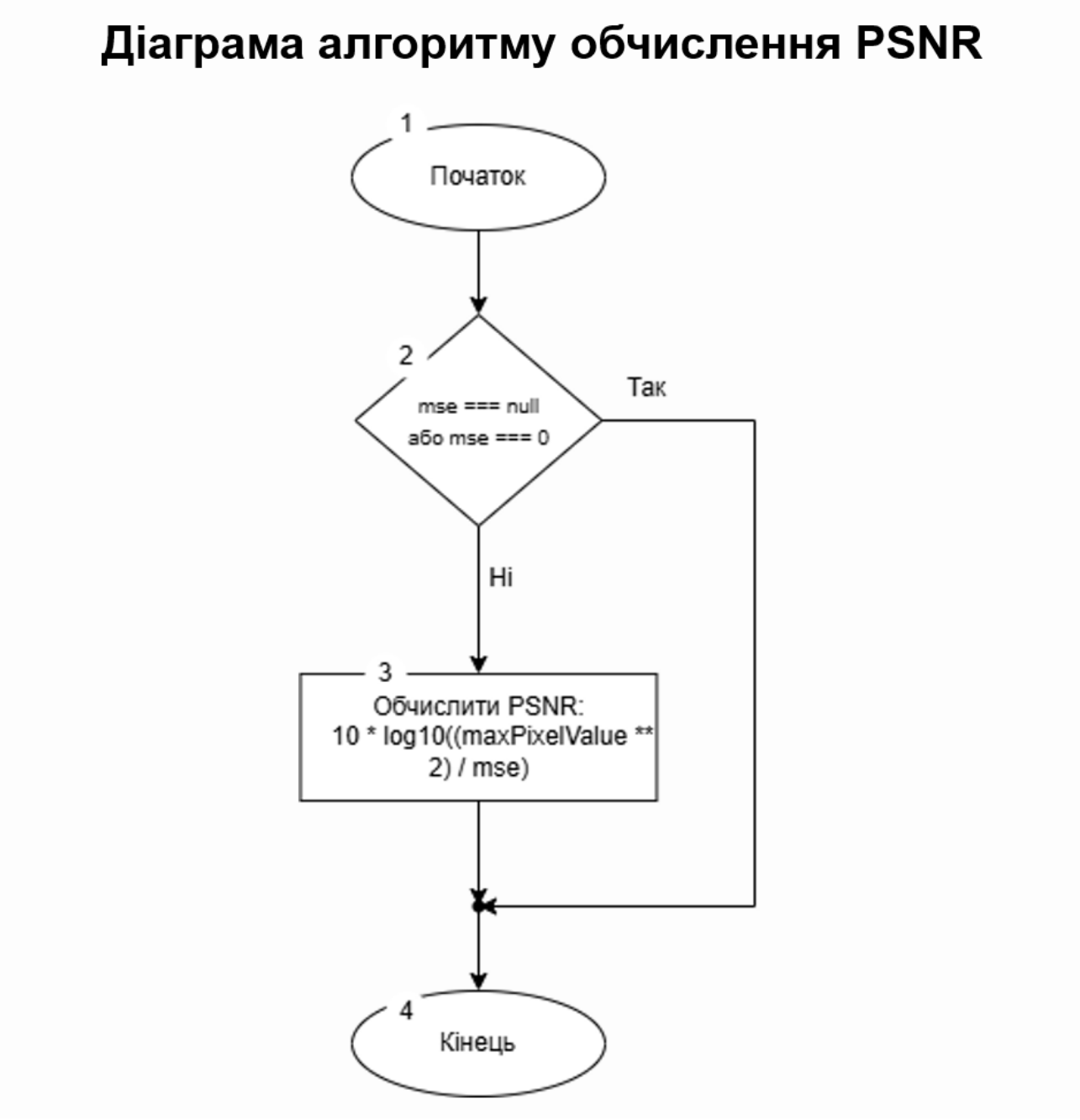


Рисунок Г.12 – Діаграма алгоритму обчислення PSNR

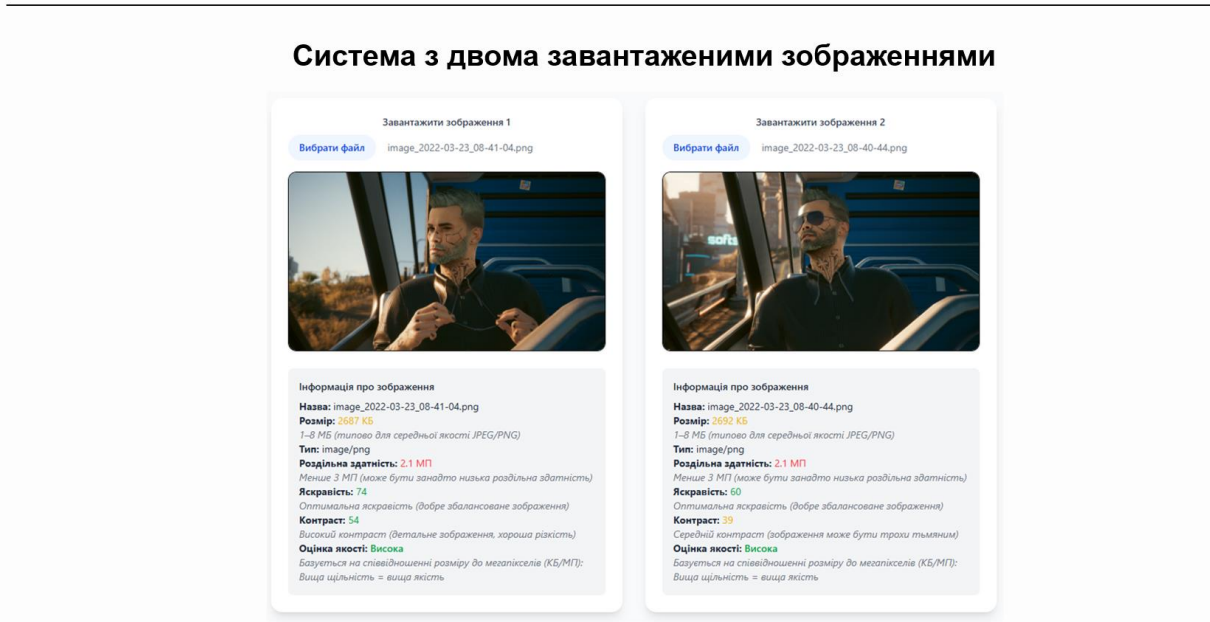


Рисунок Г.13 – Система з двома завантаженими зображеннями

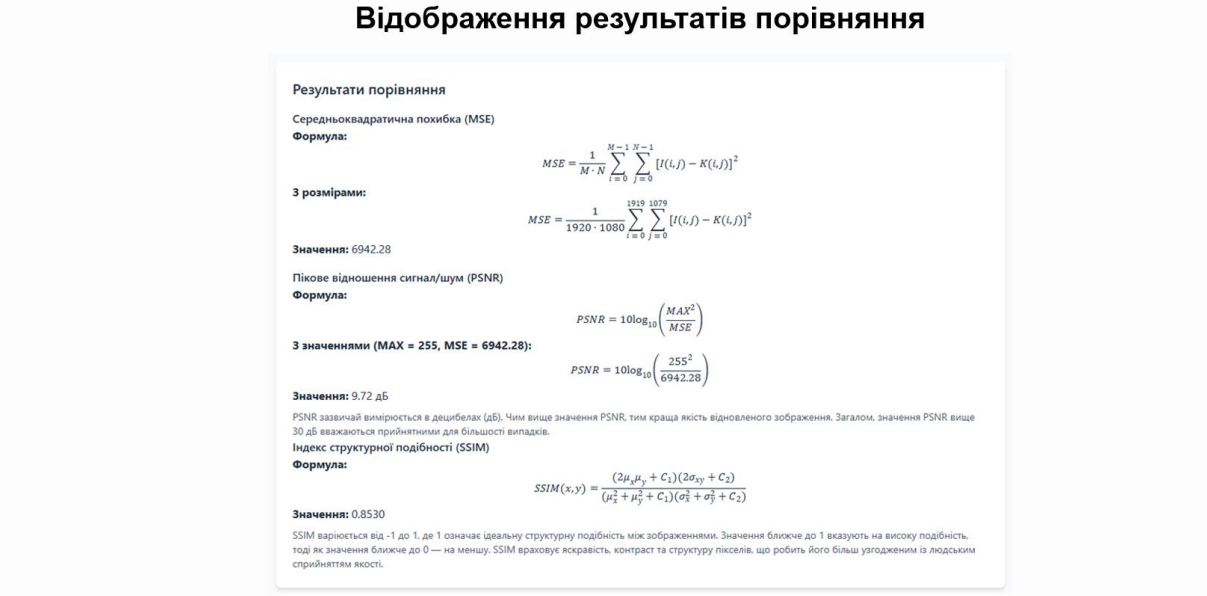


Рисунок Г.14 – Відображення результатів порівняння

Висновки

У результаті виконання бакалаврської кваліфікаційної роботи було:

- проведено аналіз існуючих методів оцінювання якості графічних зображень, виявлено їхні переваги та недоліки;
- обґрунтовано необхідність розробки комплексної системи оцінювання якості графічних зображень, яка враховує як об'єктивні метрики, так і суб'єктивне сприйняття;
- вперше запропоновано адаптивний метод оцінювання якості зображень, що динамічно визначає вагові коефіцієнти класичних об'єктивних метрик на основі характеристик спотворень;
- вперше розроблено гібридний метод оцінювання якості зображень, що інтегрує адаптивне комбінування об'єктивних метрик із суб'єктивними оцінками експертів;
- розроблено алгоритми та програмні модулі, що реалізують запропоновані адаптивний та гібридний методи оцінювання якості графічних зображень;
- проведено тестування розроблених програмних модулів на різних графічних зображеннях, що підтвердило їхню ефективність та коректність роботи.

Рисунок Г.15 – Висновки

Апробації та публікації

1. Рябоконь А.М., Романюк О.Н. Аналітичні методи оцінки якості зображень. Матеріали XII Всеукраїнської науково-практичної конференції молодих учених (2025), Київ, 15 травня 2025 р. – Видавництво Київський столичний університет імені Бориса Грінченка, 2025р. – С. 246-248.

2. Рябоконь А.М., Романюк О.Н. Використання метрики PSNR для оцінки якості зображень. Матеріали LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025), Вінниця, 24-27 вересня 2025 р. – Вінниця.

3. Рябоконь А.М., Романюк О.Н. Методи математичного моделювання для об'єктивної та експертної оцінки якості цифрових зображень. Матеріали X Всеукраїнської науково-методичної конференції «Освіта, наука та виробництво: розвиток та перспективи» (2025), Суми, 24 квітня 2025 р. – Видавництво Сумський державний університет, 2025р. – С. 192-194.

4. Рябоконь А.М., Романюк О.Н. Оцінка якості зображень на основі аналітичних та експертних методів. Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів (2025), Одеса, 17-18 квітня 2025 р. – Видавництво ОНТУ, 2025р. – С. 151-153.

5. Рябоконь А.М., Романюк О.Н. Оцінювання якості зображень у комп’ютерному моделюванні технологічних процесів на основі показників psnr та mse. Матеріали Всеукраїнської науково-практичної Internet-конференції (2025), Черкаси, 17-21 березня 2025 р. – С. 109-110.

6. Рябоконь А.М., Романюк О.Н. Комбіноване використання суб’єктивних і об’єктивних метрик оцінювання якості цифрових зображень. Матеріали XXXIII Міжнародної науково-практичної конференції MicroCAD-2025 (2025), Харків, 14-17 травня 2025 р – Видавництво НТУ «Харківський політехнічний інститут», 2025р. – С. 1528.

Рисунок Г.16 – Апробації та публікації

Рисунок Г.17 – Закінчення презентації