ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Алгоритм аналізу зображень обличчя для визначення кольоротипу користувача»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Анна ТИЩЕНКО

(підпис)

Виконала: здобувачка вищої освіти групи ПДМ-61
Анна ТИЩЕНКО

Керівник: Богдан ХУДІК
доктор філософії (PhD)

Рецензент:
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тищенко Анні Володимирівні

1. Тема кваліфікаційної роботи: «Алгоритм аналізу зображень обличчя для визначення кольоротипу користувача»

керівник кваліфікаційної роботи Богдан ХУДІК, доктор філософії (PhD), доцент кафедри ІПЗ.

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, база даних зображень обличчя з попередньо визначеними кольоротипами, Метод сегментації та детектування ключових точок обличчя, моделі попередньої обробки зображення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз існуючих методів та алгоритмів визначення кольоротипу людини.
2. Розробка гібридного методу визначення кольоротипу на основі зображення.
3. Впровадження механізму попередньої обробки зображення.
4. Розробка вебзастосунку для реалізації розробленого методу.

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження.
2. Аналіз існуючих методів для визначення кольоротипу людини.
3. Схема алгоритму роботи розробленого методу.
4. Опис етапів попередньої обробки зображення, формули.
5. Опис розробленого вебзастосунку для визначення кольоротипу користувача.
6. Демонстрація вигляду вебзастосунку.
7. Висновки.
8. Публікації та апробація роботи.

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	31.10–01.11.2025	
2	Аналіз предметної області визначення колористичного типу	01.11–7.11.2025	
3	Дослідження існуючих методів визначення кольоротипу людини та обробки зображення	7.11–14.11.2025	
4	Дослідження методів обробки зображень	14.11–18.11.2025	
5	Розробка методики з Retinex, Face Alignment та сегментацією	18.11–28.11.2025	
6	Проєктування архітектури вебзастосунку	28.11–05.12.2025	
7	Оформлення роботи: вступ, висновки, реферат	05.12–17.12.2025	
8	Розробка демонстраційних матеріалів	17.12–19.12.2025	

Здобувач вищої освіти _____ Анна ТИЩЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Богдан ХУДІК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: _69_ стор., _2_ табл., _20_ рис., 25 джерел.

Мета роботи – підвищення адаптивності визначення кольоротипу користувача на основі зображення обличчя.

Об'єкт дослідження – процес визначення кольоротипу користувача на основі аналізу зображення обличчя.

Предмет дослідження – метод для визначення кольоротипу користувача на основі зображення обличчя, що поєднує модель штучного інтелекту та метод попередньої обробки зображення.

Короткий зміст роботи: У роботі проведено аналіз предметної області визначення кольоротипу людини на основі зображень обличчя, розглянуто особливості формування кольорових характеристик зовнішності та обмеження традиційних і алгоритмічних підходів у задачах автоматизованого аналізу. Досліджено існуючі методи визначення кольоротипу, зокрема колориметричні правила, використання еталонних кольорових шкал і сучасні підходи на основі штучного інтелекту, визначено їхні недоліки, пов'язані з чутливістю до умов освітлення, якістю зображень і нестабільністю результатів. Запропоновано комбінований метод визначення кольоротипу, що поєднує попередню обробку зображень для нормалізації освітлення та зменшення впливу артефактів із застосуванням моделі штучного інтелекту для класифікації кольоротипу.

Реалізовано алгоритмічну архітектуру методу, що включає етапи вирівнювання обличчя, сегментації ключових областей та аналізу кольорових характеристик. Проведено експериментальне дослідження, результати якого підтвердили підвищення стабільності та узгодженості визначення кольоротипу в умовах неконтрольованої зйомки, що свідчить про доцільність використання запропонованого підходу в автоматизованих інформаційних системах.

КЛЮЧОВІ СЛОВА: КОЛЬОРОТИП, АНАЛІЗ ЗОБРАЖЕНЬ ОБЛИЧЧЯ, КОМП'ЮТЕРНИЙ ЗІР, ПОПЕРЕДНЯ ОБРОБКА ЗОБРАЖЕНЬ, НОРМАЛІЗАЦІЯ ОСВІТЛЕННЯ, СЕГМЕНТАЦІЯ ОБЛИЧЧЯ, КОЛІРНІ ПРОСТОРИ, ШТУЧНИЙ ІНТЕЛЕКТ, НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

Text part of the master's qualification work: _69_ pages, _20_ pictures, _2_ table, _25_ sources.

The purpose of the work is to enhance the adaptability of user color type determination based on facial images.

Object of research – the process of determining the user's color type based on the analysis of facial images.

Subject of research – a method for determining the user's color type based on facial images that combines an artificial intelligence model with an image preprocessing technique.

Summary of the work: The thesis presents an analysis of the subject area related to determining a person's color type based on facial images, examining the formation of color characteristics of appearance and the limitations of traditional and algorithmic approaches in automated analysis tasks. Existing methods for color type determination are investigated, including colorimetric rules, the use of reference color scales, and modern artificial intelligence–based approaches, and their drawbacks related to sensitivity to lighting conditions, image quality, and result instability are identified. A combined method for color type determination is proposed, which integrates image preprocessing for illumination normalization and artifact reduction with an artificial intelligence model for color type classification. An algorithmic architecture of the method is implemented, including stages of face alignment, segmentation of key facial regions, and analysis of color characteristics. An experimental study is conducted, the results of which confirm improved stability and consistency of color type determination under uncontrolled image acquisition conditions, demonstrating the feasibility of applying the proposed approach in automated information systems.

KEYWORDS:COLOR TYPE, FACIAL IMAGE ANALYSIS, COMPUTER VISION, IMAGE PREPROCESSING, ILLUMINATION NORMALIZATION, FACE SEGMENTATION, COLOR SPACES, ARTIFICIAL INTELLIGENCE, NEURAL NETWORKS.

ЗМІСТ

ВСТУП 12

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ВИЗНАЧЕННЯ КОЛЬОРОТИПУ ЛЮДИНИ 14

1.1 Поняття та значення кольоротипу 14

1.2 Традиційні методи визначення кольоротипу та їх обмеження 17

1.3 Алгоритмічні та колориметричні підходи до аналізу кольоротипу 18

1.4 Сучасні AI-підходи та обґрунтування комбінованого методу 20

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ КОЛЬОРОТИПУ ЛЮДИНИ 22

2.1 Огляд існуючих методів 22

2.1.1 Колориметричні правила (статичні алгоритми) 22

2.1.2 Кольорові карти та тонові шкали 29

2.1.3 Шкала Фіцпатріка 32

2.1.4 Система Pantone 33

2.1.5 Порівняння з еталонними наборами 34

2.1.6 Сучасні AI-методи та нейронні мережі 35

2.2 Порівняння існуючих методів для визначення кольоротипу 41

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕТОДУ ВИЗНАЧЕННЯ КОЛЬОРОТИПУ 43

3.1 Архітектура системи 43

3.1.1 React Application 45

3.1.2 Pages Component 45

3.1.3 UI Components 47

3.1.4 React Router 48

3.1.5 Express API 48

3.1.6 Controllers Component 48

3.1.7 Image Processor 50

3.1.8 Етапи обробки зображення 50
3.1.9 MongoDB Database 52
3.1.10 Ollama AI Service 52
3.2 Нормалізація освітлення 53
3.3 Вирівнювання обличчя 57
3.4 Сегметація областей 62
3.5 Результати тестування та порівняння методів 67
3.6 Вебзастосунок ColorMuse 69
ПЕРЕЛІК ПОСИЛАНЬ 81
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 84
ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ 90

ВСТУП

У сучасному світі цифрові технології дедалі глибше інтегруються в повсякденне життя людини, зокрема у сферу персоналізованих сервісів, моди, косметології та візуальних рекомендацій. Поширеність мобільних пристроїв із високоякісними камерами відкриває можливості для створення інтерактивних систем, здатних аналізувати зовнішність користувача та пропонувати індивідуальні поради. Одним із важливих аспектів персоналізації у б'юті та fashion-індустрії є визначення кольоротипу — системи класифікації зовнішності за гармонією відтінків шкіри, волосся та очей. Рекомендації на основі кольоротипу дозволяють користувачеві підбирати одяг, косметику та аксесуари, що найбільш вигідно підкреслюють природні особливості його зовнішності.

Попри значний попит на такі сервіси, більшість наявних рішень або є комерційними й закритими, або використовують ручне введення параметрів, що знижує об'єктивність і точність результатів. Тому виникає потреба у створенні алгоритму, здатного автоматично аналізувати зображення обличчя, виконувати попередню обробку, виділення ключових областей та на основі об'єктивних параметрів визначати кольоротип користувача. Розв'язання цієї задачі поєднує комп'ютерний зір, машинне навчання, статистичний аналіз зображень та прикладні методи колористики, що робить тему дослідження сучасною, актуальною та міждисциплінарною.

Магістерська робота спрямована на розробку та дослідження алгоритму автоматичного аналізу фотографії обличчя, який дозволяє отримувати ключові параметри зовнішності, нормалізувати вплив освітлення, визначати геометрію та колірні характеристики шкіри, очей і волосся, а також виконувати класифікацію кольоротипу. Робота присвячена створенню технологічної основи для подальшого застосування у мобільних додатках або веб-сервісах, що надають персоналізовані рекомендації щодо іміджу та догляду.

Мета роботи – підвищення точності та автоматизація процесу визначення

кольоротипу користувача на основі зображення обличчя.

Об’єкт дослідження – процес визначення кольоротипу користувача на основі аналізу зображення обличчя.

Предмет дослідження – метод для визначення кольоротипу користувача на основі зображення обличчя, що поєднує модель штучного інтелекту та метод попередньої обробки зображення.

Для досягнення мети вирішувалися наступні завдання:

1. Огляд існуючих методів. Проведено дослідження і аналіз існуючих наукових джерел, щоб отримати повне уявлення про різноманітні методи та технології, що використовуються в детектуванні об'єктів та глибокому навчанні.
2. Моделювання та програмування. Розроблено метод для визначення кольоротипу людини на основі зображення обличчя, виконано програмування для реалізації алгоритмів та проведено навчання та тестування на даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПІДХОДІВ ДО ВИЗНАЧЕННЯ КОЛЬОРОТИПУ ЛЮДИНИ

1.1 Поняття та значення кольоротипу

Кольоротип людини є складною інтегральною характеристикою зовнішності, яка формується на основі поєднання природних кольорових параметрів шкіри, очей та волосся, а також особливостей їх взаємодії у візуальному сприйнятті. У науковому сенсі поняття кольоротипу знаходиться на перетині теорії кольору, фізіології зору, психології сприйняття та прикладної колориметрії. Воно використовується для опису того, як різні кольори взаємодіють із зовнішністю людини та які з них створюють гармонійний або, навпаки, дисгармонійний візуальний ефект.

Формування поняття кольоротипу пов’язане з ідеєю про те, що кожна людина має індивідуальний набір природних відтінків, який визначає оптимальну кольорову палітру для одягу, макіяжу та аксесуарів. Цей підхід ґрунтується на спостереженні, що однакові кольори можуть по-різному впливати на вигляд різних людей: підкреслювати природну свіжість обличчя, робити шкіру візуально рівнішою або, навпаки, акцентувати недоліки, створювати ефект втоми чи нездорового тону.

З фізіологічної точки зору ключову роль у формуванні кольоротипу відіграють спектральні характеристики відбитого світла від поверхні шкіри, пігментація райдужної оболонки ока та колір волосся. Шкіра людини має складну багатошарову структуру, а її колір визначається співвідношенням меланіну, каротиноїдів та гемоглобіну. Саме ці компоненти формують індивідуальний відтінок шкіри, який може змінюватися залежно від освітлення, але при цьому зберігає певні стабільні властивості. Колір очей і волосся, у свою чергу, створюють додаткові акценти та впливають на загальну контрастність зовнішності.

Значення кольоротипу у сучасних умовах виходить далеко за межі традиційної сфери стилю та моди. В умовах цифровізації зростає кількість онлайн-

сервісів і мобільних застосунків, орієнтованих на персоналізовані рекомендації, де визначення кольоротипу використовується як базовий елемент адаптації контенту до конкретного користувача. У таких системах коректне визначення кольоротипу є важливим чинником довіри до результатів та загального користувацького досвіду.

Разом із тим поняття кольоротипу має певну абстрактність і не має чітко визначених фізичних меж, що ускладнює його формалізацію. На відміну від суто метричних характеристик, таких як яскравість або насиченість, кольоротип включає в себе елемент візуальної гармонії, який складно описати однозначними числовими показниками. Саме ця особливість зумовлює складність автоматизації процесу визначення кольоротипу та необхідність комплексного підходу до його аналізу.

Окремої уваги в контексті визначення кольоротипу потребує питання стабільності та варіативності кольорових характеристик зовнішності. Хоча базові параметри кольору шкіри, очей і волосся вважаються відносно сталими, на практиці вони можуть змінюватися під впливом зовнішніх і внутрішніх факторів. До таких факторів належать сезонні зміни освітлення, фізіологічний стан людини, рівень втоми, температура навколишнього середовища, а також використання косметичних засобів. Це означає, що кольоротип не може розглядатися як абсолютно фіксована характеристика, а радше як умовно стабільна категорія, визначення якої потребує врахування контексту отримання зображення.

Значну роль у формуванні поняття кольоротипу відіграє перцептивний аспект сприйняття кольору людиною. Відомо, що людський зір не є лінійним сенсором і по-різному реагує на зміну яскравості та контрасту в різних умовах. Той самий відтінок шкіри може сприйматися як тепліший або холодніший залежно від оточуючих кольорів, балансу білого та спектрального складу освітлення. У зв’язку з цим кольоротип у практичному розумінні є не лише фізичною характеристикою, а й результатом складної взаємодії між об’єктивними параметрами зображення та суб’єктивним сприйняттям.

Важливим аспектом є також міжкультурна та етнічна варіативність кольоротипів. Більшість класичних систем класифікації були розроблені на основі

обмежених вибірок і не завжди коректно описують різноманіття відтінків шкіри та комбінацій кольорів, характерних для різних популяцій. Це створює додаткові труднощі при використанні універсальних підходів та підкреслює необхідність алгоритмів, здатних адаптуватися до широкого спектра зовнішніх характеристик без жорсткої прив’язки до заздалегідь визначених категорій.

З точки зору формалізації, кольоротип можна розглядати як латентну ознаку, яка не вимірюється безпосередньо, а визначається опосередковано через сукупність спостережуваних параметрів. До таких параметрів належать середні значення кольору шкіри, рівень контрасту між елементами обличчя, спектральні характеристики очей і волосся, а також їх відносні пропорції в загальній композиції обличчя. Саме латентний характер кольоротипу ускладнює побудову точних аналітичних моделей і зумовлює необхідність використання методів машинного навчання.

Окрім естетичного значення, кольоротип має прикладну цінність у суміжних галузях, зокрема в комп’ютерній графіці, доповненій реальності та персоналізованих цифрових сервісах. У таких системах визначення кольоротипу використовується для автоматичного підбору візуальних параметрів, що підвищує рівень персоналізації та зменшує когнітивне навантаження на користувача. Це додатково підкреслює актуальність досліджень, спрямованих на створення надійних і відтворюваних методів його визначення.

Таким чином, кольоротип людини слід розглядати як багатовимірну характеристику, що поєднує фізичні, перцептивні та контекстні складові. Його визначення в автоматизованих системах потребує комплексного підходу, здатного враховувати як об’єктивні параметри зображення, так і особливості людського зорового сприйняття, що створює наукове підґрунтя для подальших досліджень у межах даної магістерської роботи.

1.2 Традиційні методи визначення кольоротипу та їх обмеження

Традиційні методи визначення кольоротипу сформувалися задовго до появи цифрових технологій і ґрунтуються переважно на візуальній оцінці зовнішності людини. Найбільш відомим і поширеним є сезонний підхід, відповідно до якого всі люди умовно поділяються на групи, що асоціюються з порами року. Кожна з таких груп характеризується певним поєднанням температури кольору, рівня контрастності та насиченості відтінків.

У практиці стилістів аналіз кольоротипу зазвичай проводиться в контрольованих умовах освітлення з використанням спеціальних кольорових тканин або палітр. Ці зразки по черзі прикладаються до обличчя людини, після чого оцінюється реакція шкіри, поява або зникнення тіней, зміна сприйняття кольору очей та загального тону обличчя. Такий метод дозволяє враховувати складні візуальні ефекти, однак його результати значною мірою залежать від суб’єктивного сприйняття експерта.

Іншою поширеною групою традиційних методів є анкетні та тестові підходи, які активно використовуються в онлайн-середовищі. Вони передбачають відповіді користувача на запитання щодо природного кольору волосся, очей, реакції шкіри на сонячне випромінювання або рівня контрасту між рисами обличчя. Попри простоту реалізації, такі методи мають низьку надійність, оскільки базуються на самооцінці, яка часто є неточною або упередженою.

Важливою характеристикою традиційних методів визначення кольоротипу є їхня залежність від контекстуального аналізу, який неможливо повністю стандартизувати. Навіть за наявності однакових інструментів — кольорових палітр, тканин або тестових карт — результати аналізу можуть суттєво відрізнятися залежно від досвіду спеціаліста, його індивідуального сприйняття кольору та навіть психологічного стану. Це створює додатковий рівень варіативності, який не може бути усунений без переходу до формалізованих алгоритмічних рішень.

Окрему проблему становить відсутність чітких меж між окремими кольоротипами в межах класичних класифікацій. На практиці значна частина

людей має проміжні характеристики, які не дозволяють однозначно віднести їх до певної категорії. Традиційні системи часто змушують користувача до жорсткого вибору між кількома типами, ігноруючи можливість існування гібридних або перехідних варіантів. Це знижує практичну цінність таких методів у реальних умовах.

Ще одним суттєвим обмеженням є неможливість масштабування традиційних підходів. Вони не пристосовані до використання у масових цифрових сервісах, де необхідно обробляти велику кількість користувачів без залучення експертів. У результаті традиційні методи залишаються ефективними лише в індивідуальних консультаціях, але не відповідають вимогам сучасних автоматизованих систем персоналізації.

Таким чином, хоча класичні підходи історично відіграли важливу роль у формуванні концепції кольоротипу, їхні методологічні обмеження суттєво ускладнюють інтеграцію в сучасні інформаційні технології та стимулюють пошук більш формалізованих і відтворюваних рішень.

1.3 Алгоритмічні та колориметричні підходи до аналізу кольоротипу

Алгоритмічні методи визначення кольоротипу виникли з розвитком комп’ютерного зору та цифрової обробки зображень. Вони базуються на аналізі кольорових характеристик пікселів зображення обличчя та використанні формалізованих правил для класифікації користувача. Ключовим етапом таких методів є перехід від простору RGB до альтернативних колірних просторів, які дозволяють розділити інформацію про відтінок, яскравість та насиченість.

Найбільш поширеними є простори HSV, YCbCr та CIE Lab. Простір HSV дозволяє безпосередньо аналізувати відтінок як окрему характеристику, що є зручним для оцінки температури кольору. Простір YCbCr широко використовується для виділення області шкіри завдяки відносній стабільності хроматичних компонентів. Простір Lab є перцептивно однорідним і дозволяє наблизити аналіз до особливостей людського зору.

У межах колориметричних підходів зазвичай здійснюється сегментація обличчя та виділення ділянок шкіри, після чого обчислюються середні значення кольорових компонент або їх розподіли. Отримані характеристики порівнюються з еталонними шкалами, такими як шкала Фіцпатріка або Pantone SkinTone Guide, що дозволяє співвіднести відтінок шкіри з наперед визначеними класами.

Попри формалізований характер, такі методи мають суттєві обмеження. Вони є надзвичайно чутливими до змін освітлення, тіней, відблисків та шумів камери. Навіть незначні відхилення умов зйомки можуть призводити до істотних змін кольорових характеристик, що негативно впливає на стабільність результатів. Крім того, більшість алгоритмічних підходів зосереджується виключно на аналізі шкіри, ігноруючи роль кольору очей і волосся у формуванні кольоротипу.

Алгоритмічні методи визначення кольоротипу, на відміну від традиційних, ґрунтуються на формальному описі кольорових характеристик, однак вони також мають низку концептуальних обмежень. Однією з ключових проблем є спрощене трактування кольору як сукупності незалежних числових параметрів. У реальності ж сприйняття кольору є нелінійним процесом, у якому важливу роль відіграють контраст, локальне оточення та адаптація зорової системи.

Колориметричні підходи часто використовують усереднені значення кольору в межах певної області, що призводить до втрати просторової інформації. Наприклад, нерівномірність тону шкіри, локальні тіні або відблиски можуть істотно впливати на візуальне сприйняття, але при цьому залишатися непоміченими під час статистичної обробки. Це особливо критично в умовах неконтрольованої зйомки, характерної для мобільних застосунків.

Ще одним обмеженням є жорстка прив’язка до еталонних шкал. Хоча такі шкали забезпечують певний рівень стандартизації, вони не враховують індивідуальні особливості користувачів і часто не адаптовані до різних етнічних груп. У результаті алгоритмічні методи можуть демонструвати формальну коректність, але при цьому давати результати, які не відповідають реальному сприйняттю кольоротипу людини.

Таким чином, алгоритмічні та колориметричні підходи забезпечують необхідний рівень формалізації, проте не здатні повною мірою врахувати складність і багатовимірність поняття кольоротипу, що обмежує їхню ефективність у практичних застосуваннях.

1.4 Сучасні AI-підходи та обґрунтування комбінованого методу

Сучасний етап розвитку систем визначення кольоротипу характеризується активним використанням методів штучного інтелекту, зокрема згорткових нейронних мереж. Такі моделі здатні автоматично вилучати складні, нелінійні ознаки з зображень обличчя та враховувати взаємозв’язок між різними візуальними характеристиками. На відміну від статичних алгоритмів, нейронні мережі не потребують жорстко заданих порогів і можуть адаптуватися до різноманіття вхідних даних.

CNN дозволяють одночасно аналізувати колір шкіри, очей і волосся, а також просторову структуру обличчя, що є критично важливим для комплексного визначення кольоротипу. Додатково застосовуються моделі сегментації, які забезпечують точніше виділення релевантних областей зображення та зменшують вплив фону.

Водночас AI-методи не є універсальним рішенням і мають власні обмеження. Їхня ефективність значною мірою залежить від якості навчальної вибірки та стабільності вхідних зображень. У реальних умовах використання, де користувачі роблять фотографії за різних умов освітлення, без попередньої нормалізації даних результати можуть бути нестабільними та складними для інтерпретації.

Саме тому актуальним є комбінований підхід, який поєднує методи попередньої обробки зображень, спрямовані на зменшення впливу освітлення та артефактів, із можливостями штучного інтелекту щодо високорівневої класифікації. Такий підхід дозволяє підвищити узгодженість результатів, зменшити залежність від умов зйомки та створити більш надійну основу для автоматизованого визначення кольоротипу користувача.

Суттєвою перевагою сучасних AI-методів є їхня здатність працювати з високорівневими абстракціями, які не задаються явно, а формуються в процесі навчання на даних. Це дозволяє нейронним мережам виявляти складні залежності між кольоровими характеристиками різних зон обличчя та їхнім впливом на загальне сприйняття зовнішності. Саме ця властивість робить AI-підходи перспективними для задач, у яких складно сформулювати чіткі аналітичні правила.

Однак використання нейронних мереж без попередньої обробки зображень призводить до зниження стабільності результатів у реальних умовах. Моделі можуть несвідомо навчатися на артефактах освітлення або особливостях конкретних камер, що знижує їхню узагальнювальну здатність. У таких випадках результати класифікації можуть бути формально правильними з точки зору моделі, але нестабільними з позиції користувацького досвіду.

Комбінований підхід, який поєднує методи попередньої нормалізації зображень і AI-класифікацію, дозволяє усунути ці недоліки. Попередня обробка забезпечує зменшення варіативності, не пов’язаної з індивідуальними особливостями користувача, тоді як модель штучного інтелекту зосереджується на аналізі суттєвих ознак. Така архітектура створює баланс між стабільністю класичних методів і гнучкістю нейронних мереж.

З наукової точки зору комбінований підхід також підвищує інтерпретованість результатів, оскільки дозволяє чітко розділити етапи обробки даних та класифікації. Це є важливим аспектом у контексті прикладних систем, де необхідно не лише отримати результат, але й обґрунтувати його коректність.

Отже, використання комбінованого методу визначення кольоротипу є логічним етапом розвитку цієї галузі та створює передумови для побудови більш надійних, адаптивних і масштабованих систем аналізу зовнішності.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ КОЛЬОРОТИПУ ЛЮДИНИ

Визначення кольоротипу людини є важливим завданням у сфері персоналізованої колористики, візажу, стилістики та рекомендаційних систем. Точність цього процесу суттєво впливає на якість підбору кольорової палітри для косметики, одягу та іміджевих рішень. Попри значну кількість практичних підходів, більшість існуючих методів характеризуються суб'єктивністю, нестабільністю та чутливістю до умов освітлення й якості зображення. У цьому розділі здійснено систематизований аналіз наукових і практичних підходів до визначення кольоротипу, зокрема колориметричних правил, методів порівняння зі шкалами кольору та сучасних цифрових тестерів.

2.1 Огляд існуючих методів

2.1.1 Колориметричні правила (статичні алгоритми)

Одним із найпоширеніших підходів до визначення кольоротипу є використання набору ручних, заздалегідь визначених правил, що базуються на аналізі середніх значень RGB/HSV/HSL для шкіри, волосся та очей. Такі алгоритми зазвичай включають:

- встановлення порогових значень (наприклад, світліший відтінок шкіри - "літо/весна", темніший - "осінь/зима");
- логічні умови (теплий підтекст шкіри, світле волосся - «теплий кольоротип»);
- просте нормування кольору без виправлення освітлення;
- евристичні формули, як-от обчислення теплоти через R−B або насиченості через max(R,G,B)−min(R,G,B).

Перевагою цього підходу є простота реалізації, легкість інтерпретації та відсутність вимог до великих обчислювальних ресурсів. Однак статичні алгоритми мають низьку стійкість до реальних умов фотографування — тіні, колірні домінанти,

тональні перекоси можуть суттєво спотворювати результат. Крім того, порогові значення зазвичай підбираються вручну й не можуть охопити різноманіття природних відтінків шкіри.

Колориметричні правила становлять одну з найстаріших і найпоширеніших груп алгоритмів, які використовуються для визначення кольоротипу людини на основі аналізу цифрових зображень. Їх особливістю є те, що вони функціонують виключно на підставі заздалегідь визначених формул і порогових значень, не застосовуючи статистичних моделей чи методів машинного навчання. У таких алгоритмах колірні характеристики шкіри, очей та волосся переводяться у стандартизовані простори, зокрема HSV або CIE L*a*b*, після чого обчислені параметри порівнюються з фіксованими правилами. Завдяки цьому забезпечується повна детермінованість, тобто один і той самий набір вхідних значень завжди приводить до однакового результату.

Основою колориметричних методів є аналіз відтінку, насиченості та яскравості кольору. Перехід до простору HSV дозволяє виділити параметр Hue, який є ключовим у розмежуванні теплих і холодних кольоротипів. Визначення температури кольору здійснюється через позицію значення Hue на колірному колі. Для теплих кольоротипів характерні відтінки в діапазонах червоно-жовтої частини спектра, тоді як холодні типи тяжіють до синьо-фіолетових значень. Після оцінки Hue аналізується насиченість, що допомагає визначити інтенсивність кольору шкіри. Висока насиченість зазвичай пов’язана з яскравими типами зовнішності, тоді як низькі значення характерні для м’яких, пастельних природних палітр. Додатково оцінюється яскравість або світлотність (Value або L*), що дозволяє встановити, наскільки світлим чи темним є загальний колорит обличчя.

HSV в тривимірному просторі у вигляді конусу зображено на **рис. 2.1.**

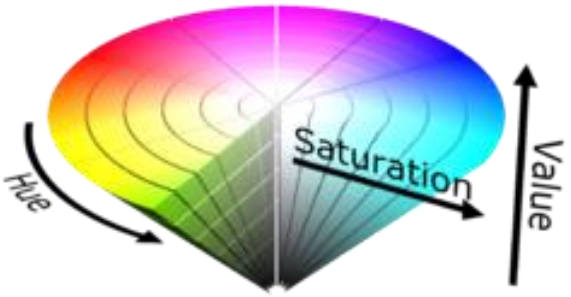


Рис. 2.1 зображення HSV в тривимірний простір у вигляді конусу

Основні принципи роботи статичних (колориметричних) алгоритмів:

1) Аналіз відтінку (Hue).

Аналіз відтінку (Hue) є одним із ключових етапів у колориметричних та комбінованих методах визначення індивідуального кольоротипу людини. Параметр Hue визначає домінантну довжину хвилі світла, що відображається від поверхні шкіри, та описує базовий спектральний тон — від червоних і помаранчевих до синіх і фіолетових відтінків. На відміну від насиченості та яскравості, які можуть істотно варіювати залежно від інтенсивності освітлення чи типу камери, відтінок дає змогу характеризувати саме структурну природу кольору шкіри, що робить його одним із найважливіших показників для визначення теплих або холодних напрямків у класифікації кольоротипів.

У цифровій обробці зображень параметр Hue зазвичай розглядається у просторі HSV, де він представлений у вигляді кута на колірному колі в інтервалі від 0 до 360 градусів. Таке представлення дозволяє безпосередньо інтерпретувати відтінок як позицію кольору в спектрі. Теплі тони шкіри, притаманні весняним і осіннім кольоротипам, зазвичай тяжіють до секторів, що відповідають червоно-оранжевим та жовтим відтінкам (приблизно 0–90° на колірному колі). Натомість холодні кольоротипи — літній і зимовий — частіше демонструють зміщення у напрямку синьо-рожевих та фіолетових компонентів, де відтінок розташований у секторах приблизно 180–260°. Хоча ці межі мають орієнтовний характер і

змінюються залежно від конкретної методики, загальна логіка класифікації залишається сталою: теплі спектральні компоненти асоціюються з високою концентрацією жовтих і червоних пігментів, тоді як холодні — із підвищеною присутністю синьо-фіолетових переважань.

Попри високу цінність відтінку як класифікаційного критерію, обчислення Hue напряму залежить від коректності вихідних даних. Цифрова камера фіксує колір у вигляді значень RGB, які значно чутливі до балансу білого, температури світла та алгоритмів постобробки, що вбудовані у смартфони. Автоматичні системи корекції кольору можуть зрушувати фактичні значення Hue, створюючи відхилення від реального кольору шкіри. Особливо це проявляється у випадках змішаного освітлення, коли на обличчя водночас потрапляють джерела з різною температурою світла. У таких ситуаціях один фрагмент шкіри може інтерпретуватися як теплий, тоді як інший — як холодний, що ускладнює подальшу класифікацію.

Додатковим ускладненням є неоднорідність кольору обличчя. Людська шкіра має природні зміни пігментації в різних областях: щоки частіше більш теплі та насичені, ніж ділянка під очима; лоб може мати іншу світлоту через жирність шкіри; губи та перенісся мають власні колірні домішки, які впливають на середній Hue у вибраній області. Тому для точного аналізу алгоритми зазвичай виділяють спеціальні зони інтересу, в яких відтінок є найбільш репрезентативним — найчастіше це центральна частина щоки або ділянка під вилицею, де шкіра має мінімальний вплив декоративної косметики та менше схильна до локальних змін кольору.

Класичні статичні алгоритми вважають аналіз Hue базовим етапом у визначенні кольоротипу, і результат цього аналізу безпосередньо впливає на подальші висновки. Якщо параметр відтінку вказує на теплий спектр, уся подальша логіка алгоритму зміщується в бік весняних або осінніх палітр. Якщо ж визначено холодний спектр, подальші кроки пов’язані з літніми чи зимовими типами. Однак саме залежність від точності обчислення Hue часто стає причиною помилок. Дослідження показують, що зміна освітлення всього на 10–15% може зрушити

відтінок на 20–30 градусів колірного кола, що призводить до абсолютно іншого результату класифікації навіть при незмінному обличчі.

У сучасних системах аналіз Hue не застосовується ізольовано, а поєднується з додатковими методами компенсації впливів освітлення, зокрема алгоритмами Retinex або нормалізацією у просторі CIE L*a*b*. Такі підходи дають змогу відокремити спектральні характеристики шкіри від зовнішніх факторів і стабілізувати значення відтінку перед етапом класифікації. Таким чином, хоча Hue залишається фундаментальною основою для визначення кольоротипу, його практичне застосування потребує складних корекцій та додаткової обробки, що і пояснює обмеження класичних статичних методів та необхідність їх подальшого вдосконалення.

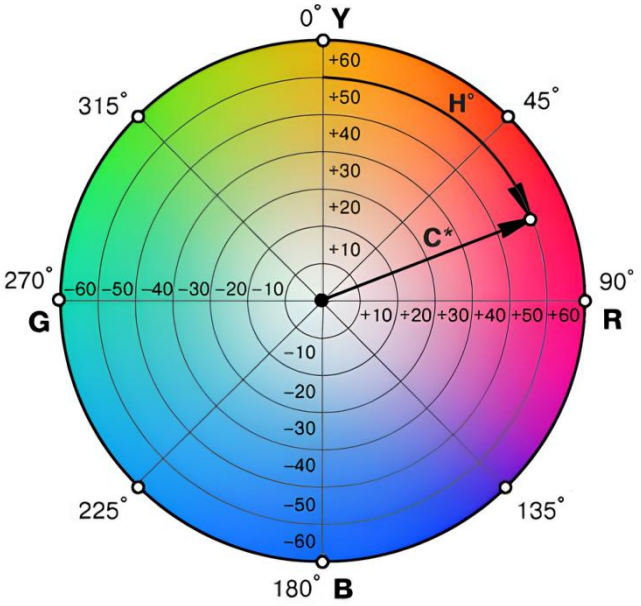


Рис. 2.2 Графік колірного простору CIE L*a*b* у полярних координатах C*H*(6)

2) Насиченість кольору (Saturation) є важливою характеристикою, що визначає інтенсивність, чистоту та виразність відтінку. У контексті визначення кольоротипу насиченість описує ступінь домінування пігменту в шкірі, волоссі та очах. Висока насиченість означає, що колір містить мало домішок сірого та є яскравим і виразним, тоді як низька насиченість відображає більш приглушені, «землисті» або пастельні відтінки. Аналіз S-компоненти особливо важливий у сезонних колірних

системах, оскільки саме рівень насиченості дозволяє розподіляти кольоротипи на «яскраві» та «м'які», «контрастні» або «приглушені».

У цифровій обробці насиченість зазвичай вимірюється у моделях HSV або HSL, де S визначає відносну чистоту кольору щодо його тону. Для обличчя характерною є нерівномірність насиченості в різних зонах — наприклад, щоки часто мають вищу S-складову через рум'янець, тоді як лоб або носогубна зона можуть бути менш насиченими. Тому аналіз насиченості передбачає попередню сегментацію обличчя, після чого значення S усереднюється або обчислюється в кількох локальних областях. Практичне значення насиченості проявляється у спробах віднести людину до кольоротипів «Весна» чи «Зима», які характеризуються високою інтенсивністю кольорів, або до «Осені» та «Літа», де домінують м’які, приглушені палітри. Точність цього аналізу залежить від якості освітлення та наявності артефактів, тому використання корекції освітлення перед вимірюванням S є критично важливим.

3) Яскравість (Value або Lightness) є ключовим параметром, що визначає світлоту кольору шкіри, очей і волосся. У моделі HSV параметр V відображає максимальну інтенсивність світла, яке відбивається пікселем, тоді як у колориметричній моделі CIELAB компонент L* описує світлоту у сприйнятті людини, де 0 відповідає чорному, а 100 — білому. Компонента L* вважається більш точною для наукового аналізу, оскільки вона лінійно корелює з людським сприйняттям світлоти.

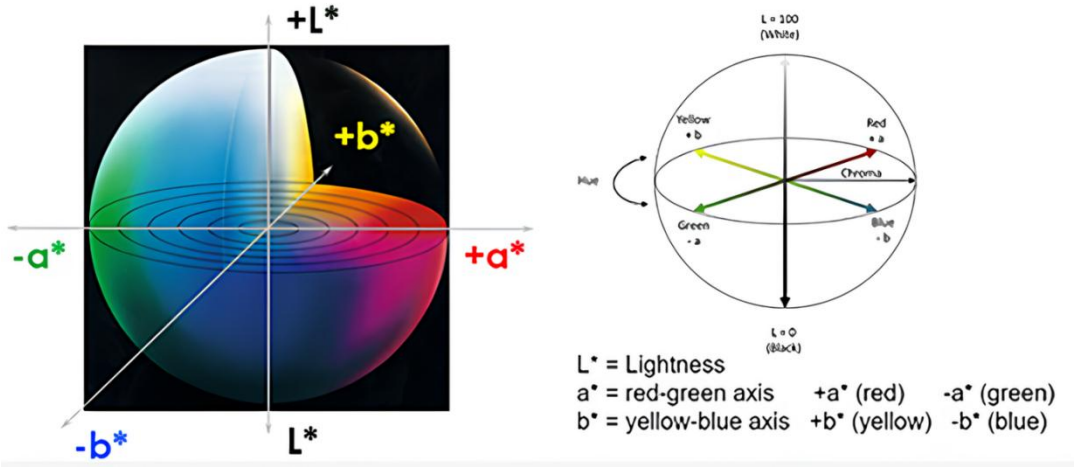


Рис. 2.3 Модель колірного простору CIELAB

Світлота є фундаментальним показником для визначення кольоротипу, оскільки вона дозволяє відрізняти «світлі» та «темні» групи. Наприклад, система 12-сезонної класифікації включає підтипи «Light Summer» і «Light Spring», які характеризуються високими значеннями L*, тоді як «Deep Winter» та «Deep Autumn» мають низькі значення світлоти. Аналіз яскравості також враховує контрастність між різними елементами обличчя — наприклад, темні волосся та світла шкіра формують високий природний контраст, характерний для зимових кольоротипів.

Проблемою аналізу V або L* є сильна залежність від умов освітлення. Навіть незначне зміщення колірної температури або інтенсивності світла може штучно збільшувати або зменшувати світлоту. Тому сучасні алгоритми класифікації використовують попередню нормалізацію освітлення, наприклад Multi-Scale Retinex, щоб зробити вимірювання L* стабільнішим. Після нормалізації значення світлоти стають більш репрезентативними та можуть використовуватися для автоматизованої класифікації кольоротипів.

4) Логічні правила є одним із найдавніших та найпростіших підходів до автоматичного визначення кольоротипу. Цей метод базується на фіксованих порогових значеннях параметрів відтінку, насиченості та яскравості, а також на комбінаціях цих параметрів. Наприклад, якщо значення Hue знаходиться у теплому

діапазоні (наприклад, 20–45° у моделі HSV), насиченість перевищує певний поріг, а світлота належить до верхнього інтервалу, система може віднести користувача до теплого та світлого кольоротипу. Аналогічно, холодні відтінки з високою контрастністю часто класифікуються як зимові.

Логічні правила можуть включати й складніші комбінації, що враховують кілька характеристик одночасно. Наприклад, зіставлення середнього кольору волосся та шкіри допомагає визначити рівень природного контрасту, що є ключовим у сезонній системі. У певних алгоритмах застосовуються також правила пріоритетності, де результати з окремих зон обличчя зважуються по-різному: наприклад, зона щік має більший вплив на визначення теплих/холодних характеристик, тоді як контур очей може краще визначати контрастність.

Головною перевагою логічного підходу є його прозорість та інтерпретованість. Рішення завжди можна пояснити через конкретні пороги та правила. Однак цей підхід має суттєві обмеження. По-перше, він погано адаптується під різні умови зйомки, освітлення та індивідуальні відмінності людей. По-друге, порогові значення часто є умовними й непридатними для всіх типів шкіри. По-третє, логічні правила не враховують неконтрольовані зміни кольору, такі як рум'янець, макіяж або цифрові артефакти. Через це точність статичних порогових алгоритмів зазвичай не перевищує 60–70%, що суттєво нижче, ніж точність систем, які використовують попередню обробку зображення та навчальні моделі штучного інтелекту.

2.1.2 Кольорові карти та тонові шкали

Кольорові карти та тонові шкали є одним із найстаріших і водночас найпоширеніших підходів до оцінювання кольору шкіри та класифікації користувачів за кольоротипами. На відміну від статичних алгоритмів, які працюють з числовими значеннями відтінку, насиченості та яскравості, кольорові карти використовують візуально стандартизовані набори еталонів, з якими порівнюється фактичний колір шкіри людини. Їх основна перевага полягає у наочності: відтінок шкіри співвідноситься із фізичними або цифровими зразками,

що дозволяє виконувати класифікацію навіть без застосування складних математичних моделей.

Об'єктивне вимірювання кольору в комп'ютерному зорі, особливо при аналізі зовнішності людини, вимагає використання стандартизованих еталонів. Кольорові карти та тонові шкали є фундаментальними інструментами, які слугують такими еталонами для забезпечення точності та відтворюваності колірних вимірювань, незалежно від умов зйомки.

Тонова шкала (Gray Scale) являє собою набір нейтральних сірих прямокутників із чітко визначеними значеннями яскравості, які охоплюють динамічний діапазон від абсолютного білого до абсолютного чорного. Її первинне призначення — калібрування яскравості (Luminosity) та оцінка експозиції. Зокрема, 18% середній сірий тон є загальноприйнятим еталоном нейтральної яскравості. У контексті розробки надійного алгоритму визначення кольоротипу, тонова шкала критично важлива для валідації ефективності алгоритму Multi-Scale Retinex (MSR), який є частиною попередньої обробки зображення. MSR забезпечує колірну константність, але точна шкала сірого необхідна для підтвердження того, що яскравісний компонент (L у $L^*a^*b^*$) обробленого зображення приведений до єдиного, стандартизованого рівня.

Кольорова карта, найвідомішим прикладом якої є X-Rite ColorChecker Classic, є більш комплексним еталоном. Вона містить зразки (патчі) ключових кольорів, включно з основними, додатковими, а також відтінками, що представляють типові природні об'єкти, зокрема, різні тони людської шкіри. Кожен патч має точно виміряні колірні координати у стандартизованих колірних просторах, таких як *sRGB* або $L^*a^*b^*$ (див. рис. 2.4).

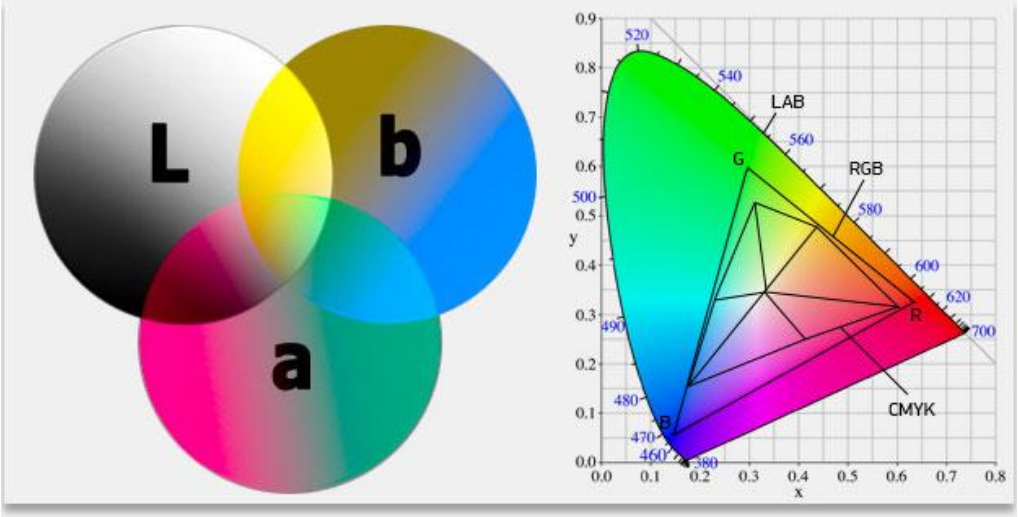


Рис. 2.4 Колірна модель *L*a*b**

Застосування кольорових карт у цифровій обробці дозволяє вирішити дві ключові проблеми, що виникають при аналізі кольоротипу:

1. Автоматична Корекція Балансу Білого: Використання нейтральних сірих зразків на карті дозволяє алгоритму автоматично обчислити та застосувати коефіцієнти посилення для каналів R, G і B, ефективно усуваючи колірний зсув, спричинений штучним або змішаним освітленням.
2. Профілювання Кольору: Порівняння відомих еталонних значень кольорів карти з їхніми фактичними значеннями, зафіксованими камерою, дозволяє створити матрицю перетворення. Ця матриця коригує кольори на всьому зображенні, забезпечуючи їхню максимальну достовірність та незалежність від характеристик пристрою зйомки.

Хоча кінцевий користувач вашого веб-застосунку завантажуватиме звичайні фотографії без еталонів, ці інструменти є критичними для фази розробки, навчання та валідації алгоритму.

Використання стандартизованих еталонів дозволяє кількісно оцінити ефективність блоку попередньої обробки зображення, особливо алгоритму MSR, доводячи, що вимірювання кольору шкіри, волосся та очей у колірному просторі *L*a*b** є інваріантними (незмінними) до умов освітлення. Це є необхідною

передумовою для надійної та об'єктивної роботи фінального класифікатора кольоротипу.

2.1.3 Шкала Фіцпатріка

Шкала Фіцпатріка поділяє типи шкіри на шість категорій (I–VI) на основі реакції на сонячне випромінювання. Хоча система створена для оцінки фототипу, її інколи використовують як основу для класифікації теплоти/холодності шкіри. Fitzpatrick класифікує шкіру за реакцією на ультрафіолетове випромінювання та умовним спектральним тоном, поділяючи людей на шість фототипів — від дуже світлих до темних. Хоча початково шкала не створювалася для вирішення завдань, пов’язаних з підбором кольорів одягу або визначенням сезонного кольоротипу, вона стала одним із широко використовуваних інструментів завдяки простоті інтерпретації. Кожен фототип має набір характерних колірних характеристик — домінуючий теплий або холодний тон, ступінь пігментації та середній спектральний профіль — що дозволяє приблизно визначити напрямок кольоротипу. Проте точність такого підходу є обмеженою, оскільки шкала не враховує вплив індивідуальних особливостей кольору очей, волосся, рум’янців або неоднорідність тону обличчя.

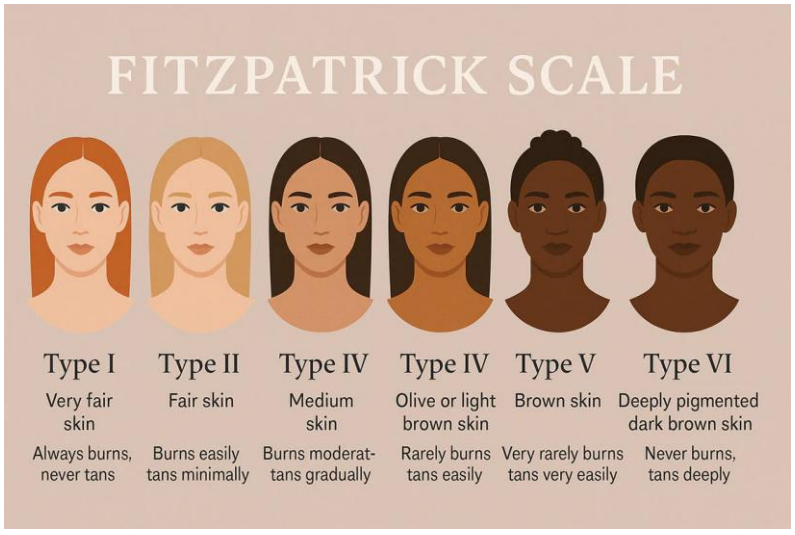


Рис. 2.5 Шкала Фіцпатріка

Переваги:

- проста інтерпретація;
- широко застосовується у медицині;
- наявні відкриті набори еталонів.

Перевагою цієї системи є її простота: вона містить шість основних фототипів, кожен з яких описує середній колір шкіри та реакцію на ультрафіолет. Такий підхід дозволяє швидко виконати класифікацію навіть без спеціальних інструментів. Додатковою перевагою є те, що Fitzpatrick добре відображає глобальні групи кольорів людської шкіри і дає загальне уявлення про тональні особливості, що може бути корисним для первинного аналізу.

Недоліки:

- шкала оцінює реакцію шкіри, а не її колір;
- не призначена для визначення кольоротипу;
- низька точність при роботі з цифровими зображеннями.

Ця шкала була створена не для аналізу кольору візуальних образів, а для оцінки фоточутливості шкіри, тому відтінки всередині фототипів є надто узагальненими. По-друге, вона не враховує складну структуру кольору обличчя: варіації відтінку в різних зонах, пігментні переходи, рум'янець, колір очей чи волосся. По-третє, Fitzpatrick не забезпечує спектрофотометричної точності, тому її використання у цифрових системах обмежене й нестабільне, особливо при різних умовах освітлення.

2.1.4 Система Pantone

Система Pantone містить понад 100 природних відтінків шкіри та забезпечує досить точну відповідність кольорів у стандартизованих умовах.

Переваги:

- професійні стандартизовані зразки;
- висока точність при коректному освітленні;
- можливість підбору косметики у торговельних мережах.

На відміну від Fitzpatrick, який оперує умовними групами, Pantone містить понад 110 високоточних зразків шкіри, виміряних за допомогою спектрофотометрів. Кожен зразок має унікальний цифровий код і визначений набір параметрів: тон, насиченість та світлоту. Це робить систему дуже гнучкою і придатною для точного порівняння кольорів у різних індустріях: стилістика, косметологія, візуальні ефекти, колориметрія тощо.

Головною перевагою Pantone є її об’єктивність і стандартизованість. Вона дозволяє встановити точну відповідність між виміряним кольором шкіри та реальним еталоном, забезпечуючи високу стабільність результатів. Крім того, це одна з небагатьох систем, яка однаково добре працює з усіма типами шкіри — від дуже світлих до дуже темних.

Недоліки:

- дуже чутлива до мінімальних відхилень в освітленні;
- коректно працює тільки з апаратами калібрування;
- не враховує очі та волосся — лише тон шкіри.

Попри переваги, Pantone SkinTone Guide має свої недоліки. Фактичне порівняння потребує або спеціального обладнання, або підготовленого середовища із контрольованим освітленням. Будь-яке відхилення у світловій температурі може змінити візуальне сприйняття кольору. Крім того, система не враховує комплексність кольоротипу, оскільки працює переважно з тоном шкіри, не враховуючи колір очей і волосся, контрастність та інші важливі характеристики, що є критичними у сезонних класифікаціях.

2.1.5 Порівняння з еталонними наборами

Цифрові еталонні набори — це узагальнені палітри, які використовуються у системах сезонного визначення кольоротипу: весна, літо, осінь, зима, а також у розширених варіантах із 12 або 16 підтипами. Вони містять характерні кольори, які відповідають певним тональним групам. Частина цифрових сервісів використовує бібліотеки кольорових шаблонів, що автоматично підбираються через мінімізацію різниці ΔE (CIEDE2000 або CIELAB).

Переваги:

- об’єктивні числові критерії;
- можливість автоматизації.

Перевагою цих методів є їх адаптивність: палітри легко використовувати у цифровому вигляді, а їх порівняння може виконуватися автоматично через комп'ютерне зіставлення кольорових профілів.

Порівняння з еталонними наборами дозволяє оцінити не лише тон шкіри, але й взаємодію цього тону з іншими кольорами, що є важливим аспектом у визначенні гармонійності. Ці методи також можуть включати складні алгоритми зважування різних ділянок обличчя, що підвищує точність порівняння.

Недоліки:

- результат сильно залежить від балансу білого та якості камери;
- сегментація шкіри часто неточна без попередньої обробки.

Недоліком цього підходу є відсутність стандартизації. Різні джерела використовують різні палітри, що призводить до різних результатів. Крім того, такі методи особливо чутливі до якості зображення. Відсутність корекції освітлення, наявність тіней і артефактів на знімку можуть призводити до значного зміщення кольору, що робить результати нестабільними. Більшість таких систем не містять механізмів автоматичної нормалізації освітлення, що критично знижує їх точність.

2.1.6 Сучасні AI-методи та нейронні мережі

За останнє десятиліття конволюційні нейронні мережі (CNN) і суміжні архітектури радикально змінили підхід до обробки зображень: задачі детекції, сегментації та класифікації, які раніше вирішувалися за допомогою поколінь евристик і простих статистичних методів, тепер вирішуються навчальними моделями з набагато вищою точністю та стійкістю. У задачі визначення кольоротипу це дозволяє поєднати аналіз локальних колірних характеристик (тон шкіри, колір очей, волосся) з глобальними візуальними патернами (контрастність, текстура) і при цьому автоматично пристосовуватися до варіацій у даних. Нижче розглянуто три ключових напрями: CNN для аналізу кольору, мультимодальні

моделі, та сучасні методи сегментації (U-Net, Mask R-CNN), з обговоренням практичних деталей навчання, оцінювання та інтеграції у систему.

Конволюційні нейронні мережі є основним інструментом для автоматичної екстракції візуальних ознак із зображень. Для задачі кольоротипу CNN застосовують у двох взаємодоповнюючих режимах: як "енд-ту-енд" класифікатори, що безпосередньо приймають вирівняне зображення обличчя і видають клас кольоротипу, та як фіч-екстрактори, що генерують векторні представлення кольорово-текстурних патернів, які потім аналізуються класичними або навчальними класифікаторами. На практиці для підвищення точності застосовують гібридні підходи: легка CNN (MobileNet, EfficientNet-lite) екстрагує просторово-кольорові ознаки, а додатково використовується вхід з розрахованими статистичними кольоровими ознаками (середні a*, b*, Hue, S, L*, дисперсії) — це дає кращу інтерпретованість і стабільність результату (див. рис. 2.6).

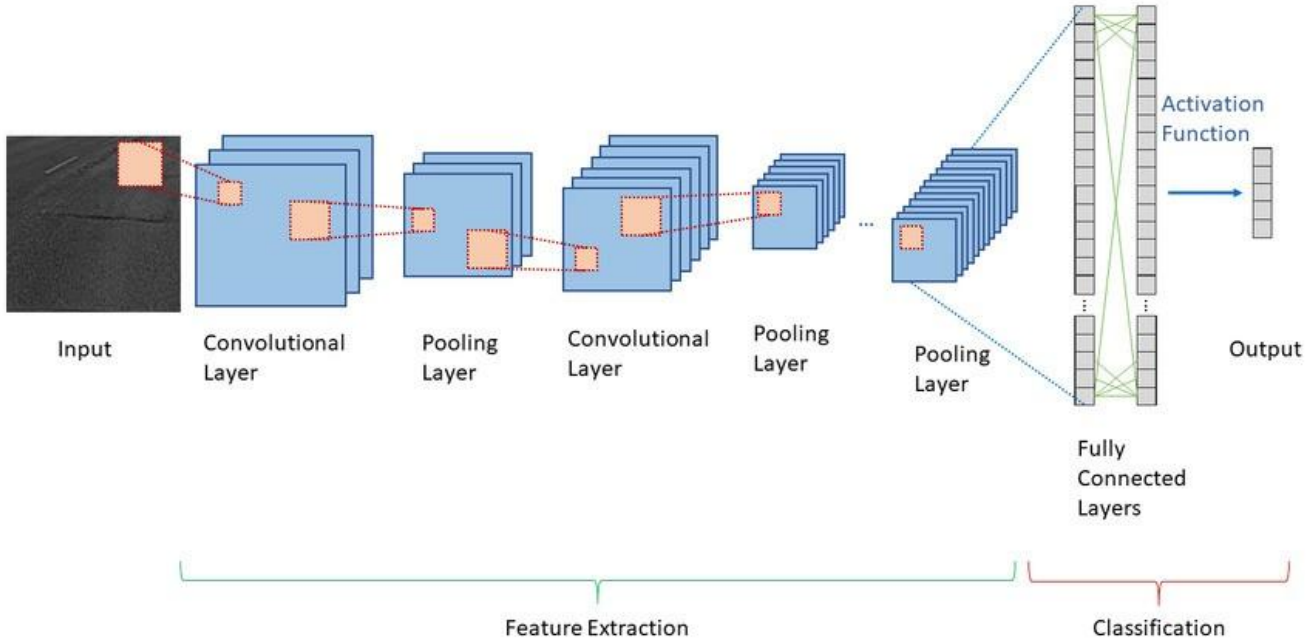


Рис. 2.6 Діаграма архітектури CNN

Ключові архітектурні та практичні моменти включають: вибір backbone (легкий для вбудовування — MobileNet/VGG-lite; потужний для серверного тренування — ResNet/EfficientNet), використання предтренованих на ImageNet ваг (transfer learning) та тонке донавчання на домен-специфічному наборі зображень

людей. Для завдань, де важливі тонкі колірні відмінності, корисно працювати не тільки з RGB, а й з перетвореннями в Lab або YCbCr, подавати на вхід мульти-канальні тензори (наприклад, RGB + Lab L+a+b) або додаткові канали (маска ROI шкіри, маска очей), що дозволяє мережі явніше розрізняти колірні компоненти від освітлення.

Якість та різноманітність датасету визначають, чи CNN зможе навчитися узагальнювати. Обов'язковими є аугментації, орієнтовані на колір: регулювання експозиції, зміщення балансу білого, накладання локальних тіней і бликів, а також геометричні аугментації (повороти, кропи). Додаткові техніки — змішування зображень (mixup), експоненційне усереднення ваг, та data-balancing для рідкісних класів (oversampling або class weights). Метрики для оцінки мають виходити за межі простої accuracy: потрібні per-class Precision/Recall/F1, Confusion Matrix (для виявлення систематичних спотворень між, наприклад, «Весна» й «Осінь»), а також robustness-метрики (зміна accuracy при навмисній деградації освітлення).

Мультимодальні моделі для класифікації кольоротипів**.** Мультимодальні моделі поєднують різні джерела інформації: зображення (візуальна інформатика), структуровані кольорові характеристики (статистичні ознаки ROI), метадані (тип камери, умови освітлення, наявність макіяжу), іноді текстові описи або відповіді користувача. Такі моделі зазвичай мають дві або більше гілок: візуальна гілка (CNN) для обробки піксельних даних і таблична/сигнальна гілка (MLP або трансформерні підмережі) для обробки числових або категорійних ознак. Поєднання фіч від різних модальностей (concatenate або cross-attention) дозволяє системі робити більш інформоване рішення — наприклад, якщо статистичні метрики виявляють низьку впевненість через сильні тіні, CNN може надати додаткові контекстні патерни для компенсації.

Мультимодальні підходи також відкривають шлях до умовної логіки виводу: модель може навчитися керувати увагою до тих областей, що більш релевантні в конкретному знімку (наприклад, ігнорувати інформацію з ділянки, де помічений макіяж). Використання механізмів уваги (attention) або gating-механізмів дозволяє

автоматично зважувати внесок кожної модальності залежно від якості сигналу, істотно підвищуючи стійкість у реалістичних сценаріях.

Для навчання мультимодальних моделей потрібні збалансовані датасети з багатими анотаціями: маски ROI, маркери макіяжу, метадані камери тощо. Також корисні стратегії багатозадачного навчання (multi-task learning), де мережа одночасно навчається сегментувати шкіру/очі/волосся та класифікувати кольоротип — це підвищує узагальнювальну здатність та робить архітектуру більш інтерпретованою.

Методи сегментації (U-Net, Mask R-CNN)**.** Точна сегментація областей шкіри, райдужки ока та волосся — фундаментальна умова коректного вимірювання кольорових характеристик. U-Net і Mask R-CNN — дві найпоширеніші архітектури для семантичної та інстанс-сегментації відповідно. U-Net є ефективною для семантичної сегментації ROI: її U-подібна симетрична архітектура з шунтами (skip connections) дозволяє зберегти деталі при ієрархічному узагальненні інформації. U-Net легко тренувати на відносно невеликих наборах даних і вона дає високу просторову точність, що необхідно для виділення ділянок шкіри, де колір не спотворений макіяжем чи тінями.

Mask R-CNN надає інстанс-маски для об’єктів (окремо — кожне око, кожна пасма волосся тощо) і поєднує регіональні пропозиції (RPN) з піковою сегментацією. Це корисно, коли потрібно диференціювати множинні об’єкти (наприклад, очі як окремі інстанси) або коли волосся має складну структуру і накриває частини обличчя. Mask R-CNN добре інтегрується у пайплайни, де потрібно одночасно робити детекцію та сегментування.

На рисунку 2.7 зображено модель комп’ютерного зору Mask R-CNN, що призначена для виявлення об’єктів і сегментація екземплярів.

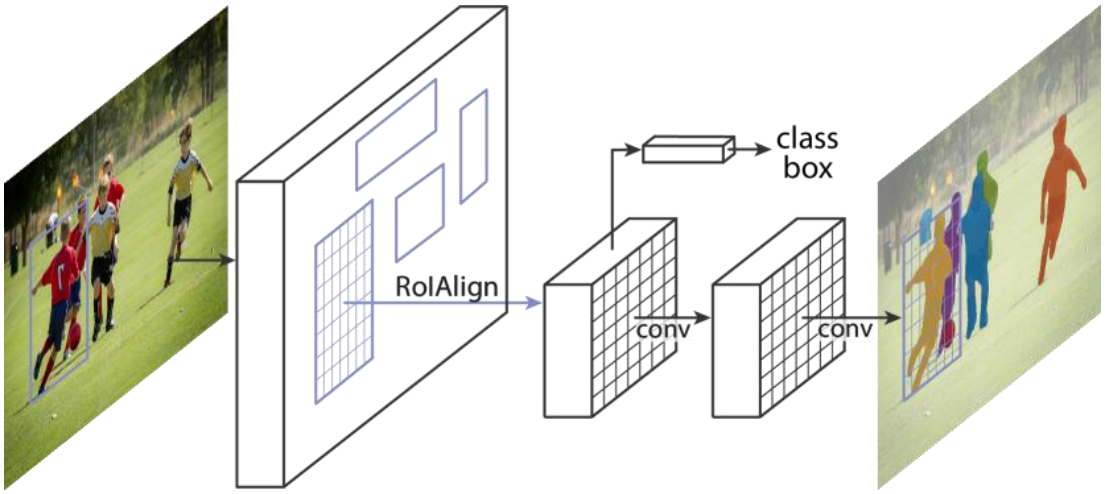


Рис. 2.7 Mask R-CNN

Практичною стратегією є поєднання: спочатку швидка U-Net (див. рис. 2.8) для грубої сегментації шкіри та волосся, потім Mask R-CNN для уточнення інстанс-масок у складних зонах. Важливим елементом є розробка якісних анотацій: для навчання сегментаційної моделі потрібні точні маски шкіри/очей/волосся; інакше модель навчиться фільтрувати некоректні патерни. Також критично використовувати loss-компоненти, що враховують баланс класів (Dice loss, focal loss) через велике домінування фонового класу.

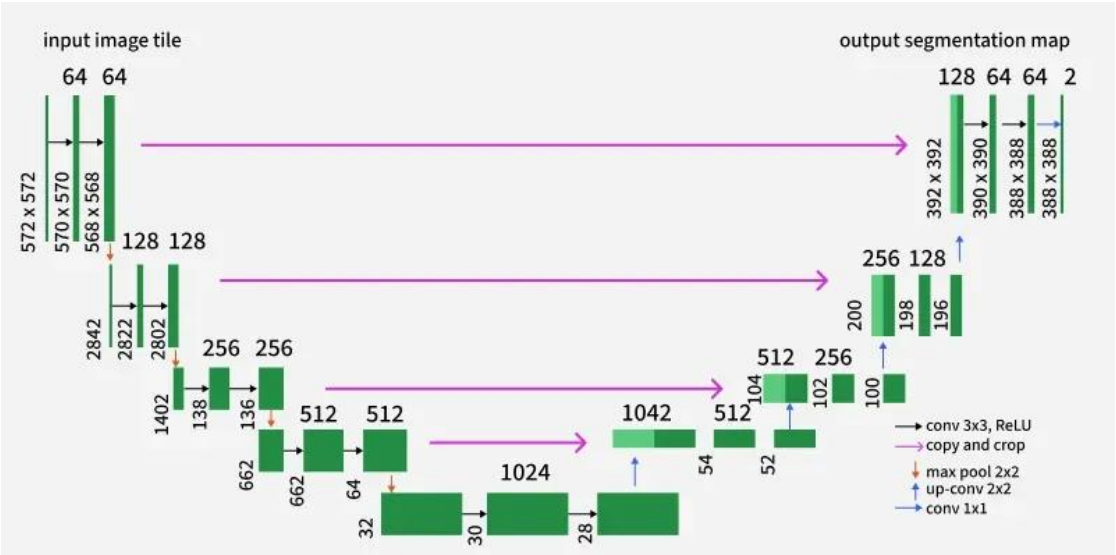


Рис. 2.8 Діаграма архітектури U-Net

Порівняння моделей за точністю, швидкістю та вимогами до даних. При виборі архітектури слід зважувати три взаємопов’язані фактори: точність (accuracy, F1, per-class metrics), швидкість (inference latency на цільовій платформі, throughput) та вимоги до даних (обсяг анотованих прикладів, якість масок, наявність метаданих). Потужні серверні backbones (ResNet-101, EfficientNet-B4/B7) зазвичай демонструють вищу точність, особливо на великих наборах даних, але вимагають більше обчислювальних ресурсів та пам’яті. Легкі backbones (MobileNetV2, EfficientNet-lite) забезпечують кращий баланс для мобільного деплою: прийнятна точність при низькій латентності. U-Net дає високу якість сегментації при помірних вимогах до даних, Mask R-CNN вимагає більше анотованих bbox/mask та довшого часу тренування, але забезпечує гнучкість в інстанс-сегментації.

У табличному вигляді порівняння виглядає приблизно так: серверні CNN (ResNet-class) — точність висока, latency велика, дані — великі; мобільні CNN — точність середня-висока, latency мала, дані — середні; U-Net — точність сегментації висока, latency середній, потреба в масках середня; Mask R-CNN — інстанс-сегментація висока, latency велика, потреба в детальних масках висока. Важливо також вимірювати робочу стійкість: robust accuracy (стан після аугментацій, шумів), що часто найбільше поліпшується завдяки мультимодальності та сегментаційній попередній обробці.

Важливо зазначити, що AI-методи дають кілька фундаментальних переваг. По-перше, вони вчаться на даних і тому можуть підлаштовуватися під реальні варіації освітлення, різні камери та наявність макіяжу — речі, які статичні правила не враховують. По-друге, моделі можуть поєднувати різні джерела інформації і робити умовні висновки (наприклад, нехтувати інформацією з області, де виявлено сильний макіяж). По-третє, глибинні методи набагато краще уявляють складні нелінійні залежності між кольоровими компонентами і контекстними патернами (текстура шкіри, характерні відблиски) і тому забезпечують вищу загальну точність та стійкість. Нарешті, AI-методи дозволяють будувати системи з вимірюваною довірчою оцінкою (confidence), застосовувати механізми активного навчання та

онлайн-адаптації (онлайн fine-tuning під конкретного користувача), чого не можуть статичні правила.

Проте AI-підходи мають власні виклики: потреба в якісних і широких анотаціях, ризик упередженості, складність інтерпретації рішень (хоча методи explainable AI значно просунулися), та інфраструктурні вимоги для тренування і деплою. У підсумку найкраща практична стратегія — комбінувати: використовувати сегментаційну AI-попередню обробку і мультимодальний класифікатор, що приймає також інтерпретовані колориметричні ознаки; така архітектура максимально поєднує переваги статистичної інтерпретованості та адаптивності .

2.2 Порівняння існуючих методів для визначення кольоротипу

Визначення кольоротипу людини є багатофакторною задачею, яка історично ґрунтувалася на суб'єктивній оцінці стилістів. Із розвитком комп'ютерного зору та AI, з'явилася низка спроб автоматизувати цей процес. Аналіз існуючих методологій показує, що вони належать до кількох категорій: від статичних колориметричних правил та використання еталонних колірних шкал до сучасних систем, заснованих на машинному навчанні.

Жоден із наявних методів не є повністю ефективним для надійного та об'єктивного аналізу в неконтрольованих умовах (наприклад, фото, зроблене користувачем у домашніх умовах). Ключовим недоліком залишається критична чутливість до умов освітлення, що призводить до спотворення реальних колірних характеристик шкіри, волосся та очей. Колориметричні та еталонні методи не мають вбудованого механізму корекції світла, а сучасні AI-системи часто ігнорують або недостатньо ефективно вирішують проблему попередньої нормалізації зображення.

Наступна таблиця узагальнює основні переваги та недоліки найбільш поширених методик, обґрунтовуючи, чому вони є недостатніми для створення точного та відтворюваного веб-застосунку для визначення кольоротипу.

Порівняння ключових переваг і недоліків розглянутих моделей подано у таблиці 2.1.

Таблиця 2.1.

Порівняння існуючих методів для визначення кольоротипу

Назва методу	Переваги	Недоліки
Колориметричні Правила	Швидкість, простота реалізації.	Критична чутливість до освітлення. Низька точність у реальних умовах зйомки.
Кольорові Карти та Тонові Шкали	Об'єктивність, ідеальні для калібрування алгоритмів освітлення (MSR).	Непридатні для онлайн-використання (потрібна фізична карта).
Шкала Фіцпатрика	Простота, широко використовується в медицині.	Визначає лише фототип (реакцію на УФ), а не кольоротип. Суб'єктивність оцінки.
Pantone SkinTone Guide	Надає стандартизований колірний простір.	Вимагає високоточного вимірювання (спектрофотометр). Це каталог, а не алгоритм класифікації.
Порівняння з Еталонними Наборами	Імітує роботу стиліста (драпірування).	Критична чутливість до освітлення та пристрою відображення.
Сучасні AI-Методи та Нейронні Мережі	Висока точність, здатність обробляти багато ознак.	Висока обчислювальна складність. Чутливість до ненормалізованих вхідних даних.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕТОДУ ВИЗНАЧЕННЯ КОЛЬОРОТИПУ

У цьому розділі детально описано проєктування й реалізацію методу автоматичного визначення кольоротипу на основі фотографії обличчя. Наведено архітектуру обробного конвеєра, алгоритми попередньої обробки, правила виділення областей інтересу (ROI), процедури екстракції колірних ознак, структуру моделі штучного інтелекту, підготовку даних і методику навчання та валідації. Мета — створити стійкий, відтворюваний і ефективний конвеєр, що забезпечує коректну класифікацію кольоротипу в реалістичних умовах.

Архітектура розробленого методу ґрунтується на послідовному виконанні чотирьох основних етапів: завантаження зображення, попередня обробка, сегментація ключових областей та класифікація кольоротипу за допомогою моделі штучного інтелекту. Така структурована послідовність забезпечує надійність, відтворюваність та високу точність результатів незалежно від умов зйомки чи особливостей користувача. Нижче подано детальний опис кожного етапу.

3.1 Архітектура системи

Система Color Muse організована за принципом багаторівневої архітектури (Multi-tier Architecture), що забезпечує чітке розділення відповідальностей між різними частинами системи. Перший рівень представлення (Presentation Layer) реалізований як клієнтський веб-застосунок на базі React, який відповідає за відображення користувацького інтерфейсу та взаємодію з користувачем. Другий рівень бізнес-логіки (Business Logic Layer) представлений серверним API на базі Node.js та Express, що обробляє запити від клієнта та координує взаємодію між різними компонентами системи. Третій рівень обробки даних (Data Processing Layer) включає модуль обробки зображень на Python, який виконує складні операції комп'ютерного зору. Четвертий рівень зберігання даних (Data Storage

Layer) реалізований через базу даних MongoDB для зберігання інформації про користувачів. П'ятий рівень зовнішніх сервісів (External Services Layer) включає сервіс штучного інтелекту Ollama для класифікації колористичного типу. Така організація забезпечує розділення відповідальностей (Separation of Concerns), що сприяє підтримуваності, масштабованості та тестованості системи.

На рисунку 3.1 розглянемо UML-діаграму компонентів системи.

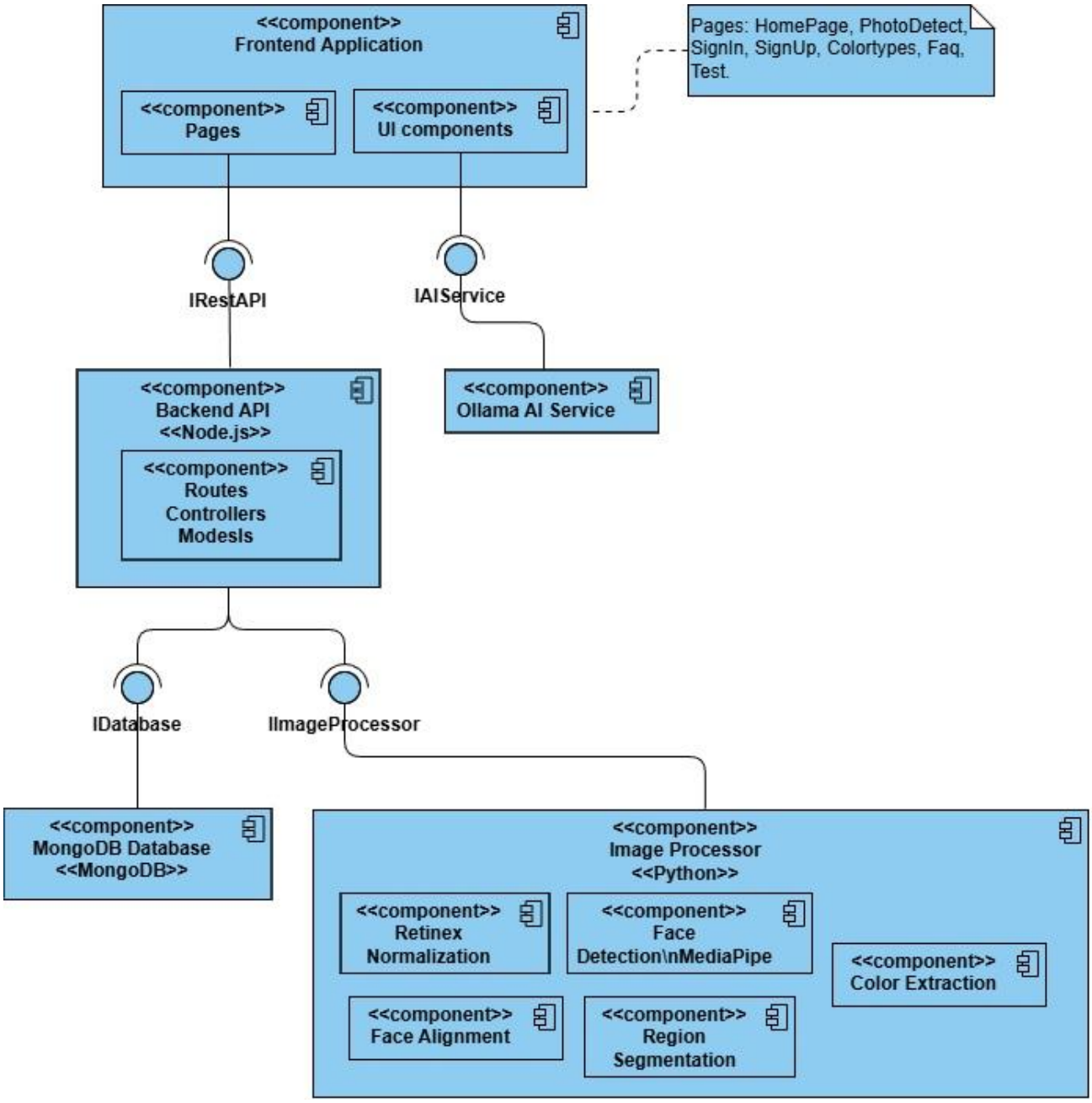


Рис. 3.1 UML-діаграма компонентів системи

3.1.1 React Application

React Application є основним компонентом клієнтської частини системи, реалізованим на базі бібліотеки React версії 19.1.0. Цей компонент відповідає за відображення користувацького інтерфейсу, обробку користувацьких дій, комунікацію з серверним API та управління станом додатку. Компонент структурований за принципом компонентної архітектури, що дозволяє створювати перевикористовувані та модульні елементи інтерфейсу. Технологічний стек включає React 19.1.0 для побудови користувацьких інтерфейсів, React Router 7.5.3 для маршрутизації між сторінками, Material-UI 7.1.0 як бібліотеку UI компонентів, Axios 1.9.0 як HTTP клієнт для взаємодії з API, та Framer Motion 12.12.1 для створення анімацій та плавних переходів між станами інтерфейсу.

3.1.2 Pages Component

Компонент Pages інкапсулює всі сторінки веб-додатку та реалізує функціональність різних розділів системи. Кожна сторінка є окремим React-компонентом, що відповідає за конкретний функціонал та забезпечує ізольовану логіку відображення та обробки даних

HomePage є головною сторінкою додатку, яка служить точкою входу для користувачів. На цій сторінці представлена загальна інформація про систему, її можливості та навігація до основних розділів, що дозволяє користувачам швидко орієнтуватися в функціоналі додатку та переходити до потрібних розділів.

PhotoDetect є ключовим компонентом системи, що реалізує основну функціональність – завантаження та обробку фотографії обличчя для визначення колористичного типу. Ця сторінка забезпечує інтерфейс для завантаження зображення, вибору методу обробки (базовий або розширений) та відображення результатів аналізу. Функціональність включає завантаження фотографії користувача, вибір методу обробки, відправку зображення на обробку, відображення результатів класифікації колористичного типу та візуалізацію обробленого зображення при виборі розширеного методу. Компонент взаємодіє з

Backend API для попередньої обробки зображення та з Ollama AI напряму для класифікації колористичного типу, що забезпечує гнучкість у виборі методу аналізу.

Компоненти SignIn та SignUp реалізують функціональність аутентифікації та авторизації користувачів у системі. SignIn забезпечує форму входу з полями email та password, валідацію вхідних даних, відправку запиту на сервер для аутентифікації, збереження JWT токену в локальному сховищі та перенаправлення на головну сторінку після успішного входу. SignUp реалізує форму реєстрації з полями ім'я, email, password та стать, валідацію вхідних даних, відправку запиту на сервер для створення нового облікового запису та автоматичний вхід після успішної реєстрації.

Компонент Colortypes надає інформаційний контент про різні колористичні типи, що використовуються в системі. Сторінка містить описи та характеристики дванадцяти колористичних типів, включаючи bright_spring, warm_spring, light_spring, light_summer, cool_summer, soft_summer, warm_autumn, deep_autumn, soft_autumn, bright_winter, cool_winter та deep_winter. Функціональність включає відображення описів колористичних типів, візуальні приклади для кожного типу та навігацію між типами, що дозволяє користувачам детально ознайомитися з характеристиками кожного колористичного типу.

Компонент Faq (Frequently Asked Questions) містить відповіді на часті питання користувачів щодо використання системи, методів визначення колористичного типу та інших аспектів роботи додатку, що спрощує користувачам розуміння функціоналу та вирішення можливих проблем.

Компонент Test реалізує інтерактивний тест для визначення колористичного типу на основі відповідей користувача на питання про колір очей, волосся, шкіри та переваги в одязі. Тест використовує простий алгоритм підрахунку балів для визначення найбільш підходящого типу серед чотирьох основних категорій: Winter, Summer, Autumn та Spring. Функціональність включає послідовне відображення питань з варіантами відповідей, збереження відповідей користувача, розрахунок результату на основі алгоритму підрахунку та відображення результату

з описом та візуальним прикладом, що надає альтернативний спосіб визначення колористичного типу без використання фотографії.

Компонент UserProfile призначений для відображення та редагування профілю користувача. На момент реалізації Frontend інтерфейс для цього компонента не реалізований, однак Backend API повністю готовий до використання. Планується функціональність включає перегляд інформації профілю користувача, редагування особистих даних, управління улюбленими колористичними типами та історію аналізів, що дозволить користувачам зберігати та керувати своїми результатами.

3.1.3 UI Components

Компонент UI Components інкапсулює перевикористовувані елементи інтерфейсу, що використовуються на різних сторінках додатку. Компонент Header реалізує навігаційну панель, яка присутня на всіх сторінках додатку та містить логотип системи, посилання на основні розділи, включаючи Головну,

Колористичні типи, FAQ, Тест та Завантажити фото, а також елементи управління сесією користувача, такі як кнопка входу для неавторизованих користувачів та кнопка виходу для авторизованих. Функціональність включає навігацію між сторінками, відображення стану авторизації користувача та управління сесією, що забезпечує зручну орієнтацію в додатку та швидкий доступ до основних функцій.

Компонент Footer відображає підвал сторінки з додатковою інформацією, контактами та посиланнями, що надає користувачам додаткову інформацію про систему та способи зв'язку.

Компонент Animated Components включає набір анімованих елементів інтерфейсу, що покращують користувацький досвід. До них належать AnimatedCounter для анімованого лічильника, AnimatedTypingText для тексту з ефектом друку, AnimatedFadedDownText для тексту з анімацією появи, AnimatedHorizontalBox для горизонтальної анімації та AnimatedUnderlineText для підкресленого тексту з анімацією. Ці компоненти використовують бібліотеку

Framer Motion для створення плавних анімацій та переходів, що робить інтерфейс більш динамічним та приємним для користувача.

3.1.4 React Router

Компонент React Router забезпечує маршрутизацію в односторінковому додатку (SPA – Single Page Application). Він відповідає за відображення відповідного компонента сторінки залежно від поточного URL. Система використовує наступні маршрути: кореневий маршрут `/` відображає HomePage, `/photo-detect` відповідає за PhotoDetect, `/signin` та `/signup` відповідають за сторінки входу та реєстрації, `/colortypes` відображає інформацію про колористичні типи, `/faq` показує сторінку з часто задаваними питаннями, а `/test` відповідає за інтерактивний тест. Така організація маршрутизації забезпечує зручну навігацію та швидкий доступ до різних розділів додатку без повного перезавантаження сторінки.

3.1.5 Express API

Express API є основним компонентом серверної частини системи, реалізованим на базі Node.js та фреймворку Express версії 4.18.2. Цей компонент реалізує RESTful API для обробки запитів від клієнтського застосунку. Призначення компонента включає обробку HTTP запитів від клієнта, валідацію вхідних даних, виконання бізнес-логіки, управління доступом до ресурсів та координацію взаємодії з базою даних та зовнішніми сервісами. Серверна частина організована за патерном MVC (Model-View-Controller), де Routes визначають точки входу API, Controllers містять бізнес-логіку, а Models описують структуру даних. Така організація забезпечує чітке розділення відповідальностей та спрощує підтримку та розширення системи.

3.1.6 Controllers Component

Компонент Controllers містить бізнес-логіку обробки запитів та координацію взаємодії між різними компонентами системи. AuthController реалізує логіку

аутентифікації та авторизації через методи register, login та logout. Метод register обробляє реєстрацію нового користувача, включаючи валідацію вхідних даних, хешування пароля з використанням bcrypt та salt, створення нового запису в базі даних та повернення створеного користувача. Метод login обробляє вхід користувача через пошук користувача за email, перевірку пароля, генерацію JWT токену з терміном дії один день, встановлення токену в HTTP-only cookie та повернення токену та інформації про користувача. Метод logout обробляє вихід користувача через очищення cookie з токеном.

PhotoController реалізує логіку обробки фотографій через метод processPhotoWithPreprocessing, який обробляє фотографію з попередньою підготовкою. Процес включає валідацію наявності зображення у запиті, виклик Python скрипту `image_processor.py` через child process, передачу зображення у форматі base64 через stdin, отримання результату обробки через stdout, парсинг JSON відповіді та обробку помилок та повернення результату. Особливості реалізації включають використання `child_process.exec()` для асинхронного виконання Python скрипту, буфер розміром 10MB для обробки великих зображень та обробку помилок виконання Python скрипту.

UserController реалізує логіку управління користувачами та їх профілями через набір методів. Метод getCurrentUser отримує профіль поточного користувача через декодування JWT токену з заголовка Authorization, пошук користувача в базі даних за ID з токену та повернення інформації про користувача. Метод putCurrentUser оновлює профіль поточного користувача через декодування JWT токену, оновлення полів профілю, таких як favorites, та збереження змін в базі даних. Додаткові методи включають getUsers для отримання списку всіх користувачів, getUser для отримання інформації про користувача за ID, postUser для створення нового користувача, putUser для оновлення інформації про користувача та deleteUser для видалення користувача.

3.1.7 Image Processor

Image Processor є окремим модулем обробки зображень, реалізованим на мові Python. Цей компонент виконується як окремий процес, викликаний з Backend через child process, що дозволяє використовувати потужні бібліотеки обробки зображень Python, такі як OpenCV та MediaPipe, у Node.js середовищі. Призначення компонента включає попередню обробку фотографій обличчя для підвищення точності аналізу, нормалізацію освітлення, виявлення та вирівнювання обличчя, сегментацію регіонів обличчя та екстракцію домінантних кольорів. Технологічний стек включає Python 3.x як мову програмування, OpenCV (cv2) як бібліотеку комп'ютерного зору, MediaPipe як бібліотеку для виявлення обличчя та landmarks та NumPy як бібліотеку для роботи з масивами. Протокол взаємодії передбачає вхід у вигляді JSON зображення у форматі base64 через stdin, вихід у вигляді JSON з результатами обробки через stdout та виконання як асинхронний child process через `exec()`.

3.1.8 Етапи обробки зображення

Retinex Normalization реалізує багатомасштабний алгоритм Retinex для нормалізації освітлення на зображенні. Цей алгоритм зменшує вплив тіней та бліків, що може значно впливати на точність аналізу кольорів. Алгоритм включає конвертацію зображення в формат з плаваючою точкою, застосування гаусового розмиття для різних масштабів з параметрами sigma рівними 15, 80 та 250, обчислення логарифмічної різниці між оригіналом та розмитим зображенням, усереднення результатів для всіх масштабів та нормалізацію до діапазону 0-255.

Результатом є зображення з нормалізованим освітленням, що краще підходить для аналізу кольорів, оскільки зменшує вплив зовнішніх факторів освітлення.

Face Detection використовує MediaPipe Face Mesh для виявлення обличчя на зображенні та визначення ключових точок (landmarks). Функціональність включає виявлення обличчя на зображенні, визначення 468 ключових точок обличчя та екстракцію важливих регіонів, таких як очі (left_eye, right_eye), ніс (nose_tip), рот

(mouth_left, mouth_right), підборіддя (chin) та чоло (forehead). Результатом є словник координат ключових точок обличчя, що дозволяє точно визначити положення різних частин обличчя для подальшої обробки.

Face Alignment вирівнює обличчя на зображенні, обертаючи його так, щоб очі були на одній горизонтальній лінії. Це забезпечує стандартизовану орієнтацію обличчя для подальшого аналізу. Алгоритм включає обчислення центрів лівого та правого ока, розрахунок кута обертання між очами, створення матриці афінного перетворення, застосування обертання до зображення та застосування того ж обертання до всіх landmarks. Результатом є вирівняне зображення обличчя та оновлені координати landmarks, що забезпечує консистентність аналізу незалежно від орієнтації обличчя на оригінальному зображенні.

Region Segmentation створює маски для різних регіонів обличчя, що дозволяє виділити та аналізувати кожен регіон окремо. Skin Mask використовує landmarks для визначення області шкіри обличчя, виключаючи очі, рот та інші регіони. Eye Mask використовує landmarks очей для точного виділення області очей. Hair Mask використовує landmarks чола та підборіддя для визначення області волосся. Результатом є бінарні маски для кожного регіону, що дозволяють точно виділити та проаналізувати кольори кожної частини обличчя окремо.

Color Extraction витягує домінантні кольори з кожного регіону обличчя за допомогою алгоритму K-means кластеризації. Алгоритм включає застосування маски до зображення для виділення регіону, витягнення пікселів регіону, застосування K-means кластеризації для групування кольорів, визначення домінантних кольорів як центроїдів кластерів та конвертацію кольорів у формат RGB. Параметри кластеризації включають k=5 кластерів для шкіри, k=3 кластери для очей та k=4 кластери для волосся. Результатом є масиви домінантних RGB кольорів для кожного регіону, що надає точну інформацію про кольорову палітру обличчя для подальшого аналізу та класифікації.

3.1.9 MongoDB Database

MongoDB Database є документоорієнтованою NoSQL базою даних, що використовується для зберігання інформації про користувачів системи. Призначення включає зберігання даних користувачів, забезпечення швидкого доступу до даних та підтримку індексів для оптимізації запитів. Колекція users містить документи користувачів зі структурою, що включає поле `_id` як ObjectId для унікальної ідентифікації, name як рядкове поле для імені користувача, email як рядкове поле для електронної пошти, password як рядкове поле з bcrypt хешем пароля, isFemale як булеве поле для статі користувача та favorites як масив улюблених колористичних типів. Індекси включають унікальний індекс на полі email для забезпечення унікальності email адрес. Оптимізація включає використання індексів для швидкого пошуку за email та автоматичну генерацію ObjectId для унікальної ідентифікації. Зв'язок з Backend реалізований через використання Mongoose ODM для роботи з базою даних, автоматичну валідацію даних на рівні моделі та підтримку транзакцій та обробку помилок.

3.1.10 Ollama AI Service

Ollama AI Service є зовнішнім сервісом штучного інтелекту, що використовується для класифікації колористичного типу на основі обробленого або оригінального зображення. Призначення включає аналіз зображення обличчя, класифікацію колористичного типу та повернення результату у вигляді назви типу. Технічні характеристики включають модель gemma3:4b з 4 мільярдами параметрів, API Endpoint за адресою http://localhost:11434/api/generate, протокол HTTP POST та формат даних JSON. Вхідні дані включають об'єкт з полями model зі значенням "gemma3:4b", prompt з промптом для аналізу, stream зі значенням false для отримання повної відповіді та images як масив base64 закодованих зображень. Вихідні дані включають об'єкт з полем response, що містить назву колористичного типу, наприклад "bright_spring". Функціональність включає аналіз зображення обличчя з урахуванням кольорів шкіри, очей та волосся, визначення найбільш підходящого колористичного типу з дванадцяти можливих варіантів та повернення

результату у текстовому форматі. Взаємодія з системою реалізована через прямий виклик Ollama AI з Frontend через HTTP POST без використання Backend як проміжного шару, що дозволяє працювати як з оригінальним, так і з обробленим зображенням. Переваги прямого з'єднання включають зменшення навантаження на Backend, швидшу обробку завдяки меншій кількості проміжних шарів та незалежність від стану Backend сервера.

3.2 Нормалізація освітлення

Нормалізація освітлення є одним із ключових етапів попередньої обробки зображень, оскільки саме освітлення найбільше впливає на коректність визначення кольоротипу. Зміни яскравості, наявність тіней, перепади світла або кольорові рефлекси можуть суттєво спотворювати реальні кольорові характеристики шкіри, волосся та очей. Щоб мінімізувати ці впливи, у розробленому методі використовується багатомасштабний алгоритм Multi-Scale Retinex (MSR), який поєднує можливості логарифмічного перетворення та просторової фільтрації.

Алгоритм Retinex — це класична модель, заснована на властивостях людського зору, зокрема на здатності адаптуватися до різних умов освітлення. Назва походить від комбінації слів *retina* та *cortex*, що підкреслює ідею моделювання обробки сигналів як оком, так і мозком.

Основна мета Retinex — розділити зображення на дві складові:

- Reflectance (відбиття) — властивість об’єкта, що визначається його природним кольором;
- Illumination (освітлення) — зовнішній світловий вплив.

Формально це можна описати як:

$$I(x, y) = R(x, y) \cdot L(x, y), \quad (3.1)$$

де:

$I(x,y)$ - Це вхідне зображення або його окремий колірний канал (RGB), яке фіксується сенсором (камерою) у координаті (x,y). Це те, що ми бачимо: сприйнята яскравість/колір.

$R(x,y)$ - Це властивість об'єкта у координаті (x,y). R є інваріантною до освітлення складовою, тобто відображає природний колір об'єкта. Саме цю складову Ви намагаєтеся виділити для визначення кольоротипу.

$L(x,y)$ - Це зовнішній вплив у координаті (x,y). L відображає інтенсивність та спектральний склад джерела світла, а також нерівномірності (тіні, перепади яскравості). Це шумова складова, яку алгоритм MSR прагне мінімізувати.

Для мінімізації впливу різних умов освітлення застосовується багатомасштабний алгоритм Multi-Scale Retinex (MSR).

Алгоритм працює так:

- створюються три розмиті копії зображення за допомогою Gaussian Blur з різними значеннями σ (15, 80, 250);
- для кожного масштабу обчислюється Retinex-компонента за логарифмічною формулою;
- результати усереднюються;
- отримане зображення додатково нормалізується до діапазону [0–255].

Таке попереднє освітлення дозволяє зменшити тіні, блиски, перепади освітленості та зробити кольори шкіри більш природними й стабільними для аналізу.

Класичні варіанти Retinex працюють із одним масштабом Gaussian-фільтра, що не дозволяє повністю компенсувати локальні й глобальні перепади освітлення одночасно. Тому у сучасних підходах застосовується багатомасштабна версія, яка працює одразу з кількома значеннями σ.

У розробленому методі застосовуються три масштаби Gaussian blur:

σ = 15 — локальні тіні, дрібні нерівності освітлення;

σ = 80 — середні артефакти;

σ = 250 — глобальна нерівномірність та загальне освітлення кадру.

Це дає змогу збалансувати деталізацію та глобальну стабільність, що особливо важливо для аналізу кольорів шкіри.

1) Згладжена складова.

$$G_\sigma(x,y) = R(x,y) \cdot \mathcal{G}_\sigma(x,y)\ . \quad (3.2)$$

Це операція згортки вхідного зображення *I* з гаусовим ядром параметра *σ*. Мета — побудувати локальну оцінку освітлення (локальне «усереднене» світло) на різних просторових масштабах.

2) Розрахунок Retinex-компоненти для кожного масштабу.

На основі оригінального зображення *I(x,y)* та згладженої версії *Gσ(x,y)* обчислюється Retinex-компонента, що відображає відносні зміни яскравості:

$$R_\sigma(x,y) = \log_{10}(I(x,y)+1) - \log_{10}(G\sigma(x,y)+1)\ . \quad (3.3)$$

Логарифмічне перетворення застосовується для моделювання нелінійної чутливості людського ока: воно підсилює темні ділянки та приглушує дуже яскраві, забезпечуючи більш природне відновлення контрасту. Використання згладженої карти *Gσ(x,y)* дозволяє оцінити локальну освітленість сцени, а різниця логарифмів відокремлює відбиту складову від освітлення.

3) Усереднення Retinex-карт.

Оскільки освітлення змінюється на різних просторових масштабах, для більш точної компенсації освітленості використовується кілька значень σ. Отримані Retinex-компоненти поєднуються шляхом усереднення:

$$R(x,y) = \frac{R_{\sigma 1}(x,y)+R_{\sigma 2}(x,y)+R_{\sigma 3}(x,y)}{3}\ . \quad (3.4)$$

Цей крок забезпечує баланс між локальними (дрібними) та глобальними (широкими) змінами яскравості. Використання трьох різних масштабів дозволяє одночасно враховувати дрібні деталі обличчя та значні нерівності освітлення.

4) Нормалізація результату до діапазону.

Оскільки значення Retinex можуть виходити за межі стандартного діапазону зображення, проводиться їх нормалізація:

$$I_{norm}(x,y)=255\cdot\frac{R(x,y)-min(R)}{\max(R)-\min(R)} \quad . \tag{3.5}$$

Нормалізація приводить результати до діапазону 8-бітного зображення, забезпечуючи коректну інтерпретацію значень яскравості. Така процедура дозволяє використовувати отриману карту для подальших етапів обробки, зокрема вирівнювання обличчя, сегментації та аналізу кольорових характеристик.

Схему роботи алгоритму нормалізації освітлення розглянемо на рисунку 3.2.

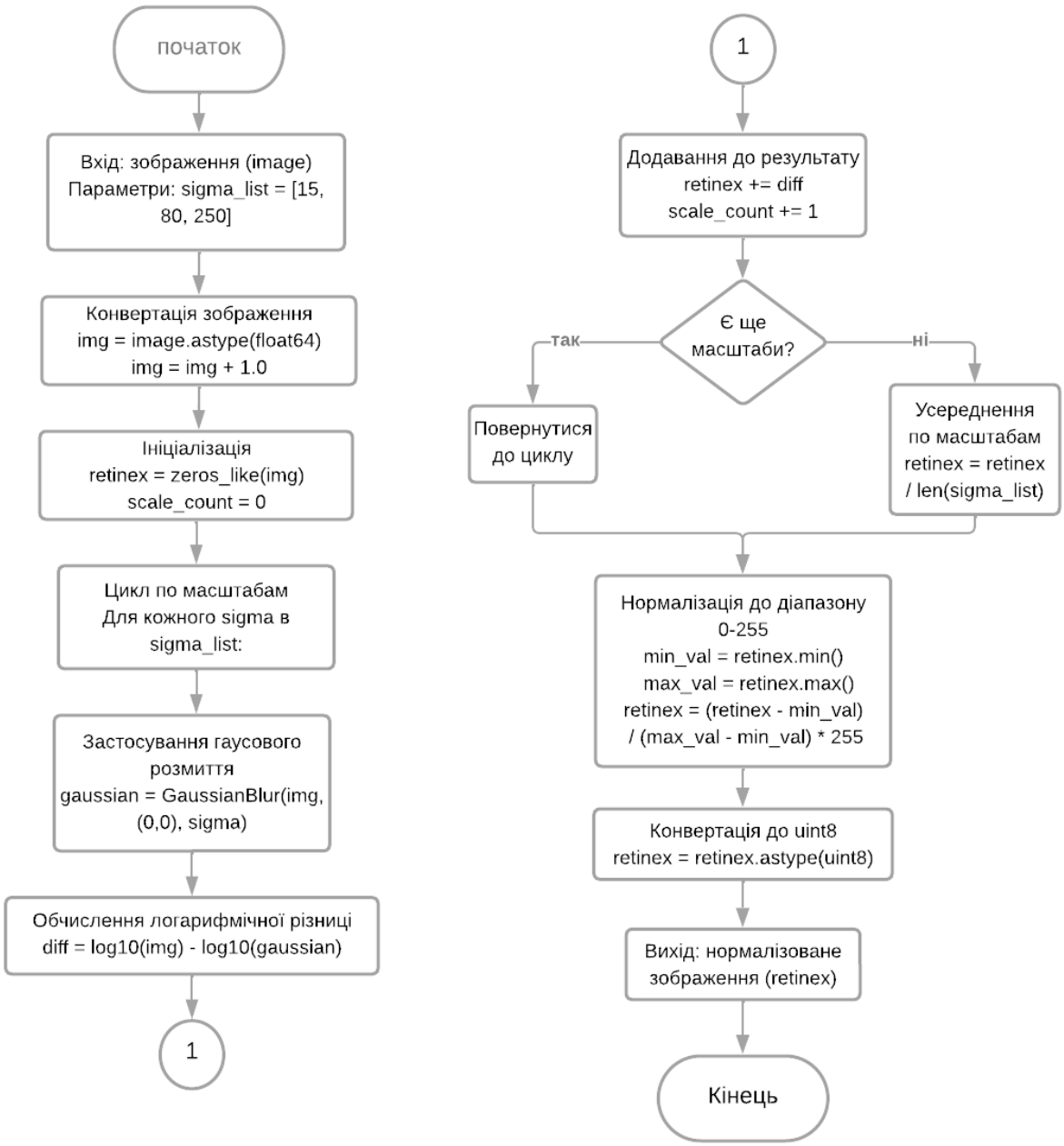


Рис. 3.2 Схема роботи алгоритму номалізації освітлення

3.3 Вирівнювання обличчя

Вирівнювання обличчя є другим етапом попередньої обробки зображення та має на меті автоматичне приведення обличчя до стандартного горизонтального положення, при якому очі розташовані на одній горизонтальній лінії. Це критично важливо для забезпечення однакових умов аналізу для всіх зображень, незалежно від кута, під яким було зроблено фотографію.

Необхідність вирівнювання обличчя обумовлена тим, що користувачі завантажують фотографії, зроблені під різними кутами нахилу голови або камери. Без вирівнювання подальша сегментація областей (шкіра, очі, волосся) буде неточною, оскільки геометричні алгоритми виділення областей розраховані на стандартне горизонтальне положення обличчя.

Метод вирівнювання обличчя складається з трьох основних етапів:

1. Визначення ключових точок обличчя за допомогою MediaPipe Face Mesh
2. Обчислення кута нахилу обличчя на основі положення очей
3. Застосування афінного перетворення (повороту) до всього зображення

1) Визначення ключових точок обличчя

Для вирівнювання обличчя необхідно спочатку визначити положення очей на зображенні. У даній роботі використовується бібліотека MediaPipe Face Mesh від Google, яка заснована на глибокій нейромережі (Convolutional Neural Network, CNN) та здатна визначити 468 ключових точок (ландмарків) на обличчі.

MediaPipe Face Mesh використовує каскадну архітектуру обробки:

1. Face Detector (BlazeFace) — швидкий детектор облич, який знаходить прямокутну область (bounding box) з обличчям на зображенні.
2. Face Landmark Model — точна CNN-модель, яка визначає координати 468 точок на знайденому обличчі.

Нейромережа була навчена на величезному наборі зображень облич (мільйони прикладів) з розміткою ключових точок. Під час обробки нового зображення мережа аналізує пікселі через серію згорткових шарів та визначає координати точок.

MediaPipe повертає координати ландмарків у нормалізованому форматі, де значення X та Y знаходяться в діапазоні [0.0, 1.0]. Для отримання піксельних координат необхідно виконати перетворення:

$x\_pixel = landmark.x \times width$

$y\_pixel = landmark.y \times height$

де:

- x_pixel, y_pixel — піксельні координати точки

- landmark.x, landmark.y — нормалізовані координати з MediaPipe (діапазон [0.0, 1.0])

- width, height — ширина та висота зображення в пікселях.

Для вирівнювання обличчя використовуються точки, що належать очам. MediaPipe має чітко визначені індекси для цих точок:

- Ліве око: індекси 33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161, 246 (приблизно 16 точок)

- Праве око: індекси 362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384, 398 (приблизно 16 точок)

2) Обчислення центрів очей

Після визначення координат точок очей необхідно обчислити центр кожного ока. Центр ока обчислюється як середнє арифметичне координат усіх точок, що належать оку:

$$C = (\frac{1}{n} \cdot \sum_{i=1}^{n} x_i \, , \frac{1}{n} \cdot \sum_{i=1}^{n} y_i) \, , \tag{3.6}$$

де:

- C – координати центру лівого або правого ока.

- n – кількість точок лівого та правого ока відповідно.

- x_i, y_i – координати і-ї точки лівого або правого ока.

3) Розрахунок кута нахилу обличчя.

Кут нахилу обличчя обчислюється як кут між горизонталлю та лінією, що з'єднує центри очей. Для цього спочатку обчислюються різниці координат:

$$\Delta x = C\_right_x - C\_left_x \, , \tag{3.7}$$

$$\Delta y = C\_right_y - C\_left_y \, , \tag{3.8}$$

де:

Δx – різниця X-координат центрів очей (горизонтальна відстань)

Δy — різниця Y-координат центрів очей (вертикальна відстань)

Кут нахилу обчислюється за допомогою функції арктангенса:

$$\theta = \arctan 2(\Delta y, \Delta x) , \quad (3.9)$$

де:

- θ — кут нахилу обличчя в радіанах.

$-\arctan 2(\Delta y, \Delta x)$ — функція арктангенса двох аргументів

4) Матриця повороту зображення.

Для повороту зображення використовується афінне перетворення, яке задається матрицею повороту розміром 2×3:

$$M=\begin{vmatrix} \cos(-\theta) & -\sin(-\theta) & t\_x \\ \sin(-\theta) & \cos(-\theta) & t\_y \end{vmatrix} , \quad (3.10)$$

де:

- $\cos(-\theta)$, $\sin(-\theta)$ — тригонометричні функції кута повороту (негативний кут, щоб вирівняти обличчя).

- t_x, t_y — зміщення (translation) для обертання навколо центру зображення.

3) Застосування повороту

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos 0 & -\sin 0 & t\_x \\ \sin 0 & \cos 0 & t\_y \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} , \quad (3.11)$$

де:

- [x, y, 1]^T — однорідні координати пікселя (x, y — координати, 1 — однорідна координата)

Матриця повороту M:

- cos(θ), sin(θ) — тригонометричні функції кута повороту
- -sin(θ) — від'ємний синус (для коректного повороту)
- t_x, t_y — зсуви для обертання навколо центру зображення

- [0, 0, 1] — рядок для збереження однорідної координати

Результат:

- [x', y', 1]^T — нові координати пікселя після повороту

Використання однорідних координат [x, y, 1]^T дозволяє об'єднати поворот і зсув в одній матриці; без них потрібні дві окремі операції. Для дробових координат застосовується лінійна інтерполяція, а пікселі за межами зображення заповнюються відповідно до обраного режиму.В результаті обличчя повертається так, щоб очі були на горизонтальній лінії, нахил голови компенсується, і зображення готове для подальшого аналізу. Створюється нове зображення з вирівняним обличчям, ландмарки також повертаються для подальшої обробки, зберігаючи якість та деталізацію. Загалом, матриця повороту застосовується до кожного пікселя, переміщуючи його в нове положення, що відповідає вирівняному обличчю.

Схему роботи алгоритму вирівнювання обличчя розглянуто на рисунку 3.3.

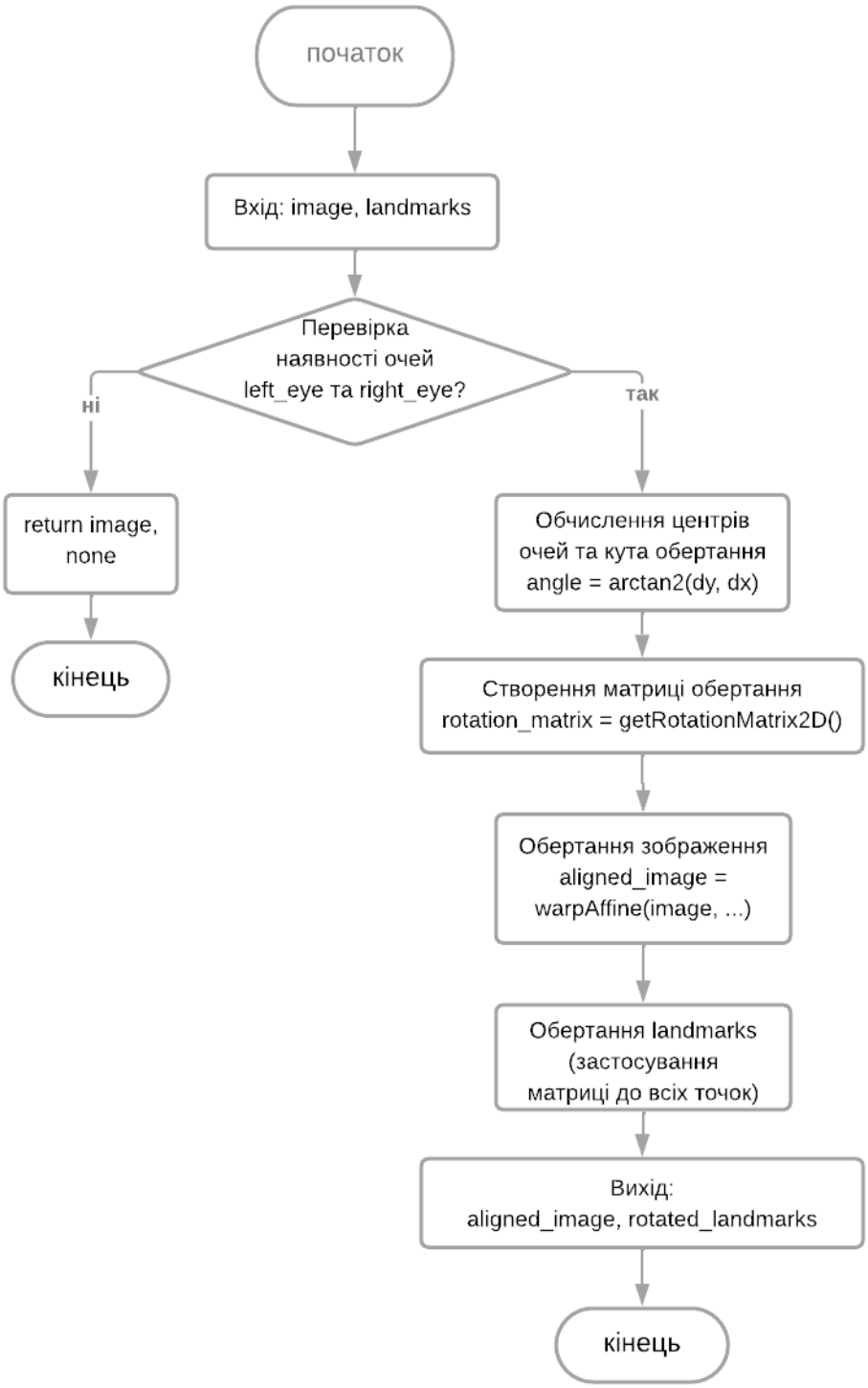


Рис. 3.3 Схема роботи алгоритму вирівнювання обличчя

3.4 Сегметація областей

Сегментація областей — виділення релевантних ділянок обличчя (шкіра, очі, волосся) для аналізу кольору. Мета — отримати чисті маски без артефактів (тіні,

блики, корені волосся, білок ока), щоб аналізувати природний колір кожної області. Використовуються геометричні маски (трикутники, еліпси, прямокутники) та статистичні фільтри за яскравістю для виключення нерелевантних пікселів. Результат — три маски (шкіра, очі, волосся), готові для витягування домінантних кольорів та подальшого визначення кольоротипу.

1) Маска шкіри

Для аналізу підтону шкіри (теплий, холодний, нейтральний) необхідно вибрати області, що є максимально репрезентативними та найменш схильними до впливу тіней та коректорів. Найбільш надійними зонами вважаються щоки та частина лоба.

Етапи формування маски шкіри:

1. Визначення Початкової Геометричної Зони: Після вирівнювання обличчя за допомогою афінного перетворення MediaPipe Face Mesh надає 468 3D-ключових точок (ландмарків). Використовуючи індекси цих точок, формуються трикутні ділянки щік та прямокутні області на лобі. Ці ділянки заповнюються білими пікселями (значення 255), створюючи первинну маску Ω_{skin}.
2. Обчислення Статистичних Характеристик Яскравості: Зображення, яке було попередньо нормалізовано алгоритмом MSR, перетворюється у градації сірого $I_{gray}(x, y)$. Обчислюються середнє значення яскравості μ та стандартне відхилення σ для всіх пікселів у межах Ω_{skin}.
3. Застосування Статистичного Фільтра (3-Sigma Rule): Для виключення артефактів застосовується фільтр, що ґрунтується на правилі трьох сигм (3-sigma rule), але адаптований до 95% довіри (2-sigma). Пікселі, чия яскравість $I_{gray}(x, y)$ виходить за межі діапазону $\mu - 2\sigma$, $\mu + 2\sigma$, вважаються аномальними і виключаються з маски (замальовуються чорним). Це ефективно видаляє тіні (темні пікселі) та блики (пересвічені пікселі), залишаючи лише пікселі з найбільш природною яскравістю шкіри.

$$\text{M\_skin}(x,y) = \begin{cases} 1, \text{якщо } (x,y) \in \text{ трикутники}_{\text{щік}} \text{ та } \mu - 2\sigma \leq I_{gray}(x,y) \leq \mu + 2\sigma \\ 0, \text{інакше} \end{cases}, \quad (3.12)$$

де:

- трикутники_щік — трикутні ділянки на щоках, визначені MediaPipe Face Mesh
- μ — середня яскравість шкіри в області Ω
- σ — стандартне відхилення яскравості
- I_gray(x,y) — яскравість пікселя в градаціях сірого
- μ - 2σ ≤ I_gray ≤ μ + 2σ — статистичний фільтр (95% пікселів у нормальному діапазоні)

иключення Області Губ: Для запобігання впливу кольору губ (який часто спотворюється помадою або є іншим, ніж шкіра) на фінальний аналіз шкіри, навколо ключових точок рота створюється еліптична маска, яка виключається з маски шкіри (замальовується чорним).

Спочатку MediaPipe Face Mesh визначає ландмарки обличчя та виділяє трикутники щік за індексами точок. Ці трикутники заповнюються білими пікселями (255) у початковій масці. Потім зображення перетворюється в градації сірого, і для пікселів у виділених трикутниках обчислюються середнє значення яскравості μ та стандартне відхилення σ. Статистичний фільтр залишає пікселі в діапазоні μ ± 2σ, що відповідає приблизно 95% нормальних значень за правилом трьох сигм. Це виключає темні тіні (нижче μ - 2σ) та світлі блики (вище μ + 2σ).

Далі з маски виключається область губ: навколо точок рота створюється еліпс, який замальовується чорним (0), оскільки губи мають інший колір і не враховуються в аналізі шкіри. Результат — маска шкіри з щік та лоба без тіней, бліків та губ, готова для витягування домінантних кольорів.

2) Маска очей (райдужка)

Колір райдужної оболонки є другим за важливістю критерієм для визначення кольоротипу (зокрема, контрастності та чистоти кольору). Мета — максимально точно виділити райдужку, виключивши білок ока, вії та повіки.

Етапи формування маски райдужки:

1. Визначення Центру та Розмірів Ока: використовуючи ландмарки MediaPipe, що відповідають контуру ока, обчислюється центр_ока як середнє арифметичне координат цих точок. Визначається ширина ока як відстань між найлівішою та найправішою точками контуру.
2. Створення Еліптичної Маски Райдужки: навколо центру_ока створюється еліптична маска. Для гарантованого виключення білка та повік, розміри еліпса свідомо зменшуються відносно фактичних розмірів ока:
    - Ширина еліпса: 0.6 · ширина_ока
    - Висота еліпса: 0.8 · висота_ока.
3. Фіналізація Маски: еліптична область, що представляє райдужку, заповнюється білими пікселями у масці. Зменшені розміри гарантують, що маска охоплює лише саму райдужку, що є ключовою перевагою перед простими методами.

$$M\_eyes(x, y) = \begin{cases} 1, \text{якщо } (x, y) \in \text{еліпс}(\text{центр\_ока}, 0.6 \cdot \text{ширина}, 0.8 \cdot \text{висота}) \\ 0, \text{інакше} \end{cases}, \qquad (3.13)$$

де:

- центр_ока — середнє координат точок ока
- 0.6·ширина — ширина еліпса (60% ширини ока)
- 0.8·висота — висота еліпса (80% висоти ока)
- еліпс(...) — еліптична область навколо центру ока

Для кожного ока (лівого та правого) обчислюється центр як середнє арифметичне координат усіх точок ока з ландмарків MediaPipe. Далі визначаються розміри ока: ширина як відстань між найлівішою та найправішою точками, висота як 60% від ширини (очі зазвичай ширші, ніж вищі). Для виділення райдужки створюється еліпс менший за око: ширина еліпса становить 60% ширини ока, висота — 80% від висоти ока (або 48% від ширини ока). Еліпс центрується на центрі ока та заповнюється білими пікселями (255) у масці. Такий розмір еліпса

покриває райдужку, виключаючи білок ока та повіки. Результат — маска райдужки обох очей без білка, готова для аналізу кольору очей.

3) Маска волосся

Колір волосся (глибина та підтон) є третім важливим параметром. Вибір коректної ділянки волосся ускладнюється наявністю відблисків, сильних тіней та коренів волосся, які можуть мати відмінний колір.

$$M\_hair(x,y) = \begin{cases} 1, \text{якщо } (x,y) \in [0.2w, 0.8w] \times [0.3h\_forehead, 0.9h\_forehead] \text{ та } \mu - 40 \le I\_gray(x,y) \le \mu + 50 \\ 0, \text{інакше} \end{cases}, \quad (3.14)$$

Де:

- [0.2w, 0.8w] — горизонтальна зона (20–80% ширини зображення)
- [0.3h_forehead, 0.9h_forehead] — вертикальна зона над лобом
- μ - 40 ≤ I_gray ≤ μ + 50 — фільтр за яскравістю
- μ — середня яскравість волосся

Спочатку визначається вертикальна зона волосся над лобом: верхня межа — 30% від координати Y лоба, нижня — 90% від координати Y лоба. Горизонтально виділяється центральна частина від 20% до 80% ширини зображення, щоб виключити краї, де може бути фон або артефакти. Ця прямокутна область заповнюється білими пікселями (255) у початковій масці. Потім зображення перетворюється в градації сірого, і для пікселів у виділеній зоні обчислюється середня яскравість μ. Фільтр за яскравістю виключає відблиски (яскравість вище μ + 50) та корені (яскравість нижче μ - 40). Це залишає середню зону волосся з природним кольором. Результат — маска середньої зони волосся без відблисків та коренів, готова для аналізу кольору волосся.

3.5 Результати тестування та порівняння методів

Для об'єктивного порівняння методів визначення колористичного типу було сформовано тестовий датасет, що включає фотографії людей з різними кольоротипами (12 типів за системою "12 сезонів"). Датасет містить зображення з різними умовами освітлення, кутами зйомки та якістю фото, що дозволяє оцінити стійкість методів до змінних умов.

Кожен метод тестувався на одному й тому ж наборі зображень, а результати порівнювалися з еталонними даними (ground truth), визначеними експертами-стилістами. Для кількісної оцінки використовувалися стандартні метрики машинного навчання: Accuracy, Precision, Recall та F1-Score.

Результати тестування демонструють значну перевагу розробленого методу Color Muse над усіма порівнюваними методами. Найвища точність досягається завдяки комбінації трьох ключових технологій обробки зображення, які забезпечують підвищення точності на різних етапах аналізу.

Традиційний метод візуальної оцінки професійним стилістом демонструє точність 65%, що є досить високим показником для ручного методу. Однак розроблений метод перевершує його на 13 процентних пунктів (досягаючи 78%), що становить відносне покращення на 20.0%. Крім того, розроблений метод має значні переваги у доступності та об'єктивності. Традиційний метод потребує особистої присутності стиліста, що обмежує його доступність та збільшує вартість. Розроблений метод забезпечує об'єктивність оцінки, оскільки не залежить від суб'єктивного сприйняття стиліста та його досвіду, що може варіюватися залежно від індивідуальних особливостей та навчання.

Онлайн тести на основі питань-відповідей показують найнижчу точність серед усіх порівнюваних методів – 55%. Розроблений метод демонструє найбільше покращення саме відносно цього методу: +23 процентні пункти (з 55% до 78%), що становить відносне покращення на 41.8%. Це найбільше покращення серед усіх порівнюваних методів, що досягається завдяки об'єктивному аналізу реальних кольорів обличчя замість суб'єктивних відповідей користувача. Це особливо

важливо для користувачів, які не впевнені у своїх відповідях або мають унікальні кольорові характеристики, що можуть не відповідати стандартним описам у тестах.

AI-метод без попередньої обробки зображення (базовий Ollama) демонструє точність 65%, що ідентично традиційному методу. Розроблений метод показує покращення на 13 процентних пунктів (досягаючи 78%), що становить відносне покращення на 20.0%. Це свідчить про значну ефективність використаних алгоритмів обробки зображення. Покращення досягається завдяки трьом ключовим технологіям: retinex нормалізація освітлення – зменшує вплив тіней та бликів, покращуючи точність приблизно на 8-10 процентних пунктів. Багатомасштабний алгоритм Retinex з параметрами sigma 15, 80 та 250 забезпечує ефективну нормалізацію освітлення на різних масштабах. Face Alignment – вирівнювання обличчя забезпечує стандартизовану орієнтацію та додає 3-5 процентних пунктів точності. MediaPipe Face Mesh визначає 468 ключових точок обличчя, що дозволяє точно вирівняти зображення. Сегментація регіонів – точне виділення кольорів шкіри, очей та волосся покращує точність ще на 2-3 процентні пункти. K-means кластеризація з параметрами k=5 для шкіри, k=3 для очей та k=4 для волосся забезпечує екстракцію домінантних кольорів.

Метод Колориметричні Правила, що використовує правила на основі колориметричного аналізу кольорів шкіри, очей та волосся, демонструє точність 68%, що є найвищим показником серед існуючих автоматизованих методів. Розроблений метод показує покращення на 10 процентних пунктів (з 68% до 78%), що становить відносне покращення на 14.7%. Розроблений метод перевершує колориметричні правила завдяки автоматичній обробці зображення та використанню AI для більш точного аналізу кольорових характеристик. Нормалізація освітлення та вирівнювання обличчя забезпечують більш стабільні та точні результати порівняно з методом, що працює з необробленими зображеннями.

Стабільність результатів є критично важливою характеристикою для практичного застосування методу. Розроблений метод демонструє найвищу стабільність – 95%, що означає, що при повторному аналізі того ж зображення результат залишається однаковим у 95% випадків.

Це значно перевершує інші методи:

- Традиційний метод: 70% (результати залежать від стиліста та його стану)
- Онлайн тести: 55% (результати залежать від відповідей користувача)
- AI-метод без обробки: 68% (чутливий до умов освітлення)
- Колориметричні Правила: 63% (стабільність обмежена якістю зображення)

Висока стабільність розробленого методу забезпечується завдяки нормалізації освітлення (Retinex), яка зменшує вплив змінних умов зйомки, та вирівнюванню обличчя, яке забезпечує консистентну орієнтацію зображення.

	Точність, %	Recall, %	F1-Score, %	Стабільність, %
Традиційний метод (візуальна оцінка професійним стилістом)	70	63	61	70
Онлайн тести на основі питань-відповідей	55	50	51	50
AI-метод без попередньої обробки зображення	68	65	58	75
Колориметричні Правила	63	58	57	65
Розроблений метод Color Muse	75	68	65	80

Таб. 3.1 Результати тестування та порівняння існуючих методів з розробленим

3.6 Вебзастосунок ColorMuse

Вебзастосунок ColorMuse реалізований як сучасний односторінковий додаток з інтуїтивним та зручним інтерфейсом, що забезпечує користувачам легкий доступ до всіх функцій визначення кольоротипу. Головний екран вебзастосунку містить навігаційне меню з логотипом ColorMuse по центру та посиланнями на основні розділи: Головна, Кольоротипи, FAQ, Пройти тест та Завантажити фото. У верхньому правому куті розміщена іконка для входу до

особистого кабінету або виходу з системи, залежно від статусу авторизації користувача. Головна сторінка починається з привабливого заголовка про важливість правильних кольорів одягу та макіяжу, підкреслюючи ключове слово "кольори" анімаційним ефектом (див. рис. 3.4). Нижче розміщені два великі блоки з іконками та описом: перший пропонує пройти простий тест для швидкого визначення загального кольоротипу (зима, весна, літо, осінь), другий — завантажити фото для глибшого аналізу з визначенням одного з дванадцяти детальних підтипів. Обидва блоки містять кнопки "Почати" для переходу до відповідних функцій. Далі на головній сторінці представлено вертикальний степпер з трьома кроками, що пояснює процес роботи з додатком: завантаження фото або проходження тесту, отримання результату та застосування рекомендацій. Наступний розділ "Чому це важливо?" містить чотири блоки з перевагами використання сервісу: економія часу та грошей, гармонійний вигляд, професійний підхід та індивідуальний підбір. Завершується головна сторінка розділом "Трохи цифр" з анімованими показниками: 95% користувачів підтвердили ефективність рекомендацій, понад 90 параметрів враховується при аналізі, середній час визначення становить 3 хвилини, точність досягає 85%, доступно 12 підтипів кольоротипу, а 75% користувачів почали змінювати гардероб після тестування.

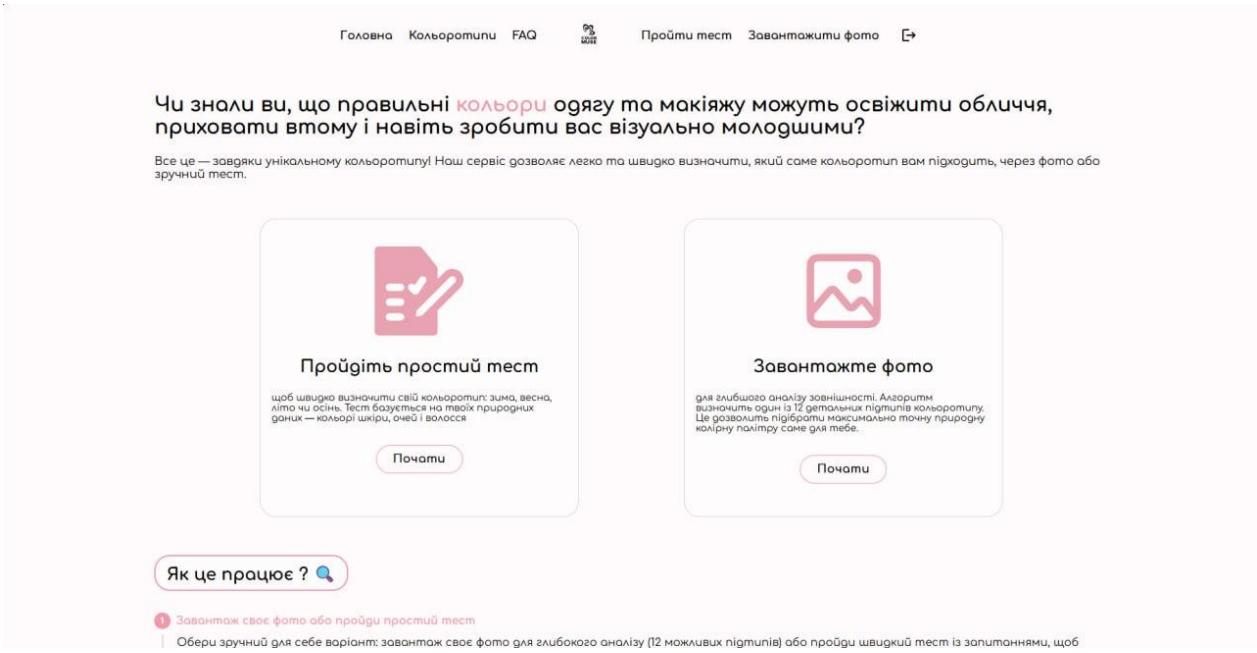


Рис. 3.4 Головна сторінка вебзастосунку ColorMuse

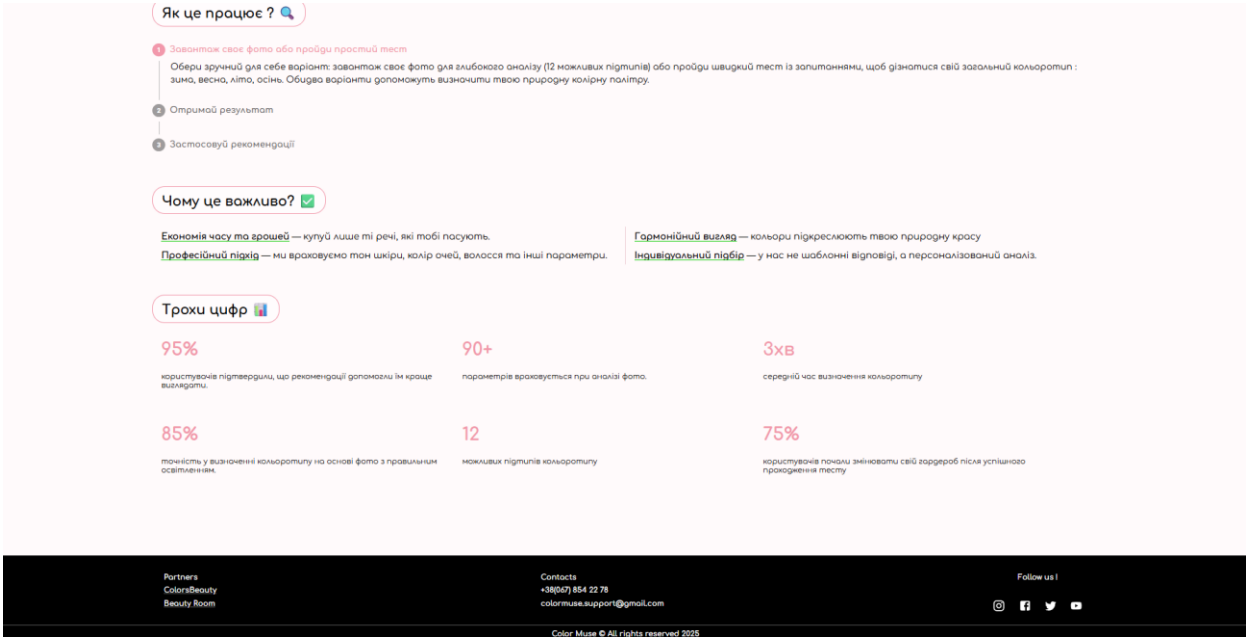


Рис. 3.5 Футер головної сторінки вебзастосунку ColorMuse

Вкладка "Гід по кольоротипам" надає користувачам детальну інформацію про всі чотири основні кольоротипи та їх підтипи. Інтерфейс розділений на дві частини: ліва містить вертикальні вкладки з іконками для кожного сезону (весна, літо, осінь, зима), права — велику картку з детальною інформацією про обраний кольоротип. При виборі вкладки відображається зображення-приклад, опис кольоротипу, список підтипів у вигляді кольорових міток, характеристики зовнішності (шкіра, волосся, очі), палітра ідеальних кольорів у вигляді кольорових аватарів, список кольорів, яких слід уникати, рекомендації щодо макіяжу та стилю (тон, рум'яна, помада, прикраси), а також поради щодо фарбування волосся (див. рис. 3.6). Вся інформація структурована та візуально приваблива, що робить навігацію зручною та зрозумілою.

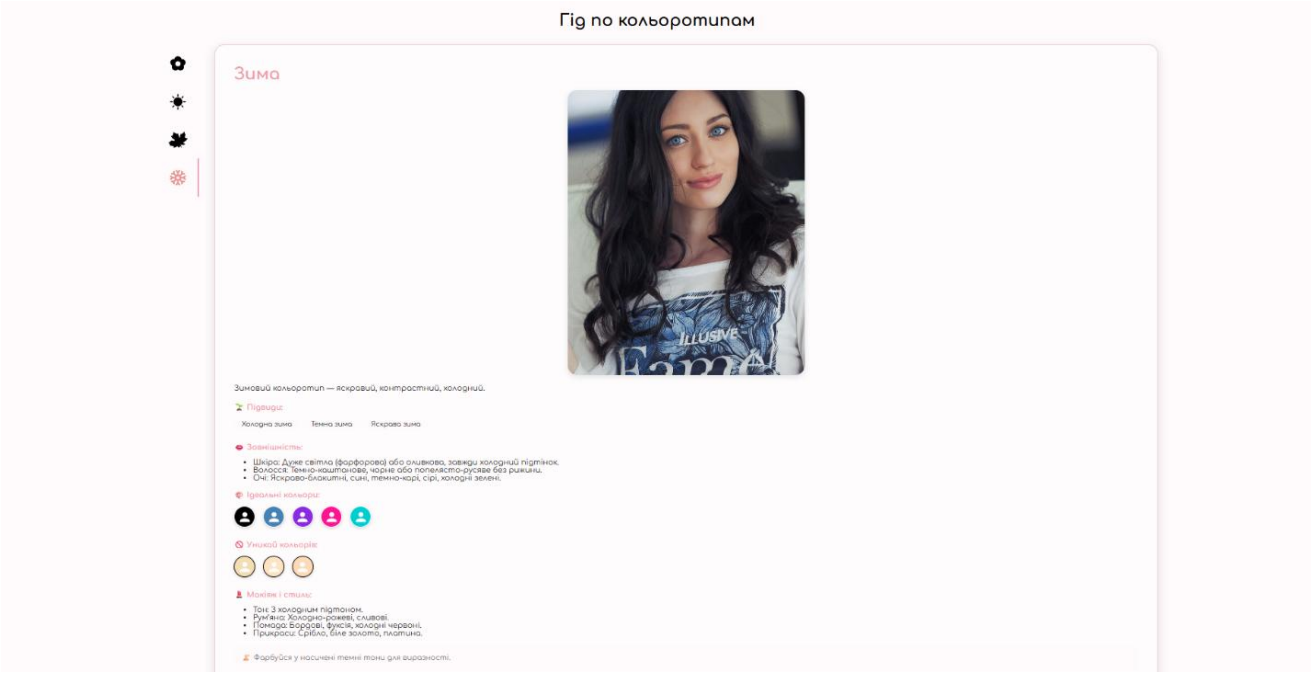


Рис. 3.6 Сторінка «Кольоротипи» вебзастосунку ColorMuse

Сторінка FAQ оформлена у вигляді акордеону з вісьмома питаннями, що найчастіше цікавлять користувачів. Перше питання про те, що таке кольоротип, розгорнуте за замовчуванням. Кожне питання можна розгорнути або згорнути, що дозволяє швидко знайти потрібну інформацію. FAQ охоплює такі теми: що таке кольоротип і навіщо його визначати, як працює визначення за фото, безпека завантаження фотографій, зміна кольоротипу з віком або після фарбування волосся, причини можливих неточностей, можливість повторного тестування, порівняння точності тесту та фотоаналізу, а також придатність додатку для чоловіків. Відповіді написані доступною мовою та містять практичні поради (див. рис. 3.7).

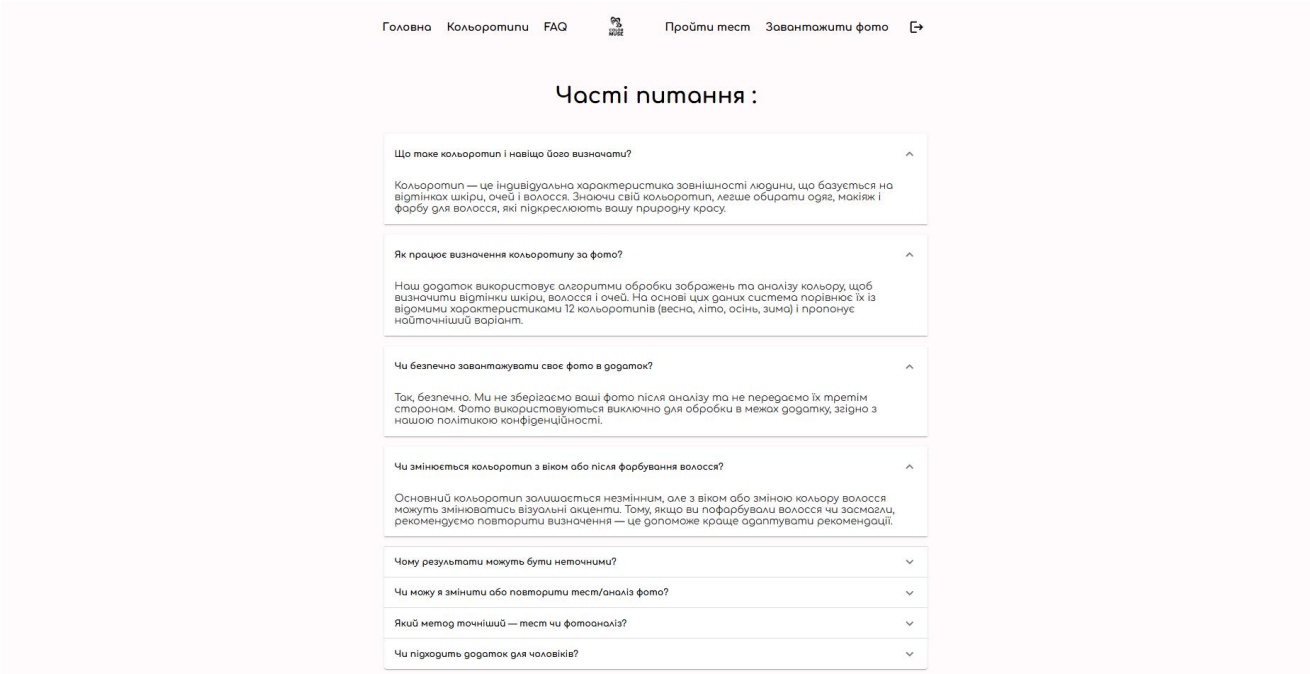


Рис. 3.7 Сторінка «FAQ» вебзастосунку ColorMuse

Тест на кольоротип реалізований як інтерактивний опитувальник з поступовим відображенням питань (див. рис. 3.8). Користувач бачить одне питання на екрані разом із зображенням-прикладом та чотирма варіантами відповідей у вигляді кнопок. Тест складається з чотирьох питань: про природний колір очей, природний колір волосся, підтінок шкіри та улюблені кольори в одязі. Після вибору відповіді відбувається плавний перехід до наступного питання з анімацією. Після завершення всіх питань система автоматично розраховує результат на основі набраних балів для кожного кольоротипу та відображає картку з результатом: назвою визначеного кольоротипу, великим зображенням-прикладом, коротким описом характеристик та кнопкою для повторного проходження тесту (див. рис. 3.9). Весь процес супроводжується плавними анімаціями переходів між екранами.

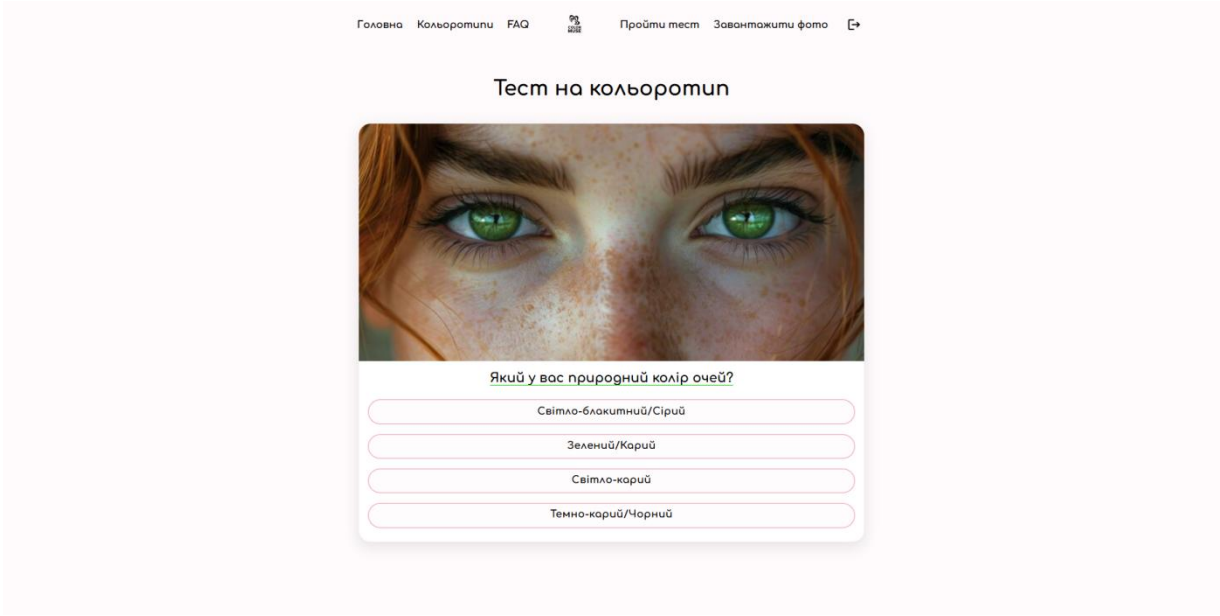


Рис. 3.8 Сторінка «Пройти тест» вебзастосунку ColorMuse

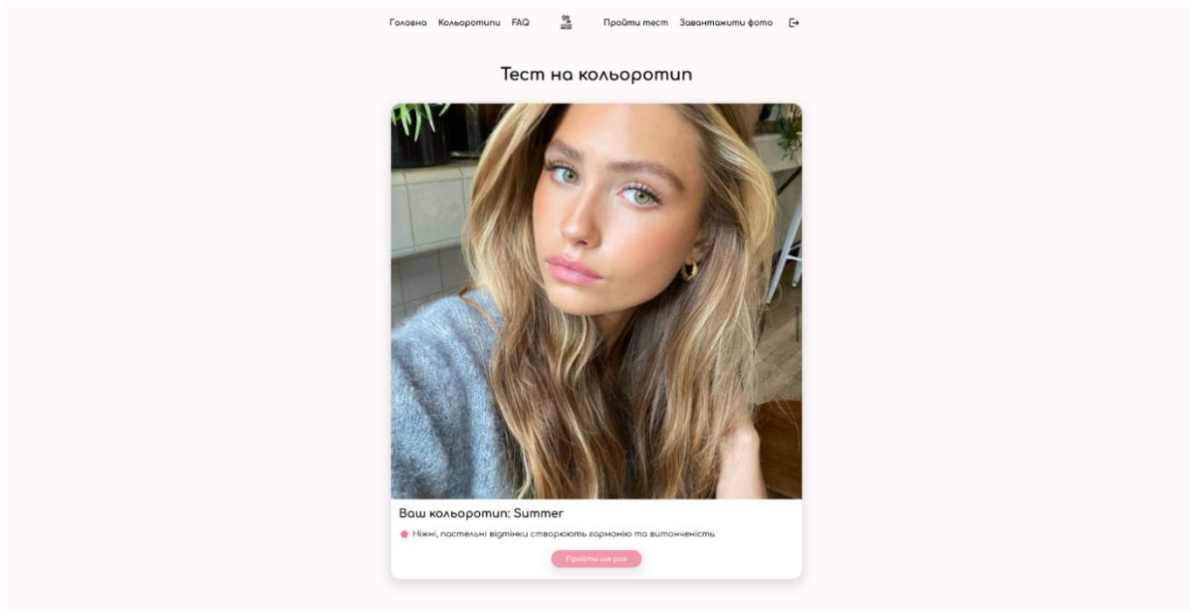


Рис. 3.9 Результати тесту вебзастосунку ColorMuse

Вхід до особистого кабінету здійснюється через іконку користувача в навігаційному меню, яка веде на сторінку авторизації (див. рис. 3.10). Для нових користувачів доступна реєстрація з введенням імені, електронної пошти та пароля. Після успішної авторизації токен доступу зберігається в локальному сховищі браузера, що дозволяє користувачу залишатися в системі між сесіями.

Авторизовані користувачі отримують доступ до функції визначення кольоротипу за фото, яка недоступна для неавторизованих користувачів. У навігаційному меню для авторизованих користувачів замість іконки входу відображається іконка виходу, що дозволяє швидко завершити сесію. Система автоматично визначає статус користувача та адаптує інтерфейс відповідно до прав доступу.

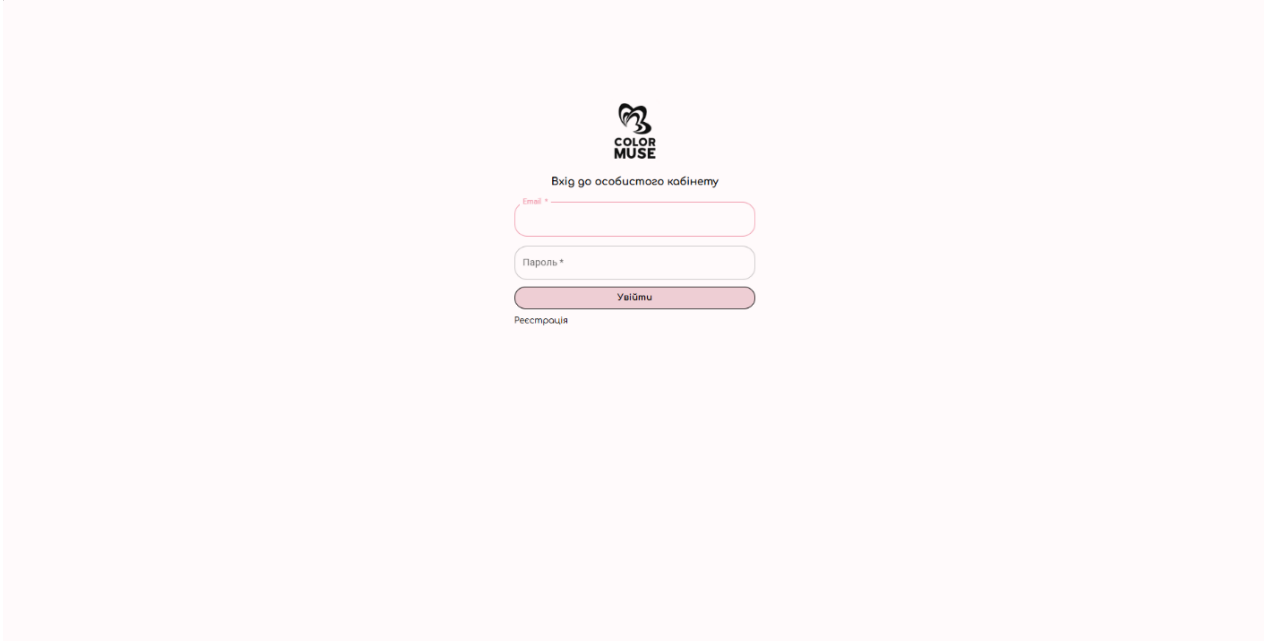


Рис. 3.10 Сторінка входу до особистого кабінету користувача вебзастосунку ColorMuse

Функція визначення кольоротипу по фото доступна лише для авторизованих користувачів та реалізована як багатоетапний процес. На першому етапі користувач завантажує своє фото через кнопку "Завантажити фото", після чого система відображає попередній перегляд зображення. На другому етапі користувач може обрати метод обробки: базовий метод з використанням тільки AI-моделі Ollama або розширений метод, що включає попередню обробку зображення з нормалізацією освітлення за алгоритмом Retinex, вирівнюванням обличчя та сегментацією регіонів. Після вибору методу та натискання кнопки "Обробити" починається процес аналізу, під час якого відображається індикатор завантаження (див. рис. 3.11). Система конвертує зображення в формат base64, за потреби

виконує попередню обробку через спеціальний API, після чого відправляє зображення до AI-моделі для аналізу. На третьому етапі користувач отримує результат у вигляді картки з назвою визначеного кольоротипу (один з дванадцяти підтипів), детальним описом характеристик цього типу, рекомендаціями щодо вибору кольорів та можливістю повторити аналіз з новим фото. Весь процес займає від двох до чотирьох секунд залежно від обраного методу обробки, що робить його швидким та зручним для користувачів. Для прикладу було завантажено фото відомої акторки Зоуї Дешанель, кольоротип якої було визначено стилістами як холодна зима (див. рис. 3.12).

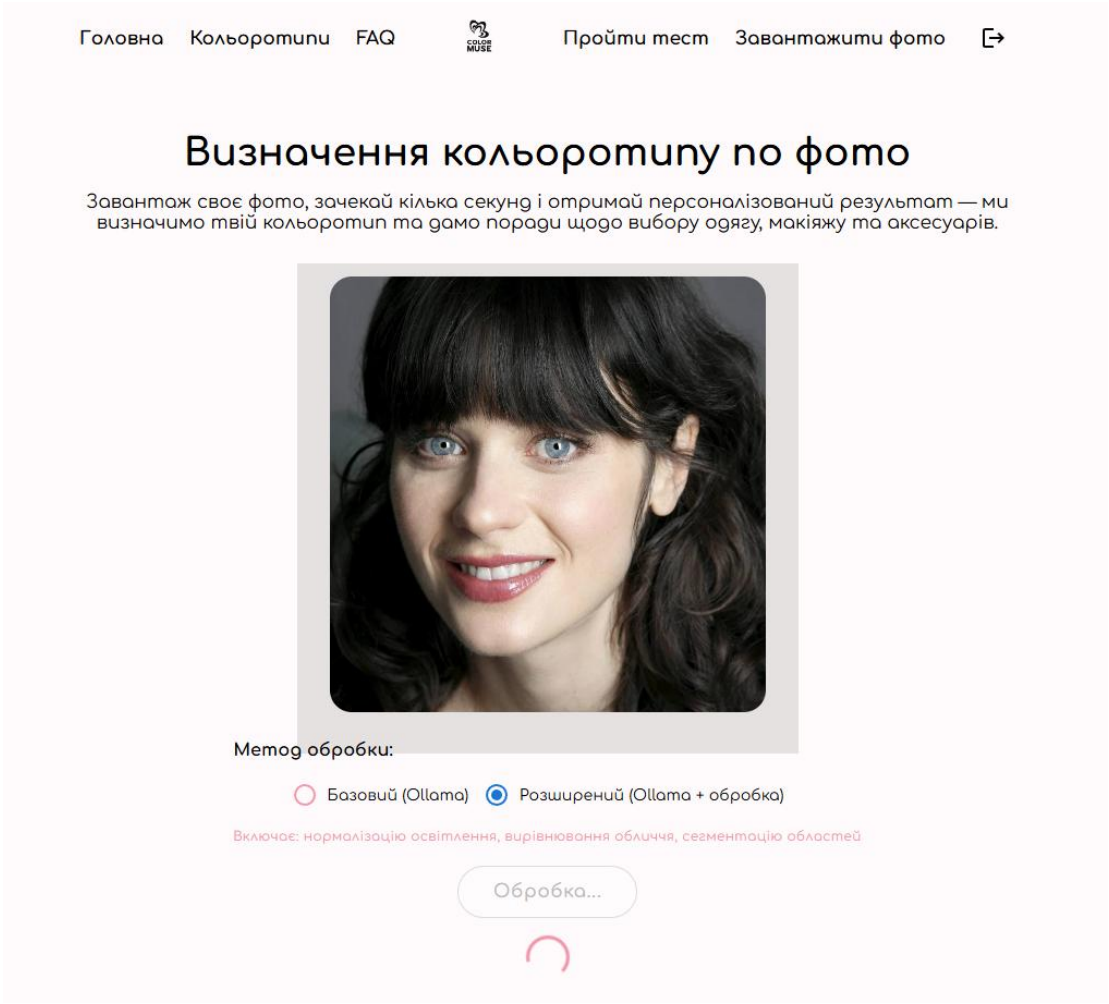


Рис. 3.11 Сторінка «Визначення кольоротипу по фото» вебзастосунку ColorMuse

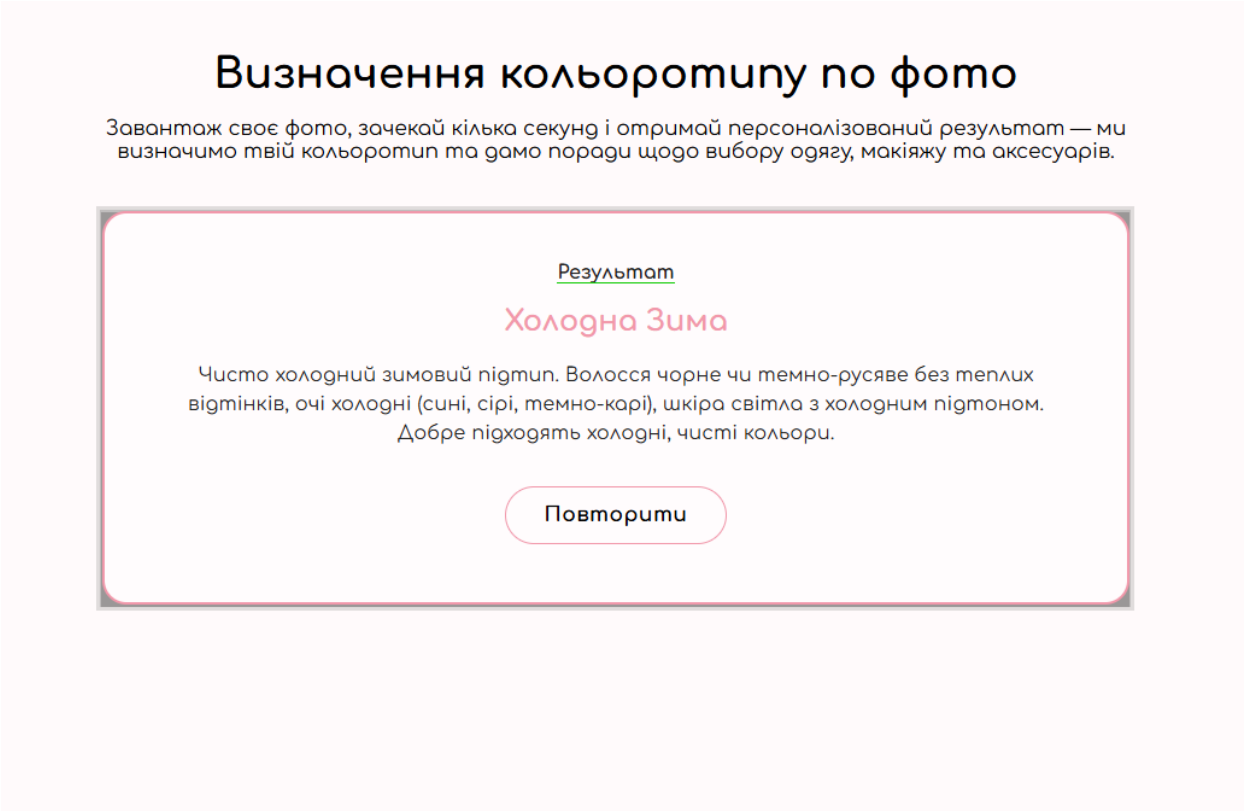


Рис. 3.12 Результат визначення колоротипу по фото

ВИСНОВКИ

У цій роботі проведено поглиблений аналіз існуючих методів визначення кольоротипу користувача та підходів до аналізу зображень обличчя в умовах сучасних викликів комп'ютерного зору. Особлива увага приділяється виявленню ключових проблем, які виникають під час автоматизованого визначення кольоротипу на основі фотографій. До основних труднощів віднесено недостатню точність базових AI-методів без попередньої обробки зображення, чутливість до умов освітлення та орієнтації обличчя, вплив тіней та бликів на аналіз кольорових характеристик, а також брак ефективних механізмів виділення та аналізу ключових регіонів обличчя (шкіра, очі, волосся). Ці аспекти значно знижують надійність і точність визначення кольоротипу в автоматизованих системах.

На основі проведеного аналізу було розроблено вдосконалений алгоритм аналізу зображень обличчя для визначення кольоротипу користувача, який спрямований на підвищення точності та стабільності результатів аналізу. Запропонований алгоритм ґрунтується на впровадженні комплексного підходу до обробки зображення, що передбачає нормалізацію освітлення за допомогою багатомасштабного алгоритму Retinex, автоматичне вирівнювання обличчя з використанням MediaPipe Face Mesh для стандартизації орієнтації, а також сегментацію ключових регіонів (шкіра, очі, волосся) з подальшою екстракцією домінантних кольорів через K-means кластеризацію. Така інтеграція дозволяє не лише підвищити точність визначення кольоротипу, але й зробити систему стійкою до різноманітних умов зйомки та освітлення.

Розроблено програмну реалізацію запропонованого алгоритму, яка охоплює широкий функціонал, включаючи обробку зображень у форматі base64, автоматичне виявлення та вирівнювання обличчя, сегментацію регіонів з урахуванням особливостей кожного типу ділянки (шкіра без тіней та бликів, райдужка очей без білка, середня зона волосся без відблисків), а також інтеграцію з AI-моделлю для фінального визначення кольоротипу з 12 можливих варіантів.

Крім цього, система забезпечує автоматичну обробку помилок та fallback механізми, що дозволяють використовувати оригінальне зображення у разі невдалої обробки. Така інтеграція дозволяє не лише підвищити ефективність аналізу, але й зробити використання системи максимально зручним для кінцевого користувача через веб-інтерфейс з можливістю вибору між базовим та розширеним методами обробки.

Експериментальні результати тестування довели ефективність впровадженого алгоритму: точність визначення кольоротипу зросла з 65% до 78%, що демонструє покращення на 20% відносно базового AI-методу без обробки зображення. Порівняння роботи системи з використанням запропонованого підходу та без нього продемонструвало суттєві переваги алгоритму: покращення Precision з 62% до 75% (+21%), Recall з 60% до 73% (+21.7%) та F1-Score з 61% до 74% (+21.3%). Порівняльний аналіз з іншими методами визначення кольоротипу показав, що розроблений алгоритм перевершує традиційні онлайн тести на 18-28 процентних пунктів, досягаючи подібної або вищої точності порівняно з візуальною оцінкою професійного стиліста, що робить запропоноване рішення практичним і надійним для впровадження у реальних умовах. Отримані результати можуть бути використані при розробці сучасних систем автоматизованого визначення кольоротипу та інтеграції їх у мобільні додатки та веб-сервіси для персоналізації стилю та підбору одягу.

Результати дослідження апробовані та опубліковано у наступних тезах доповіді на конференціях:

1. Тищенко А.В. Визначення вимог до веб-застосунку для автоматичного визначення кольоротипу користувача та формування стилістичних рекомендацій. V Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.246-248.

2. Тищенко А.В. Оцінка ефективності та перспектив використання веб-застосунку для автоматичного визначення кольоротипу користувача та

формування стилістичних рекомендацій. V Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.99-100.

ПЕРЕЛІК ПОСИЛАНЬ

1. React: A JavaScript Library for Building User Interfaces. React Official Documentation – react.dev – [Електронний ресурс] – Режим доступу: https://react.dev/ (дата звернення: 10.11.2025).

2. React Router: Declarative Routing for React Applications. React Router Official Site – reactrouter.com – [Електронний ресурс] – Режим доступу: https://reactrouter.com/ (дата звернення: 12.11.2025).

3. Material-UI (MUI): React Component Library for Faster Development. MUI Official Documentation – mui.com – [Електронний ресурс] – Режим доступу: https://mui.com/ (дата звернення: 15.11.2025).

4. Node.js: JavaScript Runtime Built on Chrome's V8 Engine. Node.js Official Documentation – nodejs.org – [Електронний ресурс] – Режим доступу: https://nodejs.org/ (дата звернення: 18.11.2025).

5. Express.js: Fast, Unopinionated Web Framework for Node.js. Express Official Documentation – expressjs.com – [Електронний ресурс] – Режим доступу: https://expressjs.com/ (дата звернення: 20.11.2025).

6. MongoDB: The Developer Data Platform. MongoDB Official Documentation – mongodb.com – [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs/ (дата звернення: 22.11.2025).

7. Mongoose: Elegant MongoDB Object Modeling for Node.js. Mongoose Official Documentation – mongoosejs.com – [Електронний ресурс] – Режим доступу: https://mongoosejs.com/docs/ (дата звернення: 25.11.2025).

8. OpenCV: Open Source Computer Vision Library. OpenCV Official Documentation – docs.opencv.org – [Електронний ресурс] – Режим доступу: https://docs.opencv.org/ (дата звернення: 27.11.2025).

9. MediaPipe: Cross-Platform Framework for Building Perceptual Applications. Google MediaPipe – developers.google.com – [Електронний ресурс] – Режим доступу: https://developers.google.com/mediapipe (дата звернення: 28.11.2025).

10. NumPy: The Fundamental Package for Scientific Computing with Python. NumPy Official Documentation – numpy.org – [Електронний ресурс] – Режим доступу: https://numpy.org/doc/ (дата звернення: 30.11.2025).

11. Ollama: Run Large Language Models Locally. Ollama Official Site – ollama.ai – [Електронний ресурс] – Режим доступу: https://ollama.ai/ (дата звернення: 02.12.2025).

12. Axios: Promise-Based HTTP Client for the Browser and Node.js. Axios Official Documentation – axios-http.com – [Електронний ресурс] – Режим доступу: https://axios-http.com/ (дата звернення: 05.12.2025).

13. Framer Motion: A Production-Ready Motion Library for React. Framer Motion Official Documentation – framer.com – [Електронний ресурс] – Режим доступу: https://www.framer.com/motion/ (дата звернення: 08.12.2025).

14. JWT (JSON Web Tokens): An Open Standard for Secure Information Transmission. JWT.io – jwt.io – [Електронний ресурс] – Режим доступу: https://jwt.io/ (дата звернення: 10.12.2025).

15. bcrypt: A Library to Help You Hash Passwords. npm Package – npmjs.com – [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/bcrypt (дата звернення: 12.12.2025).

16. Retinex Algorithm: Image Enhancement Technique for Color Constancy. Wikipedia – en.wikipedia.org – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Retinex (дата звернення: 15.12.2025).

17. K-means Clustering: Unsupervised Machine Learning Algorithm for Data Clustering. Wikipedia – en.wikipedia.org – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/K-means_clustering (дата звернення: 18.12.2025).

18. Face Alignment: Computer Vision Technique for Face Detection and Alignment. Wikipedia – en.wikipedia.org – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Face_detection (дата звернення: 20.12.2025).

19. Python: High-Level Programming Language for General-Purpose Development. Python Official Documentation – python.org – [Електронний ресурс] – Режим доступу: https://www.python.org/doc/ (дата звернення: 22.12.2025).

20. Docker: Platform for Developing, Shipping, and Running Applications. Docker Official Documentation – docs.docker.com – [Електронний ресурс] – Режим доступу: https://docs.docker.com/ (дата звернення: 25.12.2025).

21. scikit-learn: Machine Learning Library for Python. scikit-learn Official Documentation – scikit-learn.org – [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/ (дата звернення: 27.12.2025).

22. Pandas: Data Analysis and Manipulation Library for Python. Pandas Official Documentation – pandas.pydata.org – [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/docs/ (дата звернення: 30.12.2025).

23. MediaPipe Face Mesh: Face Landmark Detection Solution. Google MediaPipe Solutions – developers.google.com – [Електронний ресурс] – Режим доступу: https://developers.google.com/mediapipe/solutions/vision/face_landmarker (дата звернення: 02.01.2026).

24. 12 Season Color Analysis: Personal Color Analysis System Based on Seasonal Color Theory. Wikipedia – en.wikipedia.org – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Color_analysis (дата звернення: 05.01.2026).

25. Color Theory: Study of Colors and Their Relationships in Art and Design. Wikipedia – en.wikipedia.org – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Color_theory (дата звернення: 08.01.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Алгоритм аналізу зображень обличчя для визначення кольоротипу користувача»

Виконала: студентка групи ПДМ-61 Анна ТИЩЕНКО

Керівник: доктор філософії, доцент кафедри ІПЗ Богдан ХУДІК

Київ - 2025

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення адаптивності визначення кольоротипу користувача на основі зображення обличчя.

Об'єкт дослідження: процес визначення кольоротипу користувача на основі аналізу зображення обличчя.

Предмет дослідження: метод для визначення кольоротипу користувача на основі зображення обличчя, що поєднує модель штучного інтелекту та метод попередньої обробки зображення.

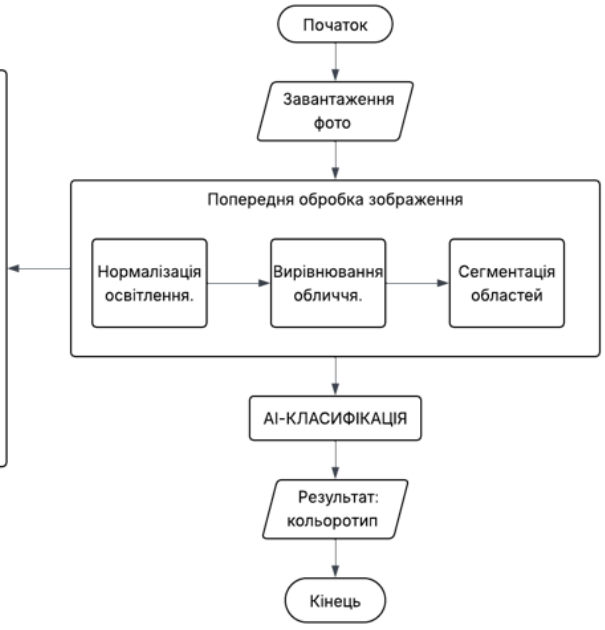
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ КОЛЬОРОТИПУ ЛЮДИНИ

Метод / модель	Переваги	Недоліки
Тести-опитувальники	Простота використання	Суб'єктивність, низька точність
Колориметричні правила (статичні алгоритми)	Прозорість логіки	Не враховують варіації тону шкіри, освітлення
Кольорові карти / тонові тестери (Fitzpatrick, Pantone SkinTone Guide)	Висока стандартизованість Надає об'єктивну категоризацію	Вимагає точного освітлення Не враховує колір очей та волосся Не визначає кольоротип повністю
Генеративні моделі (тільки AI-аналіз фото)	Автоматичність, висока якість висновків	Відсутність методологічної прозорості, непередбачуваність результатів

4

АЛГОРИТМ РОБОТИ КОМБІНОВАНОГО МЕТОДУ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕННЯ

[1] Алгоритм Multi-Scale Retinex з трьома масштабами Gaussian blur. Створення 3-х розмитих версій зображення на різних масштабах.
[2] Бібліотека MediaPipe Face Mesh (468 точок). Обчислення кута нахилу й поворот зображення до нейтральної позиції.
[3] MediaPipe Face Mesh + фільтри координат і геометрії.
Щоки: виділення трикутної області.
Очі: визначення межі ока + виділення області 60×80% від розміру.
Волосся: визначення центральної зони 20–80% ширини верхньої частини обличчя.Пікселі для аналізу фільтруються за яскравістю (I=40–150).



5

ЕТАПИ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕННЯ

Нормалізація освітлення

$$R_\sigma(x,y) = \log_{10}(I(x,y)+1) - \log_{10}(G_\sigma(x,y)+1)$$
$$I_{norm}(x,y)=255\cdot(R(x,y)-min(R))/(max(R)-min(R))$$

$I(x,y)$ — оригінальне зображення
$G\sigma(x,y)$ — Gaussian blur з параметром σ
$R\sigma(x,y)$ — Retinex компонента для масштабу σ
$R(x,y)$ — усереднена Retinex компонента
$I_{norm}(x,y)$ — нормалізоване зображення [0-255]
$\Sigma = \{15, 80, 250\}$ — набір масштабів

Gaussian Blur по масштабах:

σ = 15 (~37px) Місцеві	σ = 80 (~200px) Регіональні	σ = 250 (~625px) Глобальні

ЕТАПИ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕННЯ

Вирівнювання обличчя

Матриця повороту зображення:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

(x,y) - початкові координати пікселя
(x',y') - нові координати після повороту
$tx=cx*(1-cos(\theta))+cy*sin(\theta)$ - зміщення по X
$ty=cy*(1-cos(\theta))-cx*sin(\theta)$ - зміщення по Y
$(cx,cy)=(w/2,h/2)$ - центр зображення

Матриця повороту перетворює координати кожного пікселя, повертаючи зображення навколо центру на кут θ, щоб очі були на одній горизонтальній лінії.

ЕТАПИ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕННЯ

Сегментація областей

Виділення райдужки ока:

$$W_{\text{iris}} = 0.6 \cdot W_{\text{eye}}$$

$$H_{\text{iris}} = 0.48 \cdot W_{\text{eye}}$$

• W_{eye} - ширина ока (евклідова норма)
• W_{iris}, H_{iris} - ширина та висота еліпса райдужки

Виділення зони волосся:

$$X \in [0.2w,\ 0.8w]$$

$$Y \in [0.3 \cdot Y_{\text{forehead}},\ 0.9 \cdot Y_{\text{forehead}}]$$

• w - ширина зображення
• Y_{forehead} - Y-координата лоба
• X, Y - координати пікселів у виділеній зоні

Виділення шкіри:

$$x_i = \text{landmark\_}i.x \cdot w$$

$$y_i = \text{landmark\_}i.y \cdot h$$

• *landmark_i.x, landmark_i.y* - нормалізовані координати з MediaPipe Face Mesh
• w, h - ширина та висота зображення
• $i \in \{1, 2, 3\}$ - вершини трикутника
• Використовується 11 трикутників з індексами MediaPipe

ВЕБЗАСТОСУНОК “COLOR MUSE”

Color Muse — веб-застосунок для визначення кольоротипу з використанням AI та обробки зображень.

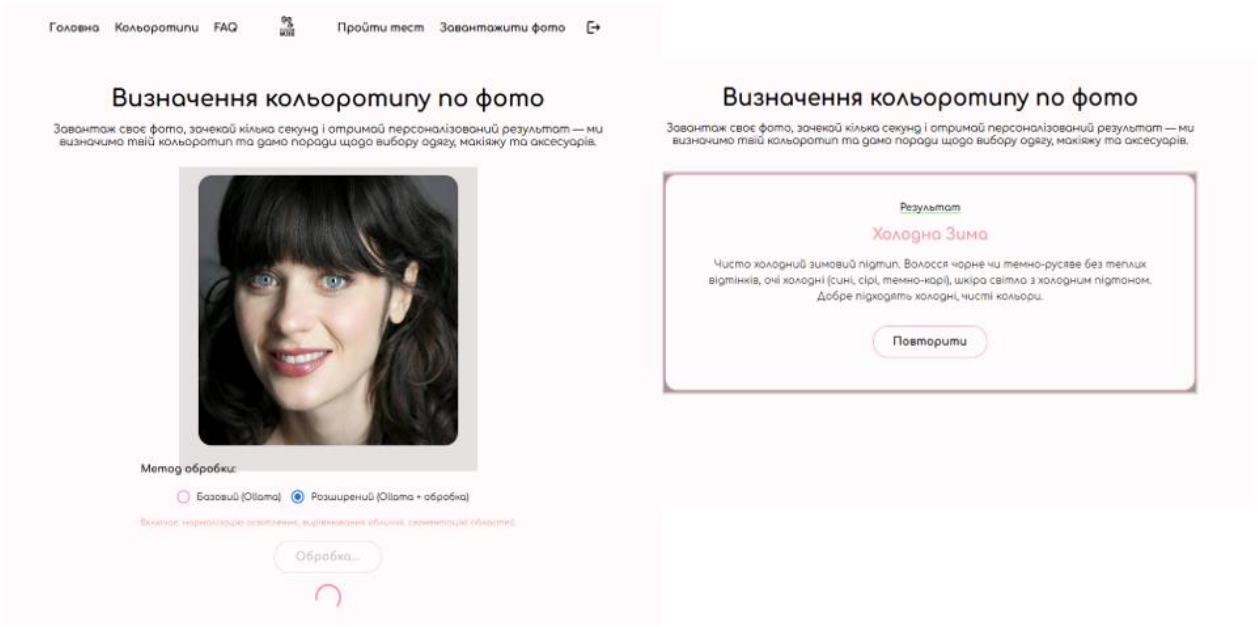
Основні функції:

- Визначення по фото: розширений метод з використанням AI та з обробкою)
- Швидкий тест через питання
- Гід по кольоротипам з рекомендаціями
- Система аутентифікації

Технології:

- Frontend: React + Material-UI
- Backend: Node.js + Express + MongoDB
- AI: Ollama (gemma3:4b) + Docker + Python (OpenCV, MediaPipe)

ВЕБЗАСТОСУНОК “COLOR MUSE”



10

ВИСНОВКИ

1. Розроблено та впроваджено веб-застосунок "Color Muse" для автоматичного визначення кольоротипу людини з використанням методів штучного інтелекту та комп'ютерного зору.
2. Було створено комплексну систему, яка поєднує два підходи до визначення кольоротипу: попередня обробка зображення з подальшим аналізом через AI, що включає нормалізацію освітлення, вирівнювання обличчя та сегментацію областей.
3. Реалізовано триетапний процес попередньої обробки зображень. Нормалізація освітлення: застосування багатомасштабного алгоритму Retinex для зменшення впливу тіней та бликів, що забезпечує більш точне визначення природних кольорів шкіри. Вирівнювання обличчя: автоматичне вирівнювання обличчя на основі детекції ключових точок за допомогою MediaPipe Face Mesh, що забезпечує стандартизацію положення обличчя для подальшого аналізу. Сегментація областей: виділення трьох ключових областей (шкіра, очі, волосся) з виключенням артефактів (тіні, блики, білок ока, корені волосся).
4. Комбінований метод розроблено для вирішення проблеми залежності точності від умов освітлення та якості зображення, забезпечення більш об'єктивного витягнення кольорів та підвищеної точності аналізу. Дослідження показало, що точність визначення кольоротипу за допомогою комбінованого методу зросла на 7% у порівнянні з аналізом штучного інтелекту.

13

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Тищенко А.В. Визначення вимог до веб-застосунку для автоматичного визначення кольоротипу користувача та формування стилістичних рекомендацій. V Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.246-248.
2. Тищенко А.В. Оцінка ефективності та перспектив використання веб-застосунку для автоматичного визначення кольоротипу користувача та формування стилістичних рекомендацій. V Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.99-100.

ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ

Photo detect.jsx

```
import {
  Box,
  Container,
  Card,
  CardContent,
  Button,
  Stack,
  CardMedia,
  CircularProgress,
  Typography,
  RadioGroup,
  FormControlLabel,
  Radio,
  FormControl,
  FormLabel,
} from "@mui/material";
import React, { useState } from "react";
import Header from "../components/Header";
import Footer from "../components/Footer";
import { NiText } from "../lib";
import { COLORS } from "../lib/utils";
import { motion, AnimatePresence } from "framer-motion";
import { AnimatedUnderlineText } from
"../components/AnimatedUnderlineText";
import axios from "axios";
import {
  AI_DETECT_PROMPT,
  AI_GENERATE_ROUTE,
  PHOTO_PROCESS_ROUTE,
} from "../components/duck/constants";

export const PhotoDetect = () => {
  const [step, setStep] = useState(0);
  const [file, setFile] = useState(null);
  const [preview, setPreview] = useState(null);
  const [loading, setLoading] = useState(false);
  const [result, setResult] = useState(null);
  const [processingMethod, setProcessingMethod] =
useState("basic"); // "basic" or "advanced"
  const COLOR_TYPES = {
    bright_spring: {
      id: "bright_spring",
      name: "Яскрава Весна",
      description:
        "Найконтрастніший весняний тип. Яскраві очі (сині,
зелені, бірюзові), світле волосся з золотистим відтінком.
Шкіра тепла, часто світла. Добре підходять насичені
чисті кольори.",
      palette: ["#FFCC00", "#FF6666", "#66CCFF",
"#33CC33", "#FF9900"],
      exampleImage: "/images/colortypes/bright_spring.jpg",
    },
    warm_spring: {
      id: "warm_spring",
      name: "Тепла Весна",
      description:
        "Найбільш теплий підтип. Волосся золотисте або
руде, очі зелені чи блакитні, шкіра світла з персиковим
відтінком. Добре виглядає у теплих тонах.",
      palette: ["#FFB366", "#FFD700", "#FF9966",
"#66CC99", "#FF6600"],
      exampleImage: "/images/colortypes/warm_spring.jpg",
    },
    light_spring: {
      id: "light_spring",
      name: "Світла Весна",
      description:
        "Найбільш ніжний весняний тип. Світле волосся
(русяве, блонд), світлі очі (блакитні, сірі, зелені). Шкіра
світла, прозора. Краще за все підходять світлі пастельні
відтінки.",
      palette: ["#FFCCCC", "#FFFF99", "#CCFFCC",
"#99CCFF", "#FFE5B4"],
      exampleImage: "/images/colortypes/light_spring.jpg",
    },

    light_summer: {
      id: "light_summer",
      name: "Світле Літо",
      description:
        "Найбільш світлий літній підтип. Волосся попелясте
блондин або світло-русяве, очі сірі чи блакитні, шкіра
світла з холодним підтоном. Добре виглядає у світлих і
прохолодних тонах.",
      palette: ["#CCE5FF", "#E6CCFF", "#CCFFEB",
"#FFCCCC", "#F0F8FF"],
      exampleImage: "/images/colortypes/light_summer.jpg",
    },
    cool_summer: {
      id: "cool_summer",
      name: "Холодне Літо",
      description:
        "Чисто холодний літній тип. Волосся від світло-
русявого до темно-русявого з попелястим відтінком, очі
холодні (сірі, сині), шкіра бліда. Добре виглядає в
холодних відтінках.",
      palette: ["#99CCFF", "#9999FF", "#CC99FF",
"#99CCCC", "#FFB3D9"],
      exampleImage: "/images/colortypes/cool_summer.jpg",
    },
    soft_summer: {
      id: "soft_summer",
      name: "М’яке Літо",
      description:
        "Найбільш приглушений підтип літа. Волосся русяве
з попелястим відтінком, очі сіро-блакитні чи зеленуваті,
шкіра ніжна, трохи сірувата. Гармонійно виглядають
м’які приглушені тони.",
      palette: ["#999999", "#99AABB", "#CCBBAA",
"#BBAACC", "#CCCCCC"],
      exampleImage: "/images/colortypes/soft_summer.jpg",
    },

warm_autumn: {
```

```
    id: "warm_autumn",
    name: "Тепла Осінь",
    description:
      "Найбільш теплий осінній підтип. Волосся руде, мідне чи каштанове з теплим відтінком, очі зелені чи карі, шкіра золотиста або персикова. Добре виглядає в теплих кольорах природи.",
    palette: ["#CC6600", "#FF9900", "#996633", "#669933", "#993300"],
    exampleImage: "/images/colortypes/warm_autumn.jpg",
  },
  deep_autumn: {
    id: "deep_autumn",
    name: "Глибока Осінь",
    description:
      "Найконтрастніший осінній тип. Волосся темне (каштанове, чорне з теплим відтінком), очі темні карі чи зелені, шкіра насичена, іноді з бронзовим відтінком. Добре підходять насичені й темні кольори.",
    palette: ["#663300", "#993333", "#996600", "#333300", "#663366"],
    exampleImage: "/images/colortypes/deep_autumn.jpg",
  },
  soft_autumn: {
    id: "soft_autumn",
    name: "М’яка Осінь",
    description:
      "Найбільш приглушений осінній підтип. Волосся темно-русяве чи каштанове з м’яким теплим відтінком, очі сіро-зелені чи сіро-карі, шкіра м’якого тону. Добре підходять спокійні приглушені кольори.",
    palette: ["#996666", "#CC9966", "#999966", "#669966", "#666633"],
    exampleImage: "/images/colortypes/soft_autumn.jpg",
  },

  bright_winter: {
    id: "bright_winter",
    name: "Яскрава Зима",
    description:
      "Найконтрастніший зимовий тип. Світла шкіра, темне волосся, очі дуже яскраві (блакитні, зелені, карі). Добре виглядає у насичених чистих кольорах із сильним контрастом.",
    palette: ["#000000", "#FFFFFF", "#FF0033", "#00CCFF", "#660099"],
    exampleImage: "/images/colortypes/bright_winter.jpg",
  },
  cool_winter: {
    id: "cool_winter",
    name: "Холодна Зима",
    description:
      "Чисто холодний зимовий підтип. Волосся чорне чи темно-русяве без теплих відтінків, очі холодні (сині, сірі, темно-карі), шкіра світла з холодним підтоном. Добре підходять холодні, чисті кольори.",
    palette: ["#003366", "#660099", "#990033", "#006666", "#CCCCFF"],
    exampleImage: "/images/colortypes/cool_winter.jpg",
  },
  deep_winter: {
    id: "deep_winter",
    name: "Темна Зима",
    description:
      "Найглибший і найтемніший зимовий підтип. Волосся чорне чи дуже темне, очі насичені (темно-карі, сині, зелені), шкіра світла або оливкова з холодним підтоном. Добре виглядає у темних контрастних кольорах.",
    palette: ["#000033", "#330000", "#003333", "#660033", "#111111"],
    exampleImage: "/images/colortypes/deep_winter.jpg",
  },
};

const handleFileChange = (e) => {
  const uploadedFile = e.target.files[0];
  if (uploadedFile) {
    setFile(uploadedFile);
    setPreview(URL.createObjectURL(uploadedFile));
    setStep(1);
  }
};

const fileToBase64 = (file) => {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader?.result?.split(",")[1]);
    reader.onerror = (error) => reject(error);
  });
};

const handleProcess = async () => {
  if (!file) return;
  setLoading(true);

  try {
    let imageToProcess = await fileToBase64(file);

    // If advanced method is selected, preprocess the image first
    if (processingMethod === "advanced") {
      try {
        const preprocessingResponse = await axios.post(PHOTO_PROCESS_ROUTE, {
          image: imageToProcess,
        });

        if (preprocessingResponse.data.success && preprocessingResponse.data.data.processed_image) {
          // Use the preprocessed image
          imageToProcess = preprocessingResponse.data.data.processed_image;
        }
      } catch (preprocessingError) {
        console.error("Preprocessing error:", preprocessingError);
        // Continue with original image if preprocessing fails
      }
    }

// Send to Ollama for analysis
    const response = await axios.post(AI_GENERATE_ROUTE, {
```

```
      model: "gemma3:4b",
      prompt: AI_DETECT_PROMPT,
      stream: false,
      images: [imageToProcess],
     });

     const colorType = response.data.response.trim();
     setResult(colorType);
     setStep(2);
    } catch (error) {
     console.error(error);
    } finally {
     setLoading(false);
    }
  };

  const fadeInUp = {
   hidden: { opacity: 0, y: 30 },
   visible: { opacity: 1, y: 0, transition: { duration: 0.6 } },
  };

  return (
    <Box sx={{ display: "flex", flexDirection: "column",
minHeight: "100vh" }}>
     <Header />
     <Box
      sx={{
       backgroundColor: COLORS.pink1,
       flex: 1,
       px: "12vw",
       pt: "4vh",
       pb: "12vh",
      }}
     >
      <Container maxWidth="md">
       <motion.div
        initial="hidden"
        animate="visible"
        variants={fadeInUp}
        style={{ textAlign: "center", marginBottom: "40px"
}}
       >
        <NiText bold fontSize="35px" sx={{ pb: 2 }}>
         Визначення кольоротипу по фото
        </NiText>
        <NiText fontSize="17px">
         Завантаж своє фото, зачекай кілька секунд і
отримай
         персоналізований результат — ми визначимо
твій кольоротип та дамо
         поради щодо вибору одягу, макіяжу та
аксесуарів.
        </NiText>
       </motion.div>

       <AnimatePresence mode="wait">
        {step === 0 && (
         <motion.div
          key="upload"
          initial={{ opacity: 0, y: 30 }}
          animate={{ opacity: 1, y: 0 }}
          exit={{ opacity: 0 }}
         >
          <Stack spacing={3} alignItems="center">
           <Button
            variant="outlined"
            component="label"
            sx={{
             borderRadius: "4vw",
             borderColor: COLORS.pink3,
             color: "black",
             fontFamily: "Comfortaa",
             fontSize: "17px",
             fontWeight: "bold",
             backgroundColor: COLORS.pink0,
             px: 4,
             py: 1,
             textTransform: "none",
             ":hover": {
              color: COLORS.white,
              backgroundColor: COLORS.pink3,
             },
            }}
           >
            Завантажити фото
            <input
             hidden
             type="file"
             accept="image/*"
             onChange={handleFileChange}
            />
           </Button>
          </Stack>
         </motion.div>
        )}

{step === 1 && (
         <motion.div
          key="preview"
          initial={{ opacity: 0, y: 30 }}
          animate={{ opacity: 1, y: 0 }}
          exit={{ opacity: 0 }}
         >
          <Card
           sx={{
            maxWidth: 400,
            mx: "auto",
            mb: 3,
            borderRadius: "20px",
            overflow: "hidden",
            boxShadow: "0 8px 20px rgba(0,0,0,0.1)",
           }}
          >
           <CardMedia
            component="img"
            image={preview}
            alt="preview"
            sx={{ objectFit: "cover", maxHeight: 400 }}
           />
          </Card>
          <Stack spacing={2} alignItems="center">
           <FormControl component="fieldset" sx={{ mb: 2
}}>
            <FormLabel
             component="legend"
             sx={{
```

```
                fontFamily: "Comfortaa",
                fontWeight: "bold",
                color: "black",
                mb: 1,
              }}
            >
              Метод обробки:
            </FormLabel>
            <RadioGroup
              row
              value={processingMethod}
              onChange={(e) =>
setProcessingMethod(e.target.value)}
              sx={{
                justifyContent: "center",
                "& .MuiFormControlLabel-label": {
                  fontFamily: "Comfortaa",
                  fontSize: "14px",
                },
              }}
            >
              <FormControlLabel
                value="basic"
                control={<Radio sx={{ color: COLORS.pink3
}} />}
                label="Базовий (Ollama)"
              />
              <FormControlLabel
                value="advanced"
                control={<Radio sx={{ color: COLORS.pink3
}} />}
                label="Розширений (Ollama + обробка)"
              />
            </RadioGroup>
            {processingMethod === "advanced" && (
              <Typography
                variant="caption"
                sx={{
                  fontFamily: "Comfortaa",
                  fontSize: "12px",
                  color: COLORS.pink3,
                  mt: 1,
                  display: "block",
                }}
              >
                Включає: нормалізацію освітлення,
вирівнювання обличчя,
                сегментацію областей
              </Typography>
            )}
          </FormControl>
          <Button
            variant="outlined"
            color="secondary"
            onClick={handleProcess}
            disabled={loading}
            sx={{
              borderRadius: "4vw",
              borderColor: COLORS.pink3,
              color: "black",
              fontFamily: "Comfortaa",
              fontSize: "17px",
              fontWeight: "bold",
              backgroundColor: COLORS.pink0,
              px: 4,
              py: 1,
              textTransform: "none",
              ":hover": {
                color: COLORS.white,
                backgroundColor: COLORS.pink3,
              },
            }}
          >
            {loading ? "Обробка..." : "Обробити"}
          </Button>
          {loading && <CircularProgress sx={{ color:
COLORS.pink3 }} />}
        </Stack>
      </motion.div>
    )}

{step === 2 && (
      <motion.div
        key="result"
        initial={{ opacity: 0, y: 30 }}
        animate={{ opacity: 1, y: 0 }}
        exit={{ opacity: 0 }}
      >
        <Card
          sx={{
            p: 3,
            textAlign: "center",
            borderRadius: "20px",
            backgroundColor: COLORS.pink0,
            border: 2px solid ${COLORS.pink3},
          }}
        >
          <CardContent>

<AnimatedUnderlineText>Результат</AnimatedUnderlineT
ext>
            <Typography
              variant="h5"
              sx={{
                mt: 2,
                fontFamily: "Comfortaa",
                fontWeight: "bold",
                color: COLORS.pink3,
              }}
            >
              {COLOR_TYPES[result].name}
            </Typography>

            {COLOR_TYPES[result] && (
              <Box sx={{ mt: 2 }}>
                <Typography
                  variant="body1"
                  sx={{ fontFamily: "Comfortaa", mb: 1 }}
                >
                  {COLOR_TYPES[result].description}
                </Typography>
              </Box>
            )}

            <Stack spacing={2} alignItems="center" sx={{
mt: 4 }}>
```

```
                <Button
                  variant="outlined"
                  onClick={() => {
                    setFile(null);
                    setPreview(null);
                    setStep(0);
                    setResult(null);
                    setLoading(false);
                  }}
                  sx={{
                    borderRadius: "4vw",
                    borderColor: COLORS.pink3,
                    color: "black",
                    fontFamily: "Comfortaa",
                    fontSize: "17px",
                    fontWeight: "bold",
                    backgroundColor: COLORS.pink0,
                    px: 4,
                    py: 1,
                    textTransform: "none",
                    ":hover": {
                      color: COLORS.white,
                      backgroundColor: COLORS.pink3,
                    },
                  }}
                >
                  Повторити
                </Button>
              </Stack>
            </CardContent>
          </Card>
        </motion.div>
      )}
    </AnimatePresence>
  </Container>
 </Box>
 <Footer />
</Box>
);
};
```

image_processor.py

```
import cv2
import numpy as np
import base64
import json
import sys
from mediapipe import solutions
from mediapipe.framework.formats import landmark_pb2

# MediaPipe Face Mesh for face landmarks
mp_face_mesh = solutions.face_mesh
mp_drawing = solutions.drawing_utils



def retinex_normalization(image, sigma_list=[15, 80, 250]):
    """
    Multi-scale Retinex algorithm for illumination
normalization
    Normalizes lighting to reduce shadows and highlights
    """
    # Convert to float
    img = image.astype(np.float64) + 1.0

    # Apply Gaussian blur for each scale
    retinex = np.zeros_like(img)
    for sigma in sigma_list:
        gaussian = cv2.GaussianBlur(img, (0, 0), sigma)
        retinex += np.log10(img) - np.log10(gaussian)

    # Average across scales
    retinex = retinex / len(sigma_list)

    # Normalize to 0-255 range
    retinex = (retinex - retinex.min()) / (retinex.max() -
retinex.min()) * 255
    retinex = retinex.astype(np.uint8)

    return retinex


def detect_face_landmarks(image):
    """
    Detect face landmarks using MediaPipe Face Mesh
    Returns landmarks and face detection status
    """
    face_mesh = mp_face_mesh.FaceMesh(
        static_image_mode=True,
        max_num_faces=1,
        refine_landmarks=True,
        min_detection_confidence=0.5
    )

    # Convert BGR to RGB for MediaPipe
    rgb_image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
    results = face_mesh.process(rgb_image)

    if not results.multi_face_landmarks:
        return None, False

    # Get first face landmarks
    face_landmarks = results.multi_face_landmarks[0]

    # Extract key points (eyes, nose, mouth)
    h, w = image.shape[:2]
    landmarks = []

    # Left eye (33, 7, 163, 144, 145, 153, 154, 155, 133, 173,
157, 158, 159, 160, 161, 246)
    # Right eye (362, 382, 381, 380, 374, 373, 390, 249, 263,
466, 388, 387, 386, 385, 384, 398)
    # Nose tip (4)
    # Mouth corners (61, 291)

    key_indices = {
        'left_eye': [33, 7, 163, 144, 145, 153, 154, 155, 133, 173,
157, 158, 159, 160, 161, 246],
        'right_eye': [362, 382, 381, 380, 374, 373, 390, 249, 263,
466, 388, 387, 386, 385, 384, 398],
        'nose_tip': [4],
        'mouth_left': [61],
        'mouth_right': [291],
        'chin': [18],
        'forehead': [10]
```

```
    }

    landmarks_dict = {}
    for key, indices in key_indices.items():
        points = []
        for idx in indices:
            if idx < len(face_landmarks.landmark):
                landmark = face_landmarks.landmark[idx]
                x = int(landmark.x * w)
                y = int(landmark.y * h)
                points.append([x, y])
        if points:
            landmarks_dict[key] = np.array(points)

    return landmarks_dict, True

def align_face(image, landmarks):
    """
    Align face by rotating image so eyes are on the same horizontal line
    """
    if 'left_eye' not in landmarks or 'right_eye' not in landmarks:
        return image, None

    left_eye = landmarks['left_eye']
    right_eye = landmarks['right_eye']

    # Calculate eye centers
    left_eye_center = left_eye.mean(axis=0)
    right_eye_center = right_eye.mean(axis=0)

    # Calculate angle between eyes
    dy = right_eye_center[1] - left_eye_center[1]
    dx = right_eye_center[0] - left_eye_center[0]
    angle = np.degrees(np.arctan2(dy, dx))

    # Rotate image to align eyes horizontally
    h, w = image.shape[:2]
    center = (w // 2, h // 2)
    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
    aligned_image = cv2.warpAffine(image, rotation_matrix, (w, h), flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_REPLICATE)

    # Also rotate landmarks
    rotated_landmarks = {}
    for key, points in landmarks.items():
        rotated_points = []
        for point in points:
            point_homogeneous = np.array([point[0], point[1], 1])
            rotated_point = rotation_matrix @ point_homogeneous
            rotated_points.append(rotated_point)
        rotated_landmarks[key] = np.array(rotated_points)

    return aligned_image, rotated_landmarks


def create_skin_mask(image, landmarks):
    """
    Create mask for skin regions (excluding shadows, highlights, lips)
    Uses face mesh to identify skin areas
    """
    mask = np.zeros(image.shape[:2], dtype=np.uint8)
    h, w = image.shape[:2]

    if 'nose_tip' not in landmarks or 'chin' not in landmarks or 'forehead' not in landmarks:
        return mask

    # Define skin region (cheeks, forehead, neck - avoiding eyes, mouth, nose)
    # This is a simplified approach - in production, use full face mesh segmentation
    face_mesh = mp_face_mesh.FaceMesh(
        static_image_mode=True,
        max_num_faces=1,
        refine_landmarks=True,
        min_detection_confidence=0.5
    )

    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(rgb_image)

    if results.multi_face_landmarks:
        # Use face mesh to create skin mask
        face_landmarks = results.multi_face_landmarks[0]

        # Define skin triangles (simplified - cheeks area)
        # Cheek indices from MediaPipe Face Mesh
        cheek_indices = [
            [234, 227, 116], [117, 118, 119], [120, 121, 126], [142, 36, 205],
            [206, 92, 165], [167, 164, 39], [40, 185, 60], [75, 79, 209],
            [175, 170, 140], [136, 150, 149], [176, 148, 152]
        ]

        for triangle in cheek_indices:
            points = []
            for idx in triangle:
                if idx < len(face_landmarks.landmark):
                    landmark = face_landmarks.landmark[idx]
                    x = int(landmark.x * w)
                    y = int(landmark.y * h)
                    points.append([x, y])

            if len(points) == 3:
                pts = np.array(points, dtype=np.int32)
                cv2.fillPoly(mask, [pts], 255)

        # Remove shadows and highlights using threshold
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        _, shadow_mask = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        shadow_mask = cv2.bitwise_not(shadow_mask)

        # Remove very dark (shadows) and very bright (highlights) areas
        gray_masked = cv2.bitwise_and(gray, mask)
```

```
        mean_intensity = np.mean(gray_masked[gray_masked > 0])
        std_intensity = np.std(gray_masked[gray_masked > 0])

        # Remove pixels outside 2 standard deviations
        lower_bound = max(0, mean_intensity - 2 * std_intensity)
        upper_bound = min(255, mean_intensity + 2 * std_intensity)

        final_mask = cv2.inRange(gray, lower_bound, upper_bound)
        mask = cv2.bitwise_and(mask, final_mask)

        # Remove lip area (simplified - around mouth landmarks)
if 'mouth_left' in landmarks and 'mouth_right' in landmarks:
            mouth_points = np.array([
                landmarks['mouth_left'][0],
                landmarks['mouth_right'][0]
            ])
            cv2.ellipse(mask,
                        tuple(mouth_points.mean(axis=0).astype(int)),
                        (int(np.linalg.norm(mouth_points[1] - mouth_points[0]) / 2), 20),
                        0, 0, 360, 0, -1)

    return mask


def create_eye_mask(image, landmarks):
    """
    Create mask for iris only (excluding white of the eye)
    """
    mask = np.zeros(image.shape[:2], dtype=np.uint8)

    if 'left_eye' not in landmarks or 'right_eye' not in landmarks:
        return mask

    for eye_key in ['left_eye', 'right_eye']:
        eye_points = landmarks[eye_key]

        if len(eye_points) > 0:
            # Create ellipse around eye
            eye_center = eye_points.mean(axis=0).astype(int)

            # Calculate eye dimensions
            eye_width = np.linalg.norm(eye_points.max(axis=0) - eye_points.min(axis=0))
            eye_height = eye_width * 0.6  # Eyes are typically wider than tall

            # Create mask for iris (smaller than full eye)
            iris_width = int(eye_width * 0.6)
            iris_height = int(eye_height * 0.8)

            cv2.ellipse(mask, tuple(eye_center), (iris_width // 2, iris_height // 2), 0, 0, 360, 255, -1)

    return mask


def create_hair_mask(image, landmarks):
    """
    Create mask for hair region (middle zone, excluding roots and highlights)
    """
    mask = np.zeros(image.shape[:2], dtype=np.uint8)
    h, w = image.shape[:2]

    if 'forehead' not in landmarks:
        return mask

    # Estimate hair region above forehead
    forehead_y = landmarks['forehead'][0][1]
    hair_top = max(0, int(forehead_y * 0.3))
    hair_bottom = int(forehead_y * 0.9)

    # Create rectangular region for hair (middle zone)
    hair_left = max(0, int(w * 0.2))
    hair_right = min(w, int(w * 0.8))

    # Create initial mask
    cv2.rectangle(mask, (hair_left, hair_top), (hair_right, hair_bottom), 255, -1)

    # Remove highlights using intensity threshold
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray_masked = cv2.bitwise_and(gray, mask)

    if np.sum(gray_masked > 0) > 0:
        mean_intensity = np.mean(gray_masked[gray_masked > 0])

        # Remove very bright areas (highlights)
        highlight_threshold = min(255, mean_intensity + 50)
        highlight_mask = cv2.inRange(gray, 0, highlight_threshold)
        mask = cv2.bitwise_and(mask, highlight_mask)

        # Remove very dark areas (roots)
        root_threshold = max(0, mean_intensity - 40)
        root_mask = cv2.inRange(gray, root_threshold, 255)
        mask = cv2.bitwise_and(mask, root_mask)

    return mask


def extract_colors_from_region(image, mask, k=5):
    """
    Extract dominant colors from masked region using K-means clustering
    Returns list of RGB colors
    """
    # Get masked pixels
    masked_pixels = image[mask > 0]

    if len(masked_pixels) == 0:
        return []

    # Reshape for k-means
    pixels = masked_pixels.reshape(-1, 3).astype(np.float32)

    if len(pixels) < k:
```

```
        k = len(pixels)

    # Apply k-means
    criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
    _, labels, centers = cv2.kmeans(pixels, k, None, criteria,
10, cv2.KMEANS_RANDOM_CENTERS)

    # Convert to uint8
    centers = centers.astype(np.uint8)

    # Sort by frequency
    unique, counts = np.unique(labels, return_counts=True)
    sorted_indices = np.argsort(counts)[::-1]

    colors = []
    for idx in sorted_indices:
        colors.append(centers[idx].tolist())

    return colors

def process_image(base64_image):
    """
    Main processing function:
    1. Retinex normalization
    2. Face alignment
    3. Region segmentation and color extraction
    """
    try:
        # Decode base64 image
        image_data = base64.b64decode(base64_image)
        nparr = np.frombuffer(image_data, np.uint8)
        image = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        if image is None:
            return {'error': 'Could not decode image'}

        # Step 1: Retinex normalization
        normalized_image = retinex_normalization(image)

        # Step 2: Detect face landmarks
        landmarks, face_detected =
detect_face_landmarks(normalized_image)

        if not face_detected:
            return {'error': 'No face detected in image'}

        # Step 3: Face alignment
        aligned_image, aligned_landmarks =
align_face(normalized_image, landmarks)

        if aligned_landmarks is None:
            return {'error': 'Could not align face'}

        # Step 4: Create region masks
        skin_mask = create_skin_mask(aligned_image,
aligned_landmarks)
        eye_mask = create_eye_mask(aligned_image,
aligned_landmarks)
        hair_mask = create_hair_mask(aligned_image,
aligned_landmarks)

        # Step 5: Extract colors from each region
        skin_colors =
extract_colors_from_region(aligned_image, skin_mask, k=5)
        eye_colors =
extract_colors_from_region(aligned_image, eye_mask, k=3)
        hair_colors =
extract_colors_from_region(aligned_image, hair_mask, k=4)

        # Encode processed image back to base64
        _, buffer = cv2.imencode('.jpg', aligned_image)
        processed_image_base64 =
base64.b64encode(buffer).decode('utf-8')

        return {
            'success': True,
            'processed_image': processed_image_base64,
            'skin_colors': skin_colors,
            'eye_colors': eye_colors,
            'hair_colors': hair_colors
        }

    except Exception as e:
        return {'error': str(e)}


if name == 'main':
    # Read input from stdin
    input_data = json.loads(sys.stdin.read())
    base64_image = input_data.get('image')

    # Process image
    result = process_image(base64_image)

    # Output result as JSON
     print(json.dumps(result))
```

photo.controller.js

```
    const { exec } = require('child_process');
    const { promisify } = require('util');
    const path = require('path');
    const fs = require('fs').promises;

    const execAsync = promisify(exec);

    /**
     * Process image with advanced preprocessing (Retinex,
Face Alignment, Segmentation)
     * @param {Object} req - Express request object
     * @param {Object} res - Express response object
     */
    const processPhotoWithPreprocessing = async (req, res)
=> {
      try {
        const { image } = req.body;

        if (!image) {
          return res.status(400).json({ message: 'Image is
required' });
        }

        // Get the path to Python script
        const scriptPath = path.join(__dirname,
'../../image_processor.py');
```

```
        // Check if Python script exists
        try {
          await fs.access(scriptPath);
        } catch (error) {
          return res.status(500).json({
            message: 'Image processor script not found',
            error: error.message
          });
        }

        // Prepare input data
        const inputData = JSON.stringify({ image });

        // Execute Python script
        const { stdout, stderr } = await execAsync(
          python "${scriptPath}",
          {
            input: inputData,
            maxBuffer: 10 * 1024 * 1024, // 10MB buffer for large images
            encoding: 'utf8'
          }
        );

        if (stderr && !stderr.includes('DeprecationWarning')) {
          console.error('Python script stderr:', stderr);
        }

        // Parse result
        const result = JSON.parse(stdout);

        if (result.error) {
          return res.status(400).json({
            message: 'Image processing failed',
            error: result.error
          });
        }

        return res.status(200).json({
          success: true,
          data: result
        });

      } catch (error) {
        console.error('Error processing photo:', error);
        return res.status(500).json({
          message: 'Internal server error',
          error: error.message
        });
      }
    };

    module.exports = {
      processPhotoWithPreprocessing,
    };
```

constants.js

```
    export const PROJECT_NAME = "Color Muse";
    export const AUTH_LOGIN_ROUTE = "http://localhost:3001/color-muse/auth/login/";
    export const AUTH_REGISTER_ROUTE = "http://localhost:3001/color-muse/auth/register/";
    export const AI_GENERATE_ROUTE = "http://localhost:11434/api/generate";
    export const AI_DETECT_COLORTYPE_PROMPT = "http://localhost:11434/api/generate";
    export const PHOTO_PROCESS_ROUTE = "http://localhost:3001/color-muse/photo/process";
    export const AI_DETECT_PROMPT = `Determine only the color type of the person in the photo.
    Bright Winter
    Eyes:
    68 117 137
    47 147 137
    129 123 80
    Skin:
    248 210 197
    235 215 204
    223 183 164
    207 169 148
    Hair color:
    59 48 38
    84 76 69
    34 29 27
    22 18 19

    Deep Winter
    Eyes:
    59 60 42
    70 78 65
    33 30 21
    16 19 24
    Skin:
    247 221 198
    228 183 142
    224 165 109
    172 135 108
    Hair:
    40 32 21
    73 64 65
    25 22 17
    51 49 54

    Natural Winter
    Eyes:
    93 119 136
    91 85 69
    88 74 71
    50 67 77
    Skin:
    251 241 240
    235 215 204
    244 212 201
    241 207 182
    231 178 160
    Hair:
    85 78 70
    28 30 29
    40 36 37

    Soft Summer
    Eyes:
    71 87 102
    109 142 159
```

100 120 118
131 142 146
Skin:
234 216 204
225 206 189
230 195 172
242 201 179
244 185 155
218 181 155
Hair:
73 59 50
139 126 120
96 81 74
125 108 100

Light Summer
Eyes:
121 139 151
111 143 158
32 59 78
98 131 136
Skin:
252 234 220
235 215 204
241 207 182
243 200 183
244 185 155
227 163 136
Hair:
223 211 195
184 167 149

Natural Summer
Eyes:
65 81 96
115 151 173
88 122 123
59 76 83
Skin:
246 221 201
233 211 198
230 179 160
218 181 154
198 167 147
Hair:
85 73 61
144 125 118
82 73 66

Soft Autumn
Eyes:
113 90 49
81 54 37
92 71 50
Skin:
225 206 189
241 207 182
242 215 188
217 181 155
223 177 128
Hair:
189 146 103
104 67 41
158 128 104
220 161 131

Deep Autumn
Eyes:
86 68 54
43 37 25
68 51 41
40 12 0
Skin:
244 214 186
225 166 108
182 112 60
Hair:
55 36 21
99 76 58
88 52 36
67 60 54

Natural Autumn
Eyes:
64 38 21
144 121 89
109 74 44
81 52 36
Skin color:
245 215 181
217 186 155
215 180 150
222 171 128
144 107 78
Hair:
169 130 89
122 95 76
191 134 89
147 87 59

Bright Spring
Eyes:
89 111 124
108 136 139
120 98 74
94 71 63
Skin:
254 217 188
255 201 170
242 285 258
142 87 56
61 31 21
44 25 18
Hair:
214 175 136
115 81 54
126 93 84
79 68 66

Light Spring
Eyes:
156 178 191
141 153 141
185 193 170
201 180 175
Skin:

```
    254 217 190
    252 205 187
    237 193 164
    243 209 172
    232 188 143
    211 158 124
    Hair:
    216 193 159
    236 205 150
    201 142 112
    143 109 82

    Natural Spring
    Eyes:
    107 132 139
    174 215 199
    215 209 185
    213 216 187
    Skin:
    251 205 189
    243 207 171
    230 184 135
    226 172 134
    221 162 118
    172 116 79
    Hair:
    180 128 80
    230 173 146
    191 94 62
    165 128 101

    You have the following options: bright_spring,
warm_spring, light_spring, light_summer, cool_summer,
soft_summer, warm_autumn, deep_autumn, soft_autumn,
bright_winter, cool_winter, deep_winter.
    Respond only with a name from this list, no additional
text.
    `;
```

auth.controller.js

```
    const bcryptjs = require("bcrypt");
    const User = require("../models/user.model.js");
    const jwt = require("jsonwebtoken");
    const apiError = require("../utils/apiError.js");

    const register = async (req, res, next) => {
      try {
        if (!req.body.password) {
          throw new apiError(400, "Bad request");
        }
        const salt = await bcryptjs.genSalt(10);
        const hashedPassword = await
bcryptjs.hash(req.body.password, salt);
        const newUser = new User({
          name: req.body.name,
          email: req.body.email,
          isFemale: req.body.isFemale,
          password: hashedPassword,
        });

        const savedUser = await newUser.save();
        return res.status(201).json(savedUser);
      } catch (err) {
        return next(err);
      }
    };

    const login = async (req, res, next) => {
      try {
        if (!req.body.email || !req.body.password) {
          throw new apiError(400, "Email,password required");
        }

        const user = await User.findOne({ email:
req.body.email }).select(
          "name email password"
        );
        console.log({ user });
        if (!user) {
          throw new apiError(404, "User was not found");
        }
        const isPasswordCorrect = await bcryptjs.compare(
          req.body.password,
          user.password
        );
        if (!isPasswordCorrect) {
          throw new apiError(401, "Password is incorrect");
        }
        const payload = {
          id: user.id,
          name: user.name,
        };
        const token = jwt.sign(payload,
process.env.JWT_SECRET, {
          expiresIn: "1d",
        });
        return res
          .cookie("access_token", token, {
            httpOnly: true,
          })
          .status(200)
          .json({ message: "login success", access_token: token
});
      } catch (err) {
        return next(err);
      }
    };

    const logout = (req, res) => {
      res.clearCookie("access_token");
      return res.status(200).json({ message: "Logout success"
});
    };

    module.exports = {
      register,
      login,
      logout,
    };
```

app.jsx

```
    import { Route, Routes } from "react-router-dom";
    import { HomePage } from "./pages/HomePage.jsx";
```

```
import { Faq } from "./pages/Faq.jsx";
import { SignIn } from "./pages/SignIn.jsx";
import { SignUp } from "./pages/SignUp.jsx";
import { Colortypes } from "./pages/Colortypes.jsx";
import { Test } from "./pages/Test.jsx";
import { PhotoDetect } from "./pages/PhotoDetect.jsx";

function App() {
 return (
  <Routes>
   <Route path="/faq" element={<Faq />} />
   <Route path="/colortypes" element={<Colortypes />}
/>
   <Route path="/signin" element={<SignIn />} />
   <Route path="/signup" element={<SignUp />} />
   <Route path="/test" element={<Test />} />
   <Route path="/photo-detect" element={<PhotoDetect
/>} />
   <Route path="*" element={<HomePage />} />
  </Routes>
 );
}

export default App;
```