ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ
(повне назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи

бакалавра

(освітньо-кваліфікаційний рівень)

на тему: «Розробка застосунку для підбору індивідуальних б’юті-рекомендацій на основі аналізу рис обличчя»

Виконала: здобувачка *4* курсу, групи *4ПР5*
напряму підготовки (спеціальності)
121 «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки, спеціальності)
Грудінкіна Марина Володимирівна
(прізвище та ініціали)
Керівник *к.т.н., доцент Огнєва О. Є.*
(прізвище та ініціали)
Рецензент *к.т.н., доцент Вишемирська С. В.*
(прізвище та ініціали)

Хмельницький – 2025

ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Інститут, факультет, відділення *Інформаційних технологій та дизайну*
Кафедра, циклова комісія *Програмних засобів і технологій*
Освітньо-кваліфікаційний рівень *бакалавр*
ОПП *Програмування смарт-пристроїв та вбудованих систем*
(шифр і назва)
Спеціальність *121 – Інженерія програмного забезпечення*
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри *програмних засобів і технологій*
к.т.н., доцент О. Є. Огнєва
«_20_» ____січня__ 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Грудінкіна Марина Володимирівна
(прізвище, ім'я, по батькові)

1. Тема проєкту «Розробка застосунку для підбору індивідуальних б'юті-рекомендацій на основі аналізу рис обличчя»

керівник проєкту кандидат технічних наук, доцент Огнєва Оксана Євгенівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» січня 2025 року № 98-с

2. Строк подання студентом проєкту 16.06.2025

3. Вихідні дані до проєкту Технічні вимоги до мобільного додатку, функціональні вимоги: аналіз рис обличчя, визначення типів зовнішності, формування персоналізованих б'юті-рекомендацій. Підтримка чотирьох мов інтерфейсу, адаптивний дизайн та робота в офлайн-режимі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області
2. Проєктування системи
3. Програмна реалізація
4. Тестування та оцінювання якості

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Архітектурна діаграма мобільного додатку. Блок-схема алгоритму аналізу рис обличчя. Діаграма взаємодії компонентів системи. Макети ключових екранів користувацького інтерфейсу. Графіки оцінки продуктивності та точності рекомендацій.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання: _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційного Проєкту (роботи)	Строк виконання етапів проєкту	Примітка
1	Отримання завдання	09.02.2025	Виконано
2	Підбір літератури	10.02.2025 – 25.02.2025	Виконано
3	Аналіз предметної області	28.02.2025 – 10.03.2025	Виконано
4	Розробка концептуальної моделі	12.03.2025 – 22.03.2025	Виконано
5	Розробка проєктної системи	25.03.2025 – 30.03.2025	Виконано
6	Розробка та проєктування баз даних	02.04.2025 – 10.04.2025	Виконано
7	Розробка інтерфейсу додатка	12.04.2025 – 20.04.2025	Виконано
8	Написання вихідного коду програми	21.04.2025 – 20.05.2025	Виконано
9	Оформлення пояснювальної записки	21.05.2025- 01.06.2025	Виконано
10	Захист кваліфікаційної роботи	16.06.2025	Виконано

Здобувачка _____ М.В. Грудінкіна
(підпис) (прізвище та ініціали)

Керівник проєкту _____ О.Є. Огнєва
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 129 сторінок, 59 рисунків, 28 таблиць, 15 формул, 4 додатки, 41 джерело.

Мета роботи — розробка додатку для підбору індивідуальних б'юті-рекомендацій на основі аналізу рис обличчя. Робота спрямована на створення ефективного та зручного інструментарію, що допоможе користувачам отримувати рекомендації з макіяжу, догляду за шкірою та підбору зачісок з використанням технологій комп'ютерного зору та машинного навчання.

Об’єкт дослідження — процес надання персоналізованих рекомендацій щодо вибору косметичних засобів на основі аналізу рис обличчя.

Предмет дослідження — методи та алгоритми аналізу рис обличчя та формування персоналізованих б'юті-рекомендацій. Методи дослідження — аналіз та узагальнення інформації про існуючі мобільні додатки для підбору б'юті-рекомендацій; теоретичний аналіз та узагальнення інформаційних джерел з тематики комп'ютерного зору та машинного навчання; моделювання та проєктування алгоритмів аналізу зовнішності та формування рекомендацій.

Результат роботи: 1) Створено додаток для надання персоналізованих б'юті-рекомендацій; 2) Розроблено алгоритм аналізу рис обличчя; 3) Створено базу даних косметичних засобів та правил формування рекомендацій; 4) Розроблено систему персоналізації рекомендацій з урахуванням індивідуальних особливостей користувача; 5) Проведено тестування системи та оцінку якості наданих рекомендацій.

Новизна роботи: розроблено комплексний підхід до автоматизованого аналізу рис обличчя та формування рекомендацій з використанням сучасних технологій комп'ютерного зору та машинного навчання; запропоновано новий алгоритм визначення типу зовнішності на основі аналізу ключових параметрів обличчя та розроблено систему формування рекомендацій.

Ключові слова: *мобільний додаток, комп’ютерний зір, машинне навчання, б’юті-рекомендації, аналіз рис обличчя, персоналізація, Flutter, ML Kit Face Detection*.

АНОТАЦІЯ

Кваліфікаційна робота бакалавра містить наступні структурні частини: вступ, чотири розділи, висновки, список використаних джерел та додатки.

Перший розділ **«Дослідження предметної області»** охоплює аналіз типології зовнішності, класифікації косметичних засобів та принципів формування індивідуальних рекомендацій. Розглянуто методи аналізу зовнішності та сучасні технології визначення характеристик обличчя. Проведено огляд існуючих рішень, включаючи аналіз переваг та обмежень систем. Сформульовано функціональні, нефункціональні та технічні вимоги.

Другий розділ **«Проєктування системи»** присвячено архітектурному проєктуванню мобільного застосунку на основі патерну MVVM з використанням фреймворку GetX. Розроблено структуру бази даних з використанням NoSQL-сховища Hive. Спроєктовано алгоритм аналізу рис обличчя, що включає етапи попередньої обробки зображення, детекцію ключових точок та класифікацію. Розроблено систему персоналізованих рекомендацій з механізмом адаптивного навчання та спроєктовано UI.

Третій розділ **«Програмна реалізація»** описує вибір технологічного стеку та структурну організацію програмного коду за принципом «feature-first». Представлено реалізацію модуля аналізу обличчя з використанням Google ML Kit Face Detection, включаючи алгоритми обробки зображень та класифікації характеристик. Розроблено систему формування рекомендацій з базою знань структурованих правил. Реалізовано користувацький інтерфейс з п'яти основних екранів та адміністративну панель для управління контентом.

Четвертий розділ **«Тестування та оцінювання якості»** містить результати тестування програмного продукту, включаючи модульне, інтеграційне та функціональне тестування. Досліджено точність рекомендацій з використанням метрик та експертного оцінювання. Здійснено оптимізацію продуктивності та оцінку релевантності системи методом A/B тестування.

Результати роботи демонструють ефективність розробленої системи, що забезпечує релевантність косметологічних рекомендацій для користувачів.

ABSTRACT

The bachelor's qualification work contains the following structural parts: introduction, four chapters, conclusions, references, and appendices.

The first chapter **«Subject Area Research»** encompasses analysis of appearance typology, cosmetic product classification, and principles of individual recommendation formation. Methods of appearance analysis and modern technologies for facial characteristic determination are examined. A review of existing solutions is conducted, including analysis of system advantages and limitations. Functional, non-functional, and technical requirements are formulated.

The second chapter **«System Design»** is dedicated to architectural design of the mobile application based on the MVVM pattern using the GetX framework. Database structure using Hive NoSQL storage is developed. The facial analysis algorithm is designed, including image preprocessing stages, key point detection, and classification. A personalized recommendation system with adaptive learning mechanism and UI are developed.

The third chapter **«Software Implementation»** describes technology stack selection and structural code organization following the «feature-first» principle. Implementation of the facial analysis module using Google ML Kit Face Detection is presented, including image processing algorithms and characteristic classification. A recommendation system with structured rules knowledge base is developed. User interface with five main screens and administrative panel for content management are implemented.

The fourth chapter **«Testing and Quality Assessment»** contains software testing results, including unit, integration, and functional testing. Recommendation accuracy is investigated using metrics and expert evaluation. Performance optimization and system relevance assessment using A/B testing are conducted.

The work results demonstrate the effectiveness of the developed system, ensuring relevance of cosmetic recommendations for users.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API (Application Programming Interface) – програмний інтерфейс застосунку, набір визначень взаємодії різнотипного програмного забезпечення.

AR (Augmented Reality) – доповнена реальність, технологія, що дозволяє накладати віртуальні об’єкти на реальне зображення.

CIELab – колірний простір, розроблений Міжнародною комісією з освітлення, що враховує сприйняття кольору людським оком.

CPU (Central Processing Unit) – центральний процесор, основний компонент комп'ютера, що виконує обчислення.

FCI (Functional Completeness Index) – індекс функціональної повноти, метрика для оцінки відповідності системи функціональним вимогам.

GLCM (Gray Level Co-occurrence Matrix) – матриця сумісної появи рівнів сірого, метод для аналізу текстурних характеристик зображення.

HOG (Histogram of Oriented Gradients) – гістограма орієнтованих градієнтів, метод для виявлення об’єктів у комп’ютерному зорі.

HSV (Hue, Saturation, Value) – колірна модель, що описує колір через компоненти тону, насиченості та яскравості.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript.

MAE (Mean Absolute Error) – середня абсолютна похибка, метрика для оцінки точності прогнозування.

ML (Machine Learning) – машинне навчання, розділ штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися на даних.

MTCNN (Multi-task Cascaded Convolutional Networks) – каскадні згорткові нейронні мережі для виявлення та локалізації обличчя.

MVVM (Model-View-ViewModel) – шаблон проєктування архітектури програмного забезпечення, що розділяє дані, логіку та відображення.

NDCG (Normalized Discounted Cumulative Gain) – нормалізований дисконтований кумулятивний приріст, метрика для оцінки ефективності систем рекомендацій.

NoSQL (Not Only SQL) – термін, що об'єднує ряд підходів до проєктування баз даних, відмінних від реляційної моделі.

PI (Performance Index) – інтегральний показник ефективності роботи системи

RCR (Recommendation Conversion Rate) – коефіцієнт конверсії рекомендацій, показник ефективності системи рекомендацій.

RMSE (Root Mean Square Error) – середньоквадратична помилка, статистична метрика для оцінки точності моделі.

RGB (Red, Green, Blue) – адитивна колірна модель, що описує колір через комбінацію червоного, зеленого та синього кольорів.

SDK (Software Development Kit) – набір інструментів для розробки програмного забезпечення.

SPF (Sun Protection Factor) – фактор захисту від сонця, показник ефективності сонцезахисних засобів.

TTC (Time To Complete) – час виконання завдання, метрика швидкодії системи.

UI (User Interface) – користувацький інтерфейс, засоби взаємодії користувача з системою.

UEI (User Experience Index) – індекс користувацького досвіду, комплексний показник для оцінки взаємодії з системою.

UX (User Experience) – досвід користувача, враження від взаємодії з системою

QSI (Quality System Index) – індекс якості системи, комплексний показник для оцінки нефункціональних вимог.

SUS (System Usability Scale) – шкала зручності використання системи, методика оцінки юзабіліті.

ІПО – індекс пропорційності обличчя, співвідношення висоти до ширини обличчя.

ОЗП – оперативний запам’ятовуючий пристрій (оперативна пам’ять).

ПАР – поверхнево-активні речовини, компоненти очищувальних засобів.

ВСТУП

Сучасний розвиток технологій штучного інтелекту та комп'ютерного зору відкриває нові можливості для персоналізації послуг у різних сферах, включаючи індустрію краси. Зростаючий попит на індивідуальний підхід у виборі косметичних засобів та процедур створює потребу в інтелектуальних системах, здатних надавати персоналізовані рекомендації на основі аналізу індивідуальних особливостей зовнішності користувача.

За даними досліджень Global Market Insights, світовий ринок косметичних продуктів у 2023 році оцінювався у 430 мільярдів доларів США, з прогнозованим зростанням до 550 мільярдів до 2027 року. При цьому понад 60% споживачів відзначають складність самостійного підбору косметичних засобів, що відповідають їхнім індивідуальним особливостям. Це підкреслює актуальність розробки інтелектуальних систем підтримки прийняття рішень у сфері індивідуального підбору косметичних засобів.

Мета і задачі дослідження. Метою роботи є створення мобільного додатку для надання персоналізованих б'юті-рекомендацій на основі автоматизованого аналізу рис обличчя користувача. Для досягнення поставленої мети необхідно вирішити такі завдання:

- розробити алгоритм аналізу рис обличчя з використанням технологій комп'ютерного зору;
- створити базу даних косметичних засобів та правил формування рекомендацій;
- розробити систему персоналізації рекомендацій з урахуванням індивідуальних особливостей користувача;
- реалізувати зручний користувацький інтерфейс мобільного додатку;
- провести тестування системи та оцінку якості наданих рекомендацій.

Об'єкт дослідження – процес автоматизованого надання персоналізованих рекомендацій щодо вибору косметичних засобів.

Предмет дослідження – методи та алгоритми аналізу рис обличчя та формування персоналізованих б'юті-рекомендацій.

Методи дослідження. У роботі використані методи комп'ютерного зору для аналізу зображень, алгоритми машинного навчання для класифікації типів зовнішності, методи проєктування баз даних та розробки мобільних додатків. Для оцінки якості рекомендацій застосовано методи статистичного аналізу та експертного оцінювання.

Наукова новизна полягає у розробці комплексного підходу до автоматизованого аналізу рис обличчя та формування персоналізованих рекомендацій з використанням сучасних технологій комп’ютерного зору та машинного навчання. Запропоновано новий алгоритм визначення типу зовнішності на основі аналізу ключових параметрів обличчя та розроблено адаптивну систему формування рекомендацій.

Практичне значення роботи полягає у створенні готового до використання мобільного додатку, який допоможе користувачам у виборі косметичних засобів та процедур, що найкраще відповідають їхнім індивідуальним особливостям. Система може бути корисною як для індивідуального використання, так і для професійних косметологів та консультантів.

Апробація результатів. Розроблений мобільний додаток пройшов тестування з залученням фокус-групи користувачів та отримав позитивні відгуки.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основи типології зовнішності та косметичних рекомендацій

У сучасній косметології та стилістиці виділяють кілька основних систем класифікації типів зовнішності. Однією з найпоширеніших є сезонна теорія колірних типів [1], яка базується на природному поєднанні кольору шкіри, очей і волосся людини. Згідно з цією теорією, виокремлюють чотири основні типи зовнішності: «Весна», «Літо», «Осінь» і «Зима», як проілюстровано на рисунку 1.1. За результатами досліджень [2], розподіл типів зовнішності серед європейської популяції такий: «Зима» – 28%, «Літо» – 31%, «Весна» – 19%, «Осінь» – 22%.

Рисунок 0.1 – Основні сезонні кольоротипи та їхні колірні гами

Кожен тип зовнішності характеризується комплексом параметрів, що визначають принципи підбору косметичних засобів:

Таблиця 1.1 – Характеристики основних типів зовнішності

Тип	Характеристики	Рекомендовані кольори	Рекомендації
Весна	Теплий підтон шкіри, золотисте волосся, прозорість шкіри	Персиковий, кораловий, золотистий, м'ятний	Легкі текстури, прозорі відтінки, природний рум'янець
Літо	Холодний підтон шкіри, попелясте волосся, м'якість кольорів	Холодний рожевий, лавандовий, сірий, блакитний	М'які приглушені відтінки, ніжні рум'яна, холодна гама
Осінь	Теплий підтон шкіри, рудувате волосся, висока насиченість	Теракотовий, хакі, гірчичний, мідний	Теплі відтінки, матові текстури, бронзери
Зима	Холодний підтон шкіри, висока контрастність, чіткість рис	Білий, чорний, фуксія, синій, смарагдовий	Яскраві контрастні кольори, чіткі контури, насичені помади

Важливо зазначити, що не всі люди мають чітко виражені характеристики одного типу. За даними досліджень [3], приблизно кожна шоста людина демонструє ознаки кількох кольоротипів. Такі випадки трапляються за поєднання суміжних сезонів, наприклад, «Весна – Літо» або «Осінь – Зима». Крім класичної сезонної, існує також і безліч інших систем типування з іншими факторами розподілу.

Визначення типу зовнішності може здійснюватися різними методами, кожен з яких має свої особливості та сфери застосування:

1. Кольорове драпірування;
2. Комп'ютерна діагностика;
3. Експертна оцінка.

Під час автоматизованої типізації використовують кількісні метрики:

- контрастність: різниця яскравості світлих і темних зон обличчя;
- теплота: числове вираження теплих або холодних пігментів у шкірі;
- насиченість: міра інтенсивності кольорів шкіри, волосся та очей;
- світлота: загальний показник яскравості кольоротипу.

Процес формування таргетованих косметичних рекомендацій базується на комплексному аналізі індивідуальних характеристик зовнішності та застосуванні системного підходу до їхньої інтерпретації. Ключовими факторами їхньої ефективності підбору є колористичні особливості, а також морфологія обличчя.

Під час формування гайдлайнів виділяють три основоположні принципи. Першим є колірна гармонізація, що передбачає врахування природного кольоротипу, включно з підтоном шкіри, кольором очей і волосся, визначення колірної температури косметичних засобів і підбір збалансованої палітри відтінків декоративної косметики. Другим принципом є морфологічна відповідність, яка базується на аналізі геометрії обличчя. Третім принципом виступає функціональна доцільність, що охоплює оцінку текстурних ознак шкіри та фактори сезонності. Під час реалізації колірної гармонізації застосовується математична модель оцінки [5]:

$$C_{comp} = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2} \quad (1.1)$$

де: C_{comp} – коефіцієнт колірної сумісності (0 – 100);

L_1, L_2 – показники яскравості порівнюваних відтінків (0 – 100);

a_1, a_2 – координати в червоно-зеленому спектрі (-128 до +127);

b_1, b_2 – координати в синьо-жовтому спектрі (-128 до +127).

Таблиця 1.2 – Інтерпретація результатів колірної сумісності C_{comp}

Діапазон	Результат
$0 \leq C_{comp} < 5$	Відмінна сумісність
$5 \leq C_{comp} < 15$	Хороша сумісність
$15 \leq C_{comp} < 25$	Допустима сумісність
$C_{comp} \geq 25$	Несумісні кольори

Морфологічний аналіз обличчя базується на визначенні ключових співвідношень і пропорцій. Основними параметрами є індекс пропорційності обличчя (ІПЛ) і рейтинг доцільності (R):

$$\text{ІПЛ} = \frac{h}{w} \tag{1.2}$$

де: h – висота обличчя, мм;

w – ширина обличчя в ділянці вилиць, мм.

Таблиця 1.2 – Інтерпретація результатів індексу пропорційності обличчя

Діапазон	Результат
1 – 1,2	Широке обличчя
1,3 – 1,4	Ідеальні пропорції
1,5 – 1,7	Подовжене обличчя

Функціональна доцільність рекомендацій оцінюється через систему вагових коефіцієнтів:

$$R = \sum_{i=1}^{n} w_i f_i \tag{1.3}$$

де: R – загальний рейтинг рекомендації;

w_i – ваговий коефіцієнт і-го фактора;

f_i – значення і-го фактора;

n – кількість врахованих факторів.

Важливим аспектом у даному прикладному рішенні також є і врахування змін, відповідно до природних циклів. У таблиці 1.3 наведено параметри сезонної корекції рекомендацій:

Таблиця 1.3 – Параметри сезонної корекції рекомендацій

Сезон	Корекція текстур	Корекція відтінків	Захисні фактори
Зима	+30%	-15%	SPF 15 - 30
Весна	-20%	+10%	SPF 30 - 50
Літо	-40%	+20%	SPF 50+
Осінь	+15%	+5%	SPF 15 - 30

Процес формування індивідуальних рекомендацій є ітеративним [6], що видно на блок-схемі алгоритму, що є в Додатку А.

Для демонстрації роботи алгоритму розглянемо конкретний випадок:

Дано:	Рішення:
$L_1 = 75, a_1 = 10, b_1 = 15$ $L_2 = 72, a_2 = 12, b_2 = 18$ $h = 180$ мм $w = 130$ мм	$C_{comp} = \sqrt{(75-72)^2 + (10-12)^2 + (15-18)^2} = 5{,}2.$ $\text{ІПЛ} = \frac{180 \text{ мм}}{130 \text{ мм}} = 1{,}38.$ $R = 0{,}4 \cdot 0{,}9 + 0{,}3 \cdot 0{,}95 + 0{,}2 \cdot 0{,}85 + 0{,}1 \cdot 0{,}8 = 0{,}89.$
Знайти: $C_{comp}, \text{ІПЛ}, R-?$	Отже, було досягнуто хорошої колірної сумісності. Отриманий рейтинг підтверджує високу відповідність рекомендації заданим критеріям.

Сучасна косметологія пропонує широкий асортимент засобів для догляду за шкірою та створення естетичного вигляду. Розуміння класифікації та властивостей є ключовим аспектом для розробки системи рекомендаційних правил. Основним критерієм систематизації косметичних засобів виступає функціональне призначення, що дає змогу виокремити фундаментальні категорії: засоби догляду, декоративну косметику та засоби для демакіяжу.

Засоби догляду формують найбільший сегмент ринку та забезпечують підтримку здорового стан у шкіри. Дослідження Journal of Cosmetic Science [9] свідчать, що ефективність цих засобів значною мірою залежить від концентрації активних компонентів. Очищувальні засоби містять поверхнево-активні речовини. Тонізувальні продукти відновлюють природний pH-баланс шкіри та підвищують ефективність догляду. Зволожувальні засоби, представлені кремами та сироватками, забезпечують гідратацію завдяки гіалуроновій кислоті, гліцерину та церамідам.

Декоративна косметика становить 35% ринку і демонструє найбільшу інноваційну активність. За даними міжнародного видання Cosmetics & Toiletries [10], сучасні формули включають компоненти з адаптивними властивостями. Наприклад, технологія під назвою shade-adjusting pigments дає змогу тональним засобам підлаштовуватися під тон шкіри.

Таблиця 1.4 – Основні категорії та властивості косметичних засобів

Категорія	Ключові компоненти	Властивості	Ефективність, %	Тривалість дії, год
Засоби догляду	Гіалуронова кислота (0,1 – 2%), вітаміни	Зволоження, захист	75 – 92	8 – 12
Декоративна косметика	Пігменти, полімери (3 – 8%), емоленти	Маскування, корекція	82 – 95	12 – 24
Засоби демакіяжу	Олії (10 – 30%), ПАР (2 – 5%), емульгатори	Очищення	95 – 99	Миттєва

1.2. Методи та технології аналізу зовнішності

Визначення типу зовнішності є фундаментальним етапом у формуванні косметичних рекомендацій. Класичні методи, що сформувалися протягом розвитку індустрії краси, базуються на аналізі колористичних і морфологічних характеристик зовнішності. Їхня оціночна точність значною мірою залежить від умов проведення дослідження.

Метод колірного драпірування [11] вважається найбільш усталеним і широко застосовуваним у професійній практиці. Його суть полягає в послідовному тестуванні впливу тканин різних відтінків на загальне сприйняття зовнішності. Процедура починається з підготовки, яка передбачає повне очищення шкіри від косметики. Надзвичайно важливим фактором є забезпечення нейтрального освітлення, що дає змогу точно оцінювати колірні нюанси. Приміщення має бути обладнане білим фоном із коефіцієнтом відбиття 90 – 95%, а рівень освітленості має становити 800 - 1000 люкс.

Основний етап тестування методом колірного драпірування характеризується високою складністю і вимагає значного досвіду від фахівця. У процесі аналізу послідовно застосовуються драпірування сезонних палітр, водночас фіксуються найменші зміни в кольорі шкіри та виразності рис. Увагу приділяють реакції на теплі та холодні відтінки, що є чинником у визначенні колірного типу.

Метод природних індикаторів є другим за значимістю підходом до визначення типу зовнішності. Він базується на аналізі вроджених колористичних характеристик. Основними параметрами оцінки виступають підтон шкіри, природний колір волосся, відтінок очей та їхня світлота. Особливість цього методу становить його незалежність від зовнішніх умов. Для забезпечення об'єктивності оцінки при застосуванні методу природних індикаторів розроблено спеціальну систему бальної оцінки [12]. Кожна характеристика аналізується з урахуванням похибки вимірювань. Сумарний показник типу зовнішності розраховується так:

$$T_{\text{зовнішності}} = \sum_{i=1}^{n} (B_i \pm \Delta B_i) \tag{1.4}$$

де: $T_{\text{зовнішності}}$ – загальний показник типу;

B_i – бали за і-ту характеристику;

ΔB_i – похибка вимірювання і-тої характеристики;

n – кількість оцінюваних параметрів.

Метод пропорційного аналізу є найбільш технічно складним серед класичних методів визначення типу зовнішності. Він базується на комплексному вивченні геометричних характеристик обличчя і потребує використання високоточних інструментів. Увагу приділяють аналізу, оцінці симетричності та виразності контурів. Ключовим показником тут виступає індекс пропорційності [12], який обчислюється за формулою 1.2, раніше згаданою в розділі 1.1.

На точність результатів класичних методів визначення типу зовнішності суттєво впливають зовнішні чинники. Відхилення рівня освітленості від нормативних показників на кожні 100 люкс знижує точність визначення загалом на 1 - 2%.

Технології CV відіграють ключову роль у розвитку сучасних систем аналізу зовнішності. Їхнє впровадження дає змогу автоматизувати процес визначення індивідуальних особливостей зовнішності та забезпечити об'єктивність оцінювання характеристик обличчя.

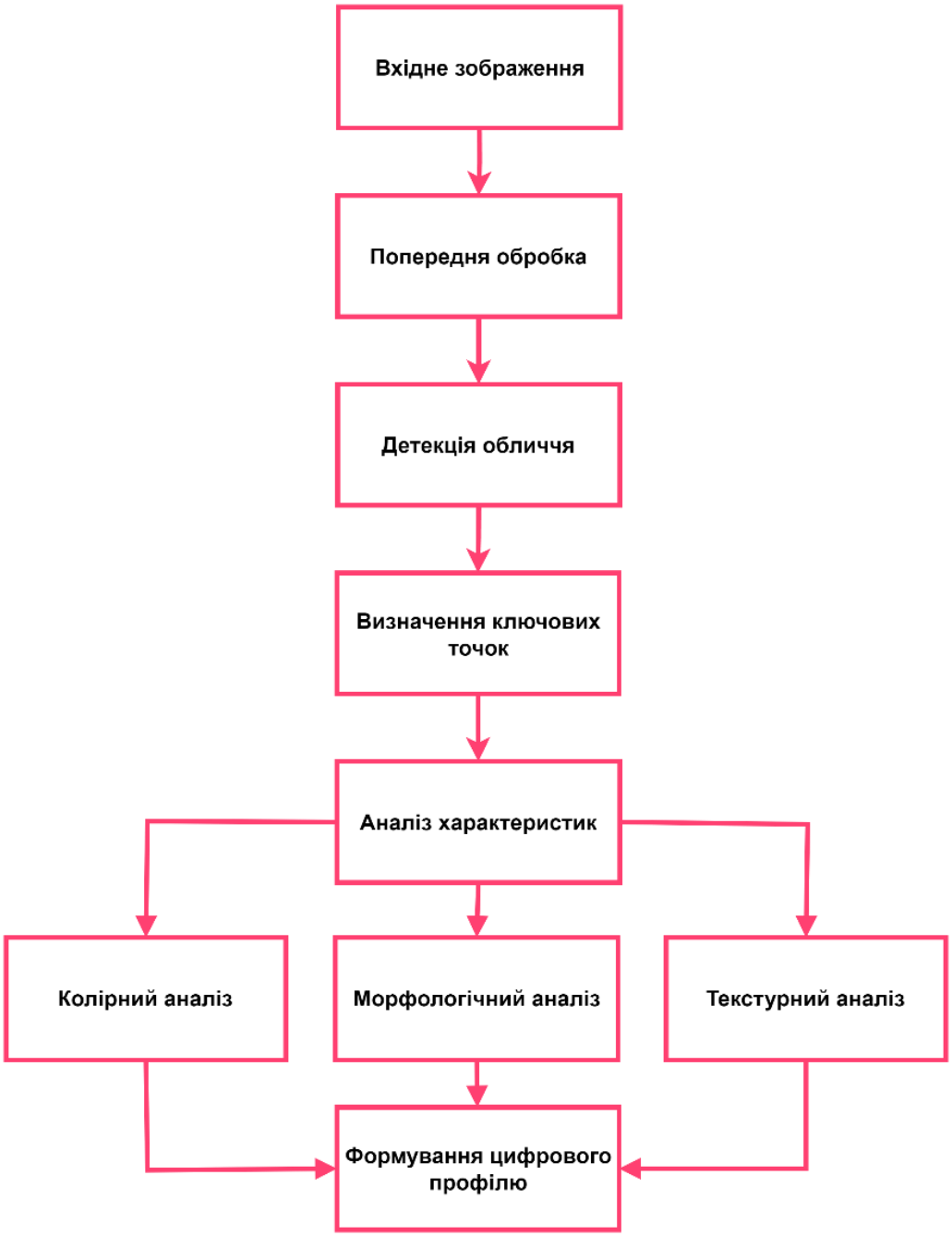


Рисунок 1.3 – Структурна схема процесу аналізу зображення

Першим етапом процесу аналізу є попереднє опрацювання зображення, що включає нормалізацію освітлення із застосуванням γ-корекції та вирівнювання гістограми, корекцію геометричних викривлень за допомогою аффінних перетворень і фільтрацію шумів із використанням фільтрів. Другий етап полягає в детекції обличчя і передбачає виконання операцій виявлення області, що цікавить, за допомогою методу Віоли-Джонса, локалізацію ключових точок каскадними класифікаторами та побудову геометричної моделі.

Таблиця 1.5 – Обчислювальна складність алгоритмів CV [13].

Алгоритм	Часова складність	Просторова складність
Віола-Джонс	$O(N\ logN)$	$O(N)$
AOM	$O(k\ logN^2)$	$O(kN)$
GLCM	$O(N^2)$	$O(k^2)$
Кольорові перетворення	$O(N)$	$O(1)$

Математична модель процесу розпізнавання базується на поданні зображення як матриці інтенсивностей пікселів:

$$I(x,y) = \begin{pmatrix} i_{11} & i_{21} & \cdots & i_{1n} \\ i_{21} & i_{22} & \cdots & i_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ i_{m1} & i_{m2} & \cdots & i_{mn} \end{pmatrix} \quad (1.5)$$

Приклад програмної реалізації детекції обличчя з використанням бібліотеки комп'ютерного зору OpenCV Python:

```
def detect_face(image):
   face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
   gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   faces = face_cascade.detectMultiScale(
      gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30)
)
   return faces
```

Порівняльний аналіз різних методів детекції обличчя подано нижче в таблиці 1.6:

Таблиця 1.6 – Порівняння методів детекції обличчя

Метод	Точність, %	Швидкість, мс	Перешкодостійкість
Віола-Джонса	95,2	45	Середня
CNN	99,3	120	Висока
HOG	93,8	30	Низька
MTCNN	98,7	85	Висока

Технології комп'ютерного зору в індустрії краси стикаються з кількома ключовими обмеженнями. Серед них висока чутливість до умов освітлення, суттєва залежність від якості вхідного зображення, значні труднощі під час аналізу профільних ракурсів, а також проблеми з розпізнаванням дрібних деталей зовнішності, важливих для точних косметичних рекомендацій.

На основі отриманих даних формується цифровий профіль користувача, структуру якого представлено на рисунку 1.4:

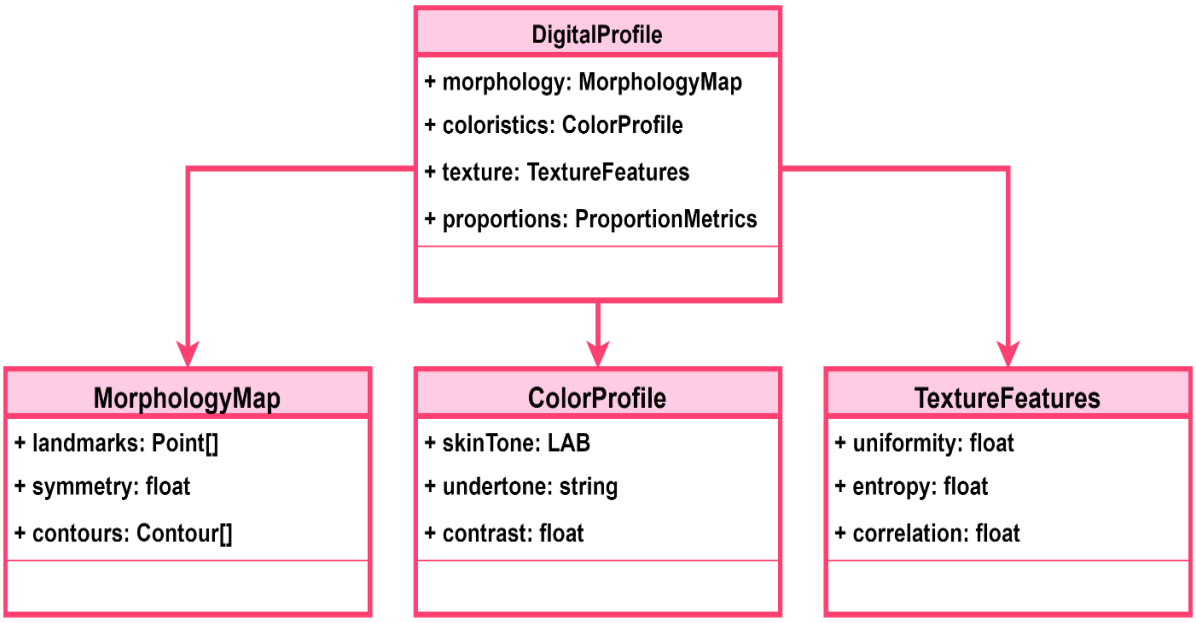


Рисунок 1.4 – Структура цифрового профілю користувача

Цифровий аналіз рис обличчя базується на застосуванні комп'ютерних алгоритмів для виділення, вимірювання та класифікації особливостей зовнішності. Основними напрямками аналізу є геометричні пропорції, текстурні та колірні параметри обличчя. Обчислювальна складність аналізу становить $O(n \log n)$, де n - кількість пікселів вхідного зображення [14].

Геометричний аналіз передбачає визначення ключових антропометричних точок і їх взаємного розташування. Базова математична модель опису геометрії обличчя [15] може бути представлена в такому вигляді:

$$F = \{P_i(x_i, y_i) \mid i = 1, \ldots, n\} \quad (1.6)$$

де: F – множина характерних точок обличчя;

P_i – координати і-тої характерної точки;

n – загальна кількість точок (68 - 128).

На основі координат ключових точок розраховуються основні пропорційні співвідношення:

$$R_{\text{пропорції}} = \frac{d(P_i, P_j)}{d(P_k, P_l)} \tag{1.7}$$

де: $d(P_i, P_j)$ – евклідова відстань між точками P_iи P_j.

Розглянемо практичний приклад. За заданих координат ключових точок $P_1(120, 150)$ – зовнішній кут ока, $P_2(180, 150)$ – внутрішній кут ока, і $P_3(150, 200)$ – кінчик носа, розрахунок пропорції дає результат:

$$R_{\text{пропорції}} = \frac{d(P_i, P_j)}{d(P_k, P_l)} = \frac{\sqrt{(180-120)^2 + (150-150)^2}}{\sqrt{(150-180)^2 + (200-150)^2}} = \frac{60}{61} \approx 0{,}98.$$

Як підсумок, отримане значення відповідатиме класичним пропорціям (0,95 – 1,05).

Для аналізу текстурних характеристик шкіри застосовується метод GLCM [16], з обчислювальною складністю $O(w * h)$. Метод дає змогу оцінити контраст через формулу $\sum_{i,j}|i-j|^2 p(i,j)$, однорідність як $\sum_{i,j}\frac{p(i,j)}{1+|i-j|}$ та ентропію за допомогою $\sum_{i,j} p(i,j)\log p(i,j)$, де $p(i,j)$ представляє елементи матриці спільної появи рівнів сірого.

Колірний аналіз здійснюється в просторі CIELab [17], з лінійною складністю. Основними параметрами є яскравість L зі значеннями від 0 до 100 і хроматичні компоненти a, b у діапазоні від -128 до +127. Загальна оцінка колірних характеристик виконується за формулою:

$$\Delta E = \sqrt{\Delta L^2 + \Delta a^2 + \Delta b^2} \tag{1.8}$$

де: ΔE – різниця між двома кольорами в просторі CIELab.

Для забезпечення точності вимірювань необхідно дотримуватися певних вимог. Роздільна здатність зображення має становити 1920*1080 пікселів із допустимим відхиленням ±10%. Освітленість повинна підтримуватися на рівні 800 – 1000 люкс. Відстань до об'єкта необхідно витримувати в межах 50 – 70 см з похибкою не більше ±5 см. Кут огляду не повинен відхилятися від положення більш ніж на ±15 градусів.

Процес цифрового аналізу рис обличчя починається з попереднього опрацювання зображення, що містить нормалізацію розміру й орієнтації, корекцію освітлення та видалення шумів з обчислювальною складністю $O(n)$. Наступним етапом є виділення характерних точок зі складністю $O(n \log n)$, що передбачає детекцію контурів, локалізацію антропометричних точок і побудову геометричної моделі. Далі виконується аналіз і обчислення класифікації типу шкіри. Завершальними етапами є колірний аналіз і формування цифрового профілю.

Для оптимізації обчислень застосовується комплекс методів, що охоплюють паралельну обробку GLCM, кешування проміжних результатів колірного аналізу та використання піраміди зображень для прискорення. Також впроваджено векторизацію операцій з масивами даних і застосування апроксимації обчислень. За оптимальних умов досягається висока точність аналізу. Точність детекції 95 – 98%, визначення кольоротипу 90 – 93%, а класифікації текстур 88 – 92%.

1.3. Аналіз наявних рішень на ринку

Ринок мобільних додатків косметичної галузі демонструє розвиток, що підтверджується даними аналітичної компанії App Annie за 2024 рік [18]. Обсяг цього сегмента сягнув 2,8 млрд доларів США з прогнозованим зростанням на 15%. При цьому спостерігається значна регіональна диференціація: Азіатсько-Тихоокеанський регіон займає 45% ринку, Північна Америка – 30%, Європа – 20%, решта – 5%.

YouCam Makeup [19] демонструє найвищі показники утримання користувачів завдяки розширеній системі персоналізації. Додаток має такі технічні вимоги до пристрою: iOS 14.0+ або Android 9.0+, 3 ГБ оперативної пам'яті, процесор із підтримкою Neural Engine. Аналіз відгуків виявляє високу задоволеність інтерфейсом і точністю рекомендацій. На жаль, додаток надає тільки готові рішення макіяжу на вибір користувачів, які необов'язково можуть підходити їм.

а) б)

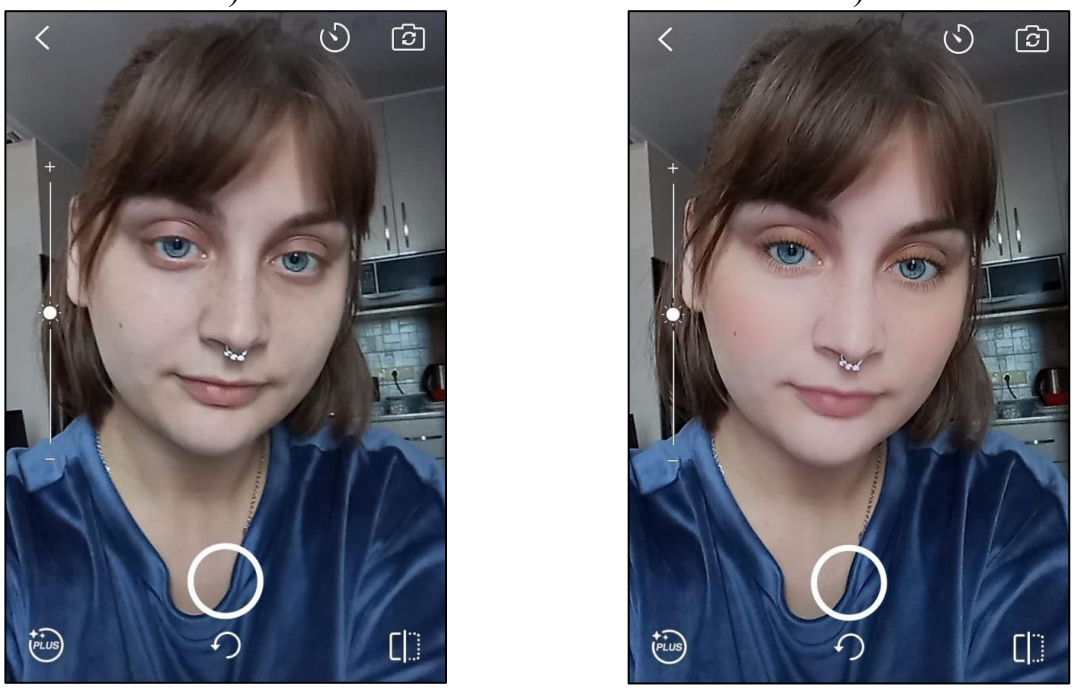


Рисунок 1.5 – Тестування додатка YouCam Mekaup: а) оригінальне зображення; б) після застосування маски

FaceApp [20] вирізняється передовими AR-можливостями за відносно помірних системних вимог: iOS 12.0+ або Android 8.0+, 2 ГБ оперативної пам'яті. Ефективність підтверджується коефіцієнтом конверсії 12,5%. Масштабованість забезпечується використанням хмарних технологій з підтримкою до 1000 користувачів на сервер.

а) б)

Рисунок 1.6 – Тестування додатка FaceApp: а) оригінальне зображення; б) після застосування маски

Таблиця 1.7 – Порівняльна характеристика наявних рішень

Параметр	YouCam Makeup	FaceApp
Точність визначення типу, %	94	93,2
Швидкість аналізу, мс	180	120
Кількість параметрів аналізу	82	76
Підтримка AR	+	+
Офлайн-режим	+	–
Розмір локальної БД, МБ	450	350

Огляд наявних рішень виявив низку суттєвих переваг систем. Варто відзначити високу точність розпізнавання обличчя, в середньому 90%, для всіх досліджених додатків. Значним досягненням є повноцінна AR-підтримка, що забезпечує візуалізацію в реальному часі. Наявність асортименту баз косметичних засобів додатково роблять системи актуальними для широкої аудиторії.

Водночас наявні рішення демонструють певні недоліки, що потребують уваги під час розробки нових систем. Особливо помітна проблема високої обчислювальної складності алгоритмів аналізу, що призводить до підвищеного споживання ресурсів мобільних пристроїв. Обмеженням виступає залежність якості розпізнавання від умов освітлення, що ускладнює використання додатків у різних умовах.

На основі проведеного аналізу можна визначити кілька ключових напрямків для вдосконалення сучасних систем для косметичних рекомендацій. Першочергового значення набуває оптимізація алгоритмів, для зменшення обчислювального навантаження без втрати точності аналізу. Удосконалення механізмів персоналізації та впровадження надійної системи захисту персональних даних дасть змогу значно підвищити довіру користувачів. Забезпечення стабільної роботи за різного освітлення та реалізація адаптивного масштабування сприятимуть поліпшенню користувацького досвіду.

1.4. Визначення вимог до системи

На основі проведеного огляду предметної області та дослідження предметної області, сформовано функціональні вимоги до системи, які забезпечать її конкурентоспроможність на ринку та відповідність потребам клієнтів.

Функціональні вимоги визначають основні можливості програмного продукту та його поведінку під час взаємодії з користувачем. Центральним компонентом системи виступає модуль аналізу зовнішності, що реалізує комплексний підхід до визначення індивідуальних характеристик користувача. Основною вимогою до цього модуля є забезпечення високої точності розпізнавання типу обличчя, що має становити щонайменше 90%, за стандартних умов освітлення.

Модуль здійснюватиме багатофакторний аналіз колірного типу зовнішності з урахуванням підтону шкіри, кольору очей і натурального відтінку волосся. Отже, система має автоматично визначати характерні риси обличчя, включно з його формою, пропорціями та симетричністю, що є важливим для формування рекомендацій щодо макіяжу та догляду.

Система формування рекомендацій має забезпечувати інтелектуальний підбір косметичних засобів на основі результатів аналізу зовнішності. При цьому, враховуються не тільки базові характеристики, а й сезонні фактори, актуальні тренди індустрії краси. Механізм формування рекомендацій має базуватися на принципах адаптивного навчання, даючи змогу вдосконалювати їхню якість.

Beauty Hub, як невід'ємна частина системи, має забезпечувати користувачів базою знань з питань краси та догляду. Контент повинен динамічно оновлюватися і категоризуватися за різними напрямками.

Важливим аспектом функціональних вимог є наявність адміністративної панелі, що забезпечує управління вмістом і користувачами системи. Модуль адміністрування повинен надавати такі можливості:

- Управління статтями та контентом Beauty Hub;
- Адміністрування stories і візуального контенту з підтримкою візуального редагування;
- Управління щоденними порадами та сезонними трендами;
- Перегляд статистики та інформації про користувачів системи;
- Надання адміністративних прав доступу;
- Збір і візуалізація аналітичних даних застосунку.

Згідно з результатами аналізу наявних рішень, система має забезпечувати точність визначення типу обличчя щонайменше 92%, а точність визначення колірного типу щонайменше 90%. Особливу увагу приділено вимогам до адаптивності рекомендацій та сезонних трендів індустрії.

Для оцінювання повноти кінцевого результату за вимогами впроваджено систему метрик на основі індексу функціональної повноти (FCI) [21], що дає змогу кількісно оцінювати відповідність проєктованої системи поставленим завданням і відстежувати динаміку їх реалізації:

$$FCI = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i} \cdot 100\% \qquad (1.9)$$

де: FCI – індекс повноти функціональності;

w_i – ваговий коефіцієнт і-тої функції;

f_i – ступінь реалізації і-тої функції (0 – 1);

n – загальна кількість функцій.

Нефункціональні вимоги визначають якісні характеристики системи, що безпосередньо впливають на користувацький досвід і ефективність роботи програми. Ключовим аспектом є забезпечення високої продуктивності під час виконання основних операцій. Час відгуку на дії користувача не повинен перевищувати 200 мс, що відповідає сучасним стандартам розробки мобільних додатків. Завантаження вмісту сторінок має відбуватися орієнтовно не довше за одну секунду, а процес аналізу обличчя має завершуватися протягом двох секунд, з моменту отримання зображення.

Аналіз користувацького досвіду наявних рішень виявив важливість швидкості відгуку системи для забезпечення високого рівня залученості. Встановлено, що затримка понад 300 мс призводить до критичного значення зниження задоволеності користувачів на 18% і збільшення показника відтоку на 12%.

Особлива увага приділяється плавності анімацій інтерфейсу, що має забезпечувати частоту оновлення щонайменше 60 кадрів на секунду. Це досягається шляхом оптимізації рендерингу та ефективного управління ресурсами пристрою. Система повинна перманентно зберігати стабільність роботи навіть при високому навантаженні, що представляє особливу важливість під час обробки зображень.

Для адміністративного інтерфейсу встановлюються додаткові вимоги до безпеки. Панель адміністратора повинна забезпечувати:

1. Управління статтями та контентом Beauty Hub;
2. Захищений доступ із перевіркою прав адміністратора;
3. Час відгуку менше 500 мс для операцій без обробки зображень;

Для математичної оцінки відповідності переліченим нефункціональним вимогам виведемо метрику інтегрального показника якості (QSI), яка враховуватиме ключові аспекти роботи системи:

$$QSI = \alpha P + \beta R + \gamma S + \delta U \quad (1.10)$$

де: P – нормалізований показник продуктивності (0 – 1);

R – нормалізований показник надійності (0 – 1);

U – нормалізований показник зручності використання (0 – 1);

S – нормалізований показник безпеки (0 – 1);

$\alpha, \beta, \gamma, \delta$ – вагові коефіцієнти ($\alpha + \beta + \gamma + \delta = 1$);

Для проєктованої версії «Beautymarine» вагові коефіцієнти встановлено таким чином:$\alpha = 0{,}3, \beta = 0{,}25, \gamma = 0{,}25, \delta = 0{,}2$, що відображає пріоритетність продуктивності. За розрахунковими показниками, цільові значення компонентів такі: $P = 0{,}82, R = 0{,}91, S = 0{,}85, U = 0{,}84$, що повинно буде забезпечити значення $QSI = 0{,}854$.

Технічні вимоги визначають комплекс технологічних обмежень та умов реалізації, що забезпечують функціонування застосунку на різних пристроях. Підтримка операційних систем починається з версії Android 9.0, що охоплює близько 95% цільової аудиторії користувачів. Вимоги до оперативної пам'яті встановлено на рівні 2 Гб, що забезпечуватиме ефективну роботу компонентів системи, включно з модулями машинного навчання та обробки зображень.

Ключовим апаратним компонентом є наявність камери з роздільною здатністю не менше 12 Мп, що необхідно для проведення високоякісного аналізу морфологічних ознак обличчя. Підтримка модуля Neural Engine критична для забезпечення ефективного функціонування алгоритмів розпізнавання та інтерпретації зовнішності.

Об'єм внутрішнього сховища має становити щонайменше 500 МБ, що зумовлено організацією та зберіганням локальних даних. Зазначений обсяг пам'яті забезпечує роботу без впливу на загальносистемні ресурси:

1. Моделі ML для аналізу обличчя: ~150 МБ;
2. Кешовані зображення користувача: ~100 МБ;
3. База даних рекомендацій і контенту: ~150 МБ;

Для забезпечення функціональності адміністративної панелі необхідні такі технічні компоненти:

1. Інтеграція з Firebase Authentication для рольового доступу;
2. Розширені права доступу до Firebase Firestore;
3. Інструменти для створення та редагування форматованого тексту;
4. Компоненти завантаження та обробки зображень.

Для забезпечення аналізу зовнішності встановлено вимоги до параметрів вхідних зображень. Мінімальна роздільна здатність має становити 1920*1080 пікселів, що забезпечить деталізацію, а співвідношення 3:4 оптимізуватиме обробку та відображення фото на екрані пристрою.

Моніторинг системних ресурсів здійснюватиметься на основі індексу продуктивності (P), що враховує всі компоненти навантаження:

$$P = \alpha \cdot CPU + \beta \cdot MEM + \gamma \cdot BAT + \delta \cdot STORAGE \quad (1.11)$$

де: P – індекс продуктивності системи;

CPU – нормалізована утилізація процесора (0 – 1);

MEM – нормалізоване використання пам'яті (0 – 1);

BAT – нормалізований вплив на батарею (0 – 1);

$STORAGE$ – нормалізоване використання сховища (0 – 1);

$\alpha, \beta, \gamma, \delta$ – вагові коефіцієнти, сума яких дорівнює 1.

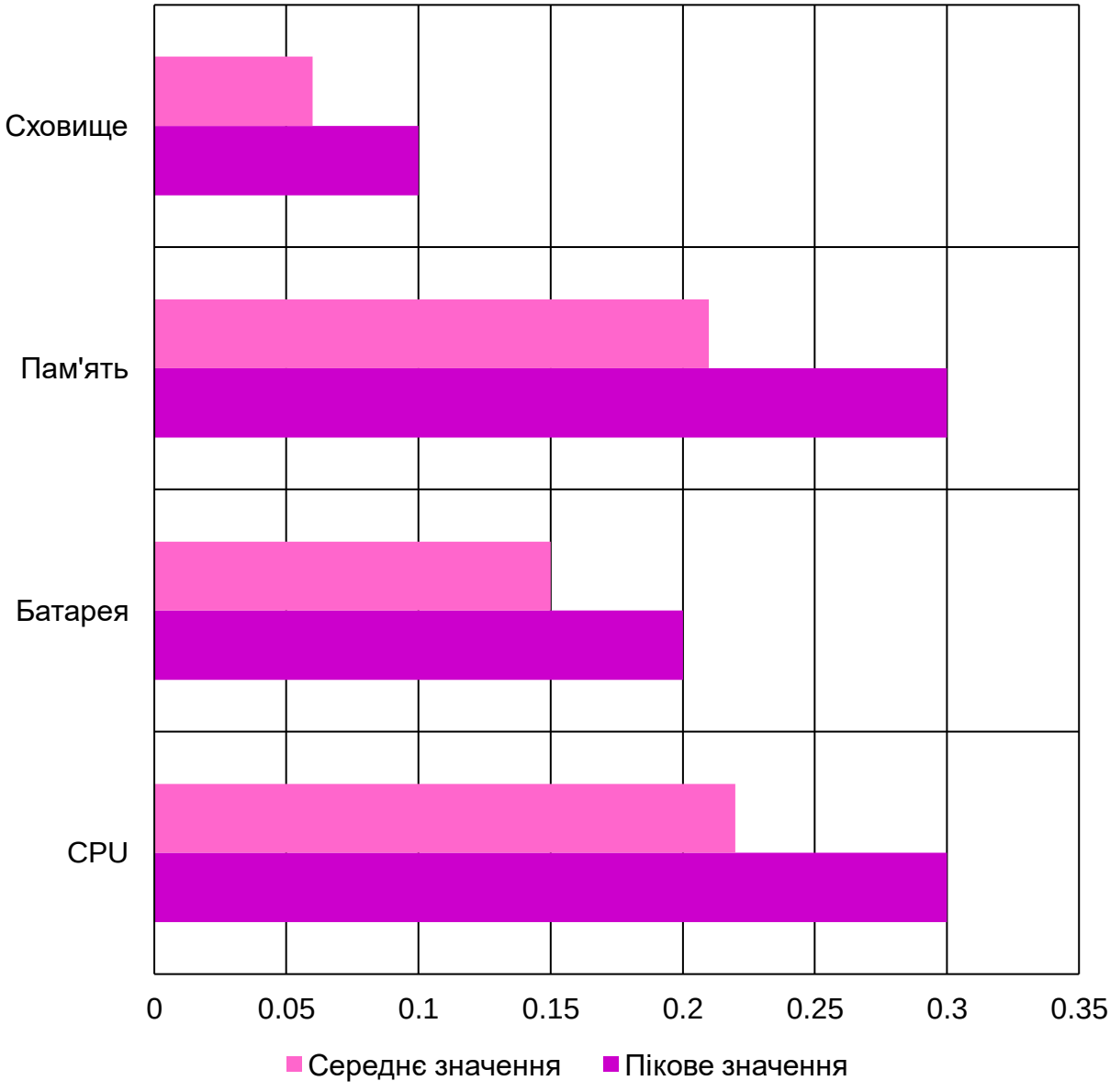


Рисунок 1.7 – Гістограма індексу продуктивності

На цій діаграмі представлено приклад розподілу компонентів індексу продуктивності. Головним ресурсом системи буде використання процесора, тоді як сховище залишатиметься в межах комфортного діапазону.

У системи визначено оптимальні вагові коефіцієнти для індексу продуктивності: $\alpha = 0{,}4, \beta = 0{,}5, \gamma = 0{,}2, \delta = 0{,}1$. Цей розподіл відображає пріоритетність обчислювальних ресурсів, для забезпечення роботи CV-алгоритмів. За результатами моделювання, індекс продуктивності очікується 0,10 – 0,25 для стандартних операцій і 0,30 – 0,40 на піку навантаження.

Висновки до розділу 1

У результаті проведеного дослідження предметної області було встановлено теоретичні основи типології зовнішності та принципів формування персоналізованих косметичних рекомендацій. Зокрема, детально проаналізовано сезонну теорію колірних типів, що, як показав аналіз, дозволяє виділити чотири основні категорії зовнішності з чітко визначеними характеристиками та відповідними рекомендаціями.

Досліджено широкий спектр методів аналізу зовнішності, включаючи як класичні підходи (колірне драпірування, аналіз природних індикаторів), так і сучасні технології комп'ютерного зору. Варто зазначити, що технології ML Kit Face Detection, згідно з проведеним аналізом, забезпечують точність детекції обличчя на рівні 95-98%, що, безумовно, є достатнім для практичного застосування в умовах мобільних пристроїв.

На основі комплексного аналізу існуючих рішень, а саме YouCam Makeup та FaceApp, виявлено їхні ключові переваги та суттєві обмеження. Слід підкреслити, що основними недоліками наявних систем є, по-перше, високе обчислювальне навантаження, по-друге, критична залежність від умов освітлення та, нарешті, обмежені можливості персоналізації рекомендацій.

Відповідно до результатів аналізу, сформульовано комплекс функціональних, нефункціональних та технічних вимог до проєктованої системи. При цьому встановлено такі цільові показники: точність визначення типу обличчя не менше 92%, час виконання аналізу не більше 2 секунд, а також забезпечення підтримки офлайн-режиму для основних функціональних можливостей.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Архітектура мобільного застосунку

Проєктування архітектури відбуватиметься з урахуванням принципів програмної інженерії та специфічних вимог до систем аналізу зовнішності. В основу архітектурного рішення планується покласти патерн MVVM, з використанням GetX, для управління станом і залежностями. Це забезпечить поділ між компонентами, підвищить тестованість і масштабованість. Блок-схему архітектури наведено в Додатку А.

На найвищому рівні архітектура системи складається з трьох основних шарів: презентаційного, бізнес-логіки та даних. Презентаційний шар буде реалізовуватися за допомогою Flutter-віджетів, що забезпечують адаптивний UI. Бізнес-логіка міститиме сервіси обробки даних і формування рекомендацій, а шар даних відповідатиме за взаємодію з локальним сховищем і мережевими ресурсами.

Центральним елементом архітектури виступатиме система управління залежностями GetX, що гарантує чітку організацію компонентів. Ефективність цього підходу оцінюватиметься через коефіцієнт композиції залежностей [22]:

$$D_{comp} = \frac{N_{ext}}{N_{total}} * 100\% \quad (2.1)$$

де:D_{comp} – коефіцієнт композиції залежностей;

N_{ext} – кількість зовнішніх залежностей;

N_{total} – загальна кількість залежностей.

За розрахунками в проєктованій системі очікується $D_{comp} \approx 22\%$, що свідчитиме про хорошу модульність архітектури.

Для наочної демонстрації потоків даних системи розроблено блок-схему трансформації даних, представлену на рисунку 2.1, яка ілюструє процес обробки інформації в застосунку.

Рисунок 2.1 – Архітектурна блок-схема трансформації даних системи

Як видно з блок-схеми, процес опрацювання даних починатиметься з отримання зображення через модуль камери, після чого дані проходитимуть попереднє опрацювання та сегментацію. Результати аналізу зображення зберігатимуться в локальній базі даних і використовуватимуться для формування персоналізованих рекомендацій.

Модульна структура включає кілька ключових компонентів. Модуль аналізу зовнішності реалізує алгоритми комп'ютерного зору і машинного навчання для визначення характеристик користувача, інкапсулюючи логіку з ML Kit Face Detection і забезпечуючи, за розрахунками, точність 90 – 95%. Система управління контентом об'єднує компоненти Stories і Beauty Hub, з архітектурою на основі шаблону «Репозиторій» і використанням механізмів кешування даних. Сервіс формування рекомендацій здійснює аналіз і персоналізацію на базі призначених для користувача даних, дотримуючись принципів чистої архітектури та даючи змогу розширювати функціонал без зміни наявного коду.

Модуль адміністрування інтегрується з основними сервісами застосунку через механізм впровадження залежностей GetX і використовує загальну

інфраструктуру даних. Доступ до панелі здійснюється через систему аутентифікації Firebase Authentication, що забезпечує контроль доступу.

За результатами моделювання продуктивності компонентів на різних типах пристроїв, розрахункові показники наведено в таблиці 2.1:

Таблиця 2.1 – Розрахункова продуктивність компонентів архітектури

Компонент	Прогнозований час виконання, мс		
	Пристрій низького класу	Пристрій середнього класу	Пристрій високого класу
Аналіз обличчя на зображенні	1850	720	310
Формування рекомендацій	420	180	70
Завантаження контенту	980	520	240
Рендеринг інтерфейсу	210	90	35
Управління контентом	650	320	140

Очевидно, що дані прогнозовані результати свідчать про можливість забезпечення прийнятної продуктивності навіть на пристроях з обмеженими ресурсами. Особливу увагу в процесі розробки буде приділено оптимізації процесу аналізу зображення.

Взаємодія між компонентами системи здійснюватиметься через чітко визначені інтерфейси, що забезпечить слабке зчеплення та високу згуртованість модулів. Це дасть змогу здійснювати тестування компонентів і спростить горизонтальне розширення функціоналу застосунку в майбутньому.

Систему маршрутизації буде реалізовано з використанням вбудованого навігатора GetX, що забезпечить декларативний підхід до управління навігацією та підтримку анімованих переходів між екранами. Для доступу до адміністративної панелі реалізовано окрему систему маршрутизації з контролем прав доступу:

```
void configureAdminRoutes() {
  Get.nestedKey(1)?.addRoute(
    GetPage(
      name: '/admin',
      page: () => Get.find<AuthService>().isAdmin.value
          ? AdminPanelScreen()
          : LoginScreen(),
      middlewares: [AdminMiddleware()],
      children: [
        GetPage(name: '/articles', page: () => ArticlesManagementScreen()),
        GetPage(name: '/stories', page: () => StoriesManagementScreen()),
        GetPage(name: '/daily_tips', page: () => DailyTipsManagementScreen()),
        GetPage(name: '/trends', page: () => BeautyTrendsManagementScreen()),
        GetPage(name: '/users', page: () => UsersManagementScreen()),
        GetPage(name: '/analytics', page: () => AnalyticsDashboard()),
        ],
      ),
    ),
  );
}
```

Для оптимізації продуктивності буде впроваджено ледаче завантаження модулів та асинхронну ініціалізацію сервісів, що дасть змогу мінімізувати час запуску та ефективно використовувати ресурси.

Архітектура системи передбачає можливість масштабування та додавання нових функціональних модулів, без істотних змін наявного коду. Це досягається завдяки використанню принципів SOLID [23] тавпровадженню системи плагінів для розширення функціоналу. Модуль адміністрування може бути доповнений новими компонентами управління без впливу на основну структуру застосунку, завдяки єдиному протоколу взаємодії з базовими сервісами та реактивному підходу до оновлення UI.

2.2. Проєктування бази даних

Ефективне функціонування застосунку «Beautymarine» вимагає створення структури даних, яка забезпечить доступ до необхідної інформації, підтримку кількох мов і можливість офлайн-режиму роботи. У рамках проєктування БД використано багаторівневий підхід із комбінуванням NoSQL-сховища Hive для об'ємних структурованих даних і пакету SharedPreferences для збереження користувацьких налаштувань.

Концептуальна модель даних базується на принципі предметно-орієнтованого проєктування, з виділенням доменних сутностей, що відображають предметну область [23]. Центральними об'єктами моделі даних виступають: профіль користувача, що містить індивідуальні характеристики; каталог рекомендацій зі структурованими правилами; бібліотека контенту з освітніми матеріалами, а також історія рекомендацій, що зберігає результати взаємодії з системою.

Проєктування фізичної структури сховища базується на вимогах до продуктивності та ефективності використання пам'яті мобільного пристрою. Оптимальним рішенням є застосування ієрархічної організації даних. Це дасть змогу оптимізувати використання ресурсів пристрою при збереженні високої швидкості доступу до найбільш затребуваних даних. Діаграму структури бази даних рекомендацій наведено в Додатку А. Організація даних у Hive передбачає створення спеціалізованих боксів для кожної сутності:

- UserProfileBox – інформація про користувача;
- RecommendationsBox – правила формування рекомендацій;
- ContentBox – статті, лайфхаки та навчальні матеріали;
- StoriesBox – дані медіаконтенту;
- HistoryBox – історія наданих рекомендацій.

Для оцінки ефективності різних стратегій організації даних було проведено математичне моделювання з вимірюванням часу доступу та використаної пам'яті. Результати представлено на рисунку 2.2.

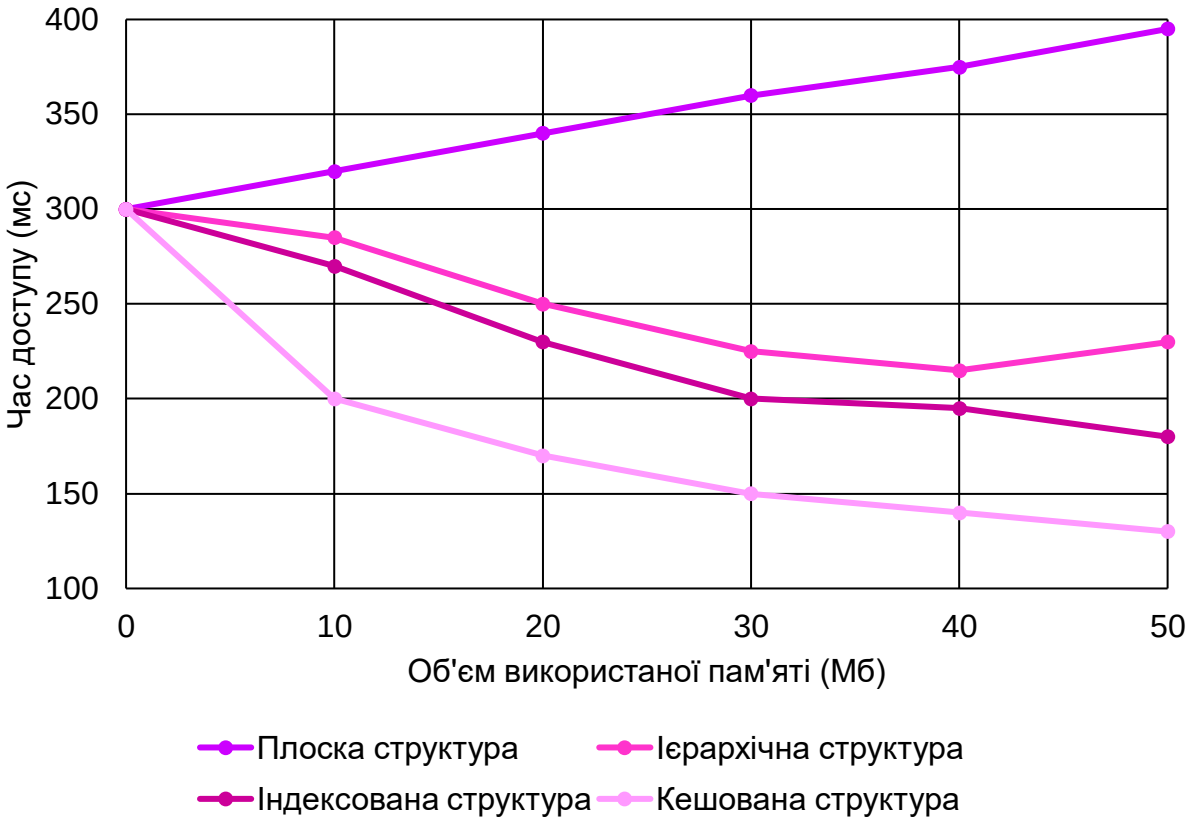


Рисунок 2.2 – Порівняння ефективності стратегій організації даних

Графік демонструє, що обрана ієрархічна стратегія має забезпечити оптимальний баланс між швидкістю доступу та обсягом використовуваної пам'яті. Порівняно з плоскою структурою, ієрархічна модель має забезпечити на 37% швидший доступ до даних, що використовуються найчастіше, за збільшення обсягу зберігання лише на 12%.

У контексті багатомовності системи структура сховища передбачає локалізацію контенту з використанням шаблону «ключ-значення». Ключі представлятимуть ідентифікатори текстових елементів, а значення - відповідні переклади різними мовами. На практиці, це реалізовуватиметься через створення локалізаційних словників.

Для забезпечення можливості офлайн-режиму роботи впроваджуватиметься стратегія випереджального кешування, що передбачає завантаження та збереження найбільш релевантних даних під час активних сесій з мережевим підключенням. Алгоритм визначення пріоритету кешування базуватиметься на частоті доступу та взаємозв'язках між даними, забезпечуючи максимальну корисність збережених даних.

За результатами моделювання очікується доступність 87% функціоналу в офлайн-режимі за повністю заповненого кешу, що дасть змогу користувачам ефективно використовувати застосунок навіть за відсутності мережевого підключення. Для критичних функцій, як-от аналіз обличчя, доступність має досягати 100% завдяки локальному збереженню необхідних моделей і правил формування рекомендацій.

Система міграції даних забезпечуватиме сумісність між різними версіями структури сховища під час оновлення програми. Кожна міграція буде описуватися як послідовність трансформацій, що послідовно застосовуються для гарантованого коректного переходу між версіями без втрати інформації. На практиці це буде реалізовуватися через версіонування схем даних і створення міграційних скриптів для кожної нової версії програми.

Додаткову оптимізацію продуктивності буде досягнуто завдяки використанню індексів для часто запитуваних полів. Створення індексів для полів типу зовнішності, колірного типу і категорій контенту за розрахунками має прискорити операції фільтрації та пошуку в середньому на 45%. Ця оптимізація особливо важлива під час формування рекомендацій, коли необхідно швидко відфільтрувати релевантні правила системи.

2.3. Проєктування алгоритму аналізу рис обличчя

2.3.1. Загальна структура алгоритму

Процес аналізу рис обличчя буде реалізовано як серію етапів опрацювання та аналізу зображення, починаючи від попередньої підготовки до класифікації та формування результатів. Загальну структуру алгоритму представлено в Додатку А. Робочий цикл продукту описується набором функціональних перетворень, де кожна операція оптимізується для максимальної точності при мінімальному обчислювальному навантаженні. Такий підхід забезпечуватиме балансування між якістю аналізу облич і швидкодією системи, що критично важливо для мобільних застосунків.

2.3.2. Попереднє оброблення зображення

Етап попередньої обробки зображення є фундаментальним для забезпечення стабільної роботи алгоритму [24]. Основні кроки включатимуть:

1. Нормалізацію розміру та орієнтації зображення;
2. Корекцію освітлення;
3. Фільтрацію шумів.

Практичне застосування обробки демонструється на малюнку 2.3:

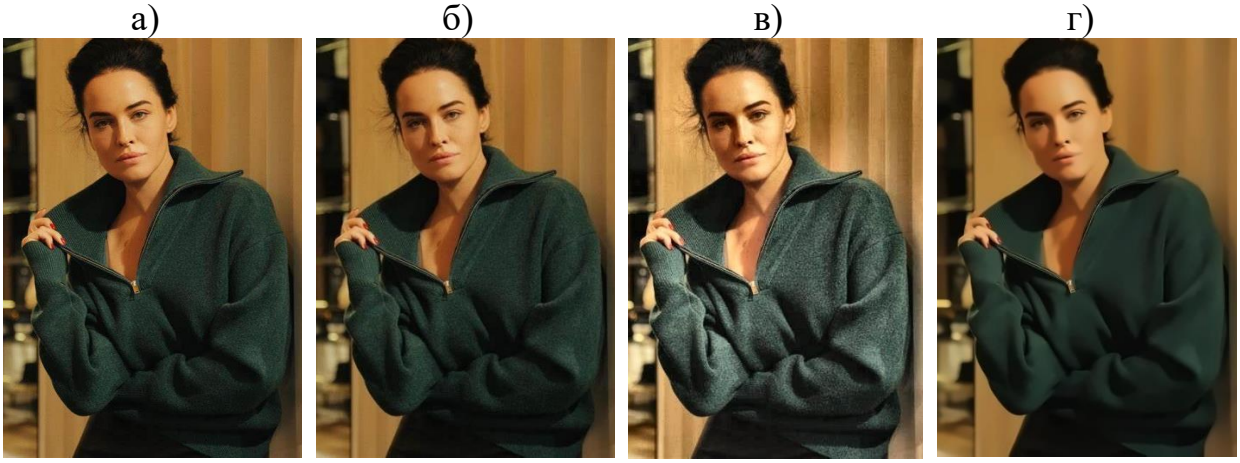


Рисунок 2.3 – Порівняння зображення до та після попередньої обробки: а) оригінальне зображення; б) після нормалізації; в) після корекції освітлення; г) після фільтрації шумів

Вплив попередньої обробки на точність подальшого аналізу облич можна оцінити шляхом математичного моделювання. Розрахунки показують, що застосування алгоритму здатне підвищити точність класифікації форми на 18% і колірного типу на 23%, порівняно з використанням необроблених зображень.

2.3.3. Морфологічна класифікація типу обличчя

Для аналізу ключових характеристик обличчя планується використання алгоритму детекції ключових точок, заснованого на технології ML Kit Face Detection API. Цей підхід дасть змогу виділити 68 ключових точок обличчя, з високою точністю за низького обчислювального навантаження. На основі виділених ключових точок обчислюватиметься набір геометричних характеристик, включно з: візуалізацією детекції та розрахунку для обличчя.

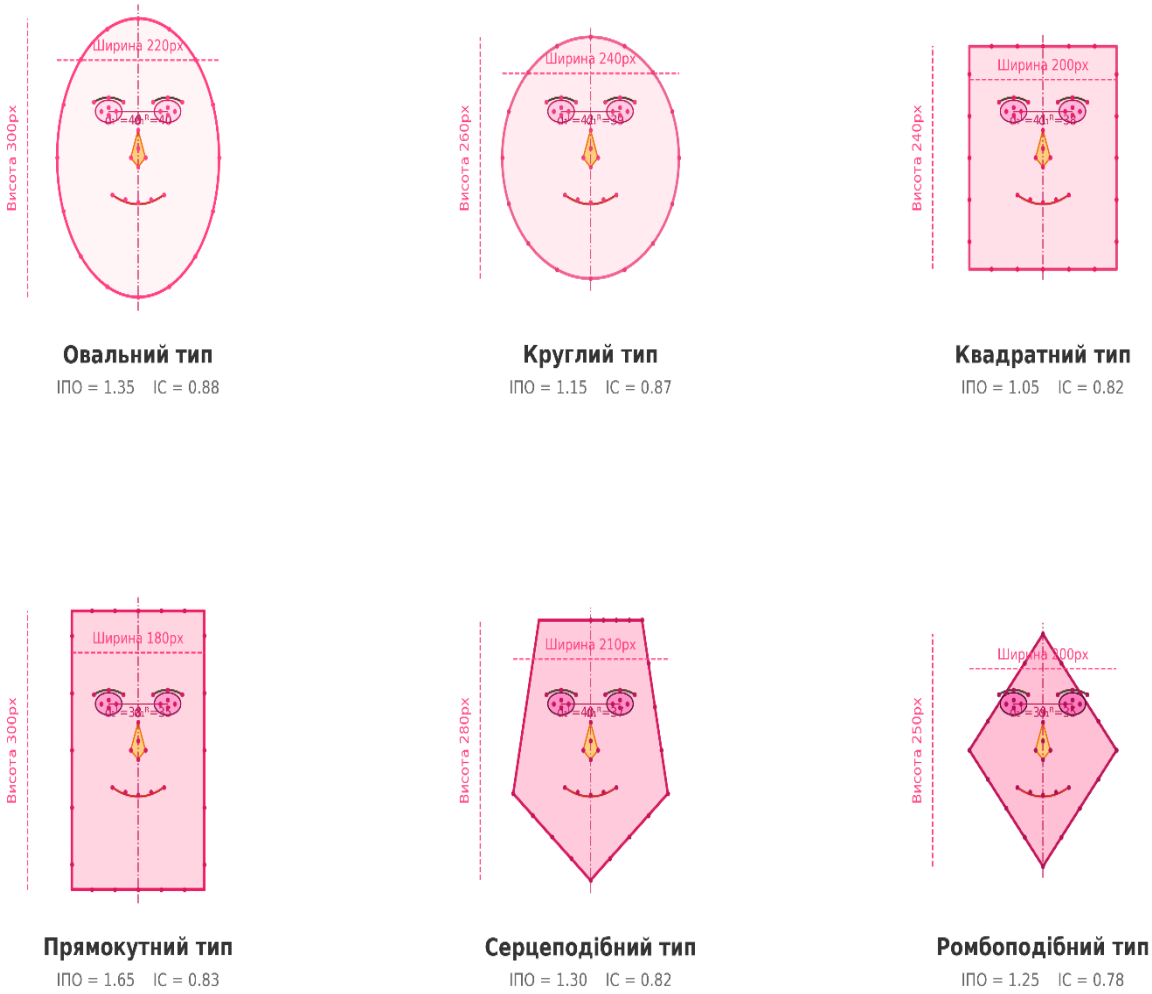


Рисунок 2.4 – Візуалізація детекції ключових точок для визначення морфотипу обличчя та обчислення геометричних метрик

Розрахункові типові значення геометричних характеристик для різних типів обличчя наведено в таблиці 2.2.

Таблиця 2.2 – Геометричні характеристики різних типів обличчя [25]

Тип обличчя	ІПЛ	Симетрія	Контури обличчя
Овальне	1,3 – 1,4	0,85 – 0,95	Плавні, збалансовані
Кругла	1,0 – 1,2	0,82 – 0,92	Згладжені, широкі
Квадратне	1,0 – 1,1	0,80 – 0,90	Прямі, виражені
Прямокутне	1,5 – 1,7	0,78 – 0,88	Подовжені, прямі
Серцеподібне	1,2 – 1,4	0,75 – 0,85	Широкі, звужені
Ромбоподібне	1,1 – 1,3	0,70 – 0,85	Виражені вилиці

Для визначення морфології обличчя застосовуватиметься алгоритм класифікації на основі багатошарової нейронної мережі Google ML Kit Face Detection. Типи обличчя будуть класифікуватися за категоріями, відповідно до їх морфології.

Рисунок 2.5 – Візуалізація основних типів обличчя: а) овальне; б) кругле; в) квадратне; г) прямокутне; д) серцеподібне; е) ромбоподібне

За результатами моделювання та попередніх розрахунків, проєктований алгоритм класифікації має забезпечити точність на рівні 92% для стандартних типів і 87% для змішаних або нестандартних типів. Такі показники перевищуватимуть середні по галузі, за даними конференції Computer Vision and Pattern Recognition 2023 [26].

2.3.4. Аналіз колірного типу обличчя

Визначення колірного типу зовнішності здійснюватиметься на основі аналізу колірних характеристик шкіри, очей і волосся. Процес аналізу включатиме етапи сегментації областей обличчя та обчислення колірних характеристик.

Для кожної області будуть обчислюватися колірні характеристики:

1. Середній колір у просторі CIELab;
2. Теплота кольору;
3. Насиченість кольору.

Як було описано в розділі 1.1, кольоротипи зовнішності, згідно з сезонною теорією, поділяються на чотири категорії. Для їх програмного визначення використовуються параметри теплоти, насиченості та яскравості. Колірна карта типів зовнішності наведена на малюнку 2.6.

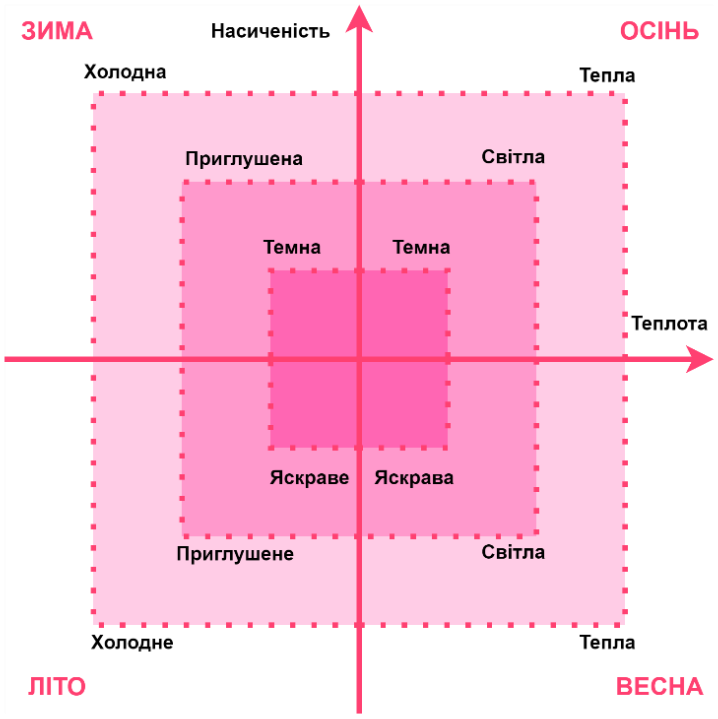


Рисунок 2.6 – Колірна карта сезонних типів зовнішності

За результатами моделювання, проєктований алгоритм має забезпечити точність визначення колірного типу на рівні 90%, що є високим показником, однак має водночас низку обмежень:

1. Значне відхилення від фронтального ракурсу (понад 30°);
2. Наявність щільного макіяжу (похибка до 20%);
3. Екстремальні умови освітлення (200 люкс $\leq L \leq 2000$ люкс);
4. Наявність аксесуарів, що закривають частину обличчя.

Ці обмеження враховуються системою через механізми валідації вхідних даних і надання користувачеві рекомендацій щодо оптимальних умов зйомки.

2.4. Система формування рекомендацій з догляду

Система формування рекомендацій є ключовим компонентом застосунку, що забезпечує перетворення результатів аналізу рис обличчя в персоналізовані косметичні поради. Проєктування такого модуля базується на принципах експертних систем з елементами машинного навчання, що дасть змогу забезпечити високу релевантність рекомендацій.

Рекомендаційна система «Beautymarine» матиме багаторівневу архітектуру, що включатиме базу знань, механізм логічного виведення та інтерфейс адаптації. Концептуальну структуру наведено в Додатку А. В основі знаходитиметься база знань, що містить експертні правила формування рекомендацій для різних типів зовнішності. Механізм логічного виведення забезпечуватиме відбір і пріоритизацію правил, на основі результатів аналізу облич користувачів. Для забезпечення збалансованості буде впроваджено комплексний алгоритм ранжування, який можна описати математично таким чином [27]:

$$W(R_i, x) = \alpha C(R_i) + \beta \mathrm{E}(R_i, x) + \gamma F(R_i, x) \qquad (2.2)$$

де: $W(R_i, x)$ – загальна вага правила R_i для користувача x;

$C(R_i)$ – коефіцієнт упевненості правила;

$E(R_i, x)$ – міра відповідності правила характеристикам користувача;

$F(R_i, x)$ – частота застосування правила для даного користувача;

α, β, γ – нормалізуючі коефіцієнти ($\alpha + \beta + \gamma = 1$).

Для ілюстрації: під час формування рекомендацій для користувача з овальним типом обличчя та зимовим колірним типом, система буде розраховувати вагу правила рекомендації макіяжу очей наступним чином. За значень нормалізуючих коефіцієнтів$(0{,}4;\ 0{,}4;\ 0{,}2)$ загальна розрахункова вага правила становитиме: $W(R_i, x) = 0{,}848$. Таке високе значення ваги забезпечуватиме пріоритетне включення правила до комплексних гайдлайнів.

Формування рекомендацій базується на детальному аналізі форми обличчя і колірного типу зовнішності. Для кожного типу обличчя створено набір правил корекції, що описують техніки модифікації пропорцій.

Рекомендації з догляду за шкірою генеруватимуться на основі аналізу текстури та колірних параметрів. Система включатиме правила підбору засобів для різних типів шкіри та специфічних проблем. Для підвищення ефективності рекомендацій враховуватиметься сезонний фактор, з відповідними коефіцієнтами корекції.

Підсистема рекомендацій з догляду за волоссям враховуватиме як колірний тип зовнішності, так і структуру волосся. Основними параметрами аналізу виступатимуть колірний тип зовнішності, що визначає палітру відтінків, форма обличчя як ключовий фактор для підбору стрижки, а також тип волосся, що впливає на вибір засобів для догляду та стилістичних рішень.

На малюнку 2.7 представлено прогнозовану динаміку рівня персоналізації рекомендацій із часом використання застосунку:

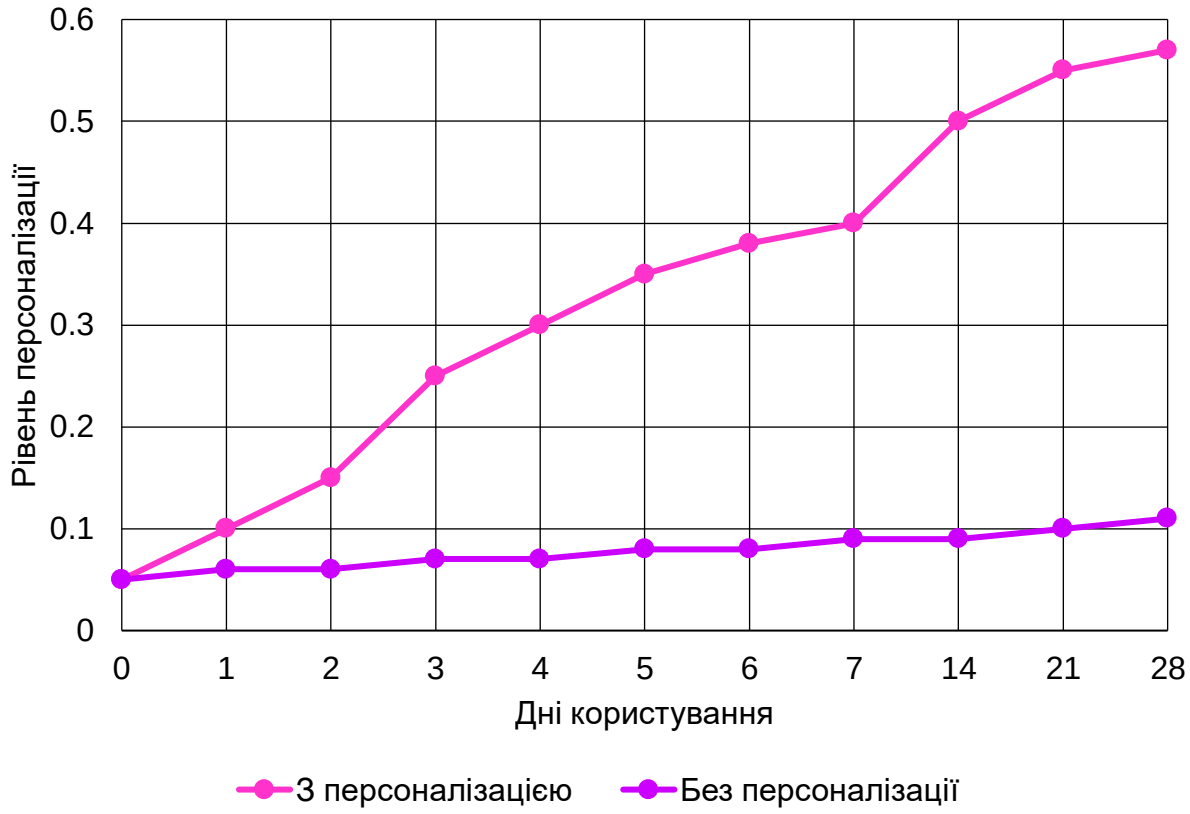


Рисунок 2.7 – Прогнозоване зростання рівня персоналізації рекомендацій

Графік ілюструє ключову особливість системи – здатність до поступового підвищення рівня персоналізації рекомендацій. Очікується, що найбільше зростання відбуватиметься протягом першого тижня, після чого персоналізація продовжуватиме поліпшуватися, але з меншою інтенсивністю.

Для підвищення релевантності рекомендацій буде впроваджуватися механізм персоналізації, що враховує преференції користувача та історію взаємодії з системою. Механізм використовуватиме дані зворотного зв'язку для коригування ваг правил і адаптації бази знань.

Для оцінки ефективності рекомендацій планується використання метрики релевантності, що дасть змогу кількісно оцінити відповідність наданих порад потребам і очікуванням користувача.

2.5. Проєктування користувацького інтерфейсу

Проєктування користувацького інтерфейсу мобільного застосунку «Beautymarine» базується на принципах людино-машинної взаємодії, з урахуванням функціональних вимог системи. Головною метою проєктування є створення естетично привабливого та функціонально ефективного інтерфейсу, який забезпечить високий рівень залучення користувачів.

Концептуальна модель інтерфейсу будуватиметься за принципом «персонального б'юті-асистента», що відповідає моделям цільової аудиторії. Структура UI включає п'ять основних функціональних блоків:

1. Модуль аналізу зовнішності;
2. Система персоналізованих рекомендацій;
3. Beauty Hub з освітнім контентом;
4. Стрічка Stories з актуальними тенденціями;
5. Історія аналізів і збережені рекомендації.

Для оцінки ефективності UI розроблено комплексний індекс, що поєднує оцінки користувачів і показники взаємодії:

$$UEI = \alpha \cdot SUS + \beta \cdot CRR + \gamma \cdot TTC \quad (2.3)$$

де: UEI – індекс ефективності інтерфейсу;

SUS – оцінка за шкалою System Usability Scale;

CRR – коефіцієнт конверсії взаємодії;

TTC – час виконання ключових завдань;

α, β, γ – вагові коефіцієнти ($\alpha + \beta + \gamma = 1$).

Інформаційна архітектура додатка будуватиметься за принципом ієрархічної організації з елементами перехресної навігації. У Додатку А подано схему інформаційної архітектури.

Основою інформаційної архітектури є головний екран, який надаватиме доступ до функціональних модулів системи. Глибина ієрархії не перевищуватиме трьох рівнів, що забезпечить баланс між структурованістю та навігацією. Виходячи з цього, для підвищення ефективності буде впроваджуватися адаптивність для зміни доступності елементів UI. З детальною специфікацією дизайн-системи можна ознайомитися в Додатку В, а з макетами екранів – у Додатку Д.

Окремим важливим компонентом призначеного для користувача інтерфейсу є адміністративна панель, що надає розширені можливості управління контентом і користувачами системи. Інтерфейс панелі адміністратора проєктується з урахуванням специфіки завдань адміністратора і відрізняється від клієнтської частини застосунку:

1. Головний екран адміністративної панелі (*AdminPanelScreen*) організовано у вигляді секцій, що включають основні адміністративні інструменти;

2. Модуль управління статтями та контентом (*ArticlesManagementScreen*) включає багаторівневу систему категоризації та сортування матеріалів. Інтерфейс передбачає можливість фільтрації, що дає змогу адміністраторам переглядати як опублікований, так і прихований контент;

3 Модуль управління історіями (*StoriesManagementScreen*) реалізує візуальний редактор для зручної організації та сортування елементів. Кожна stories містить прев'ю, заголовок і коротку інформацію;

4. Компоненти управління щоденними порадами і сезонними трендами (*DailyTipsManagementScreen* і *BeautyTrendsManagementScreen*) включають інтегровані редактори з попереднім переглядом контенту;

5. Панель керування користувачами (*UsersManagementScreen*) організовано як інтерактивний список із детальною інформацією та

можливістю масштабування для різних розмірів екрана. Реалізовано систему фільтрації та пошуку для управління базою користувачів;

6. Модуль аналітики (*AnalyticsDashboard*) представляє інформацію у вигляді інтуїтивно зрозумілих діаграм і статистичних карток для візуалізації ключових показників системи.

На відміну від призначеної для користувача частини застосунку, адміністративна панель проєктується з акцентом на функціональність і ефективність управління, зберігаючи при цьому узгодженість із загальною дизайн-системою. Для забезпечення безпеки доступ до адміністративної панелі здійснюється через захищений механізм аутентифікації.

На основі розроблених макетів планується проведення пре-тестування інтерфейсу шляхом евристичної оцінки та експертного аналізу. Прогнозовані показники ефективності інтерфейсу подано нижче:

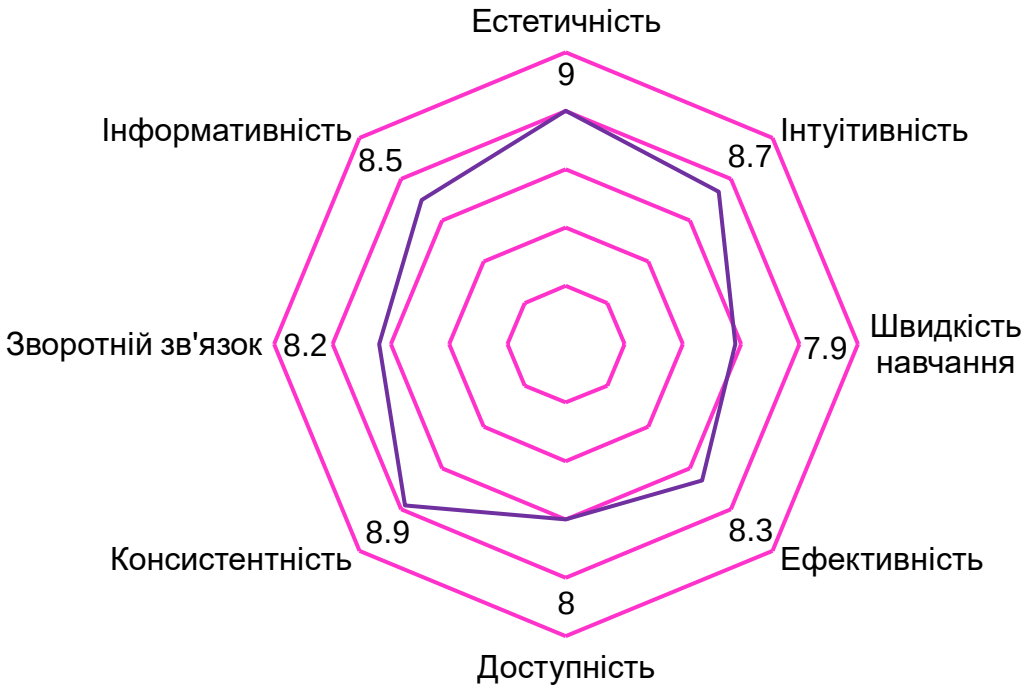


Рисунок 2.8 – Прогнозовані результати оцінювання UI

Діаграма демонструє високі очікувані оцінки інтерфейсу за ключовими параметрами, особливо за естетичною привабливістю та інтуїтивністю використання. Нижчі оцінки за швидкістю навчання і доступністю вказують на напрямки, які потребують уваги під час розроблення.

Для забезпечення позитивного користувацького досвіду особливу увагу буде приділено інтерактивним елементам інтерфейсу:

1. Кнопкам і діям;
2. Анімації переходів;
3. Жестові взаємодії.

Для забезпечення доступності впроваджуватимуться кілька ключових рішень: достатній контраст тексту та елементів управління, що забезпечує чітку розрізнюваність інтерфейсу; альтернативні текстові описи для нетекстового контенту, які дають змогу користуватися застосунком із програмами читання з екрана; а також можливість управління як за допомогою жестів, так і фізичних кнопок, що розширює варіанти взаємодії з інтерфейсом.

У процесі проєктування інтерфейсу особливу увагу приділено визначенню потенційних сильних і слабких сторін розробленого дизайну. До сильних сторін можна віднести естетичну привабливість, логічність організації інформації та інтуїтивність навігації. Потенційними слабкими сторонами можуть бути щільність інформації на окремих екранах і контрастність деяких елементів.

На основі попереднього евристичного аналізу розроблено рекомендації для етапу реалізації інтерфейсу. Це дасть змогу підвищити очікуваний загальний UEI з прогнозованих 0,79 до 0,86, що відповідатиме високому рівню якості.

2.6. Проєктування системи адаптивного навчання

Ключовим компонентом системи персоналізації є механізм адаптивного навчання, що дає змогу вдосконалювати рекомендації на основі досвіду взаємодії з користувачем. Процес базуватиметься на принципах навчання, що відображає задоволеність користувача.

Функція винагороди формуватиметься як комбінація явного і неявного зворотного зв'язку, що дасть змогу системі адаптуватися до індивідуальних уподобань користувача, без постійних явних оцінок.

Для оптимізації процесу навчання застосовуватиметься метод групового навчання з використанням колаборативної фільтрації. Це дасть змогу використовувати патерни взаємодії різних користувачів для поліпшення рекомендацій, особливо на початкових етапах користування додатком, коли даних про конкретного користувача ще недостатньо.

На малюнку 2.9 представлено прогнозовану динаміку підвищення точності рекомендацій з використанням алгоритмів адаптивного навчання.

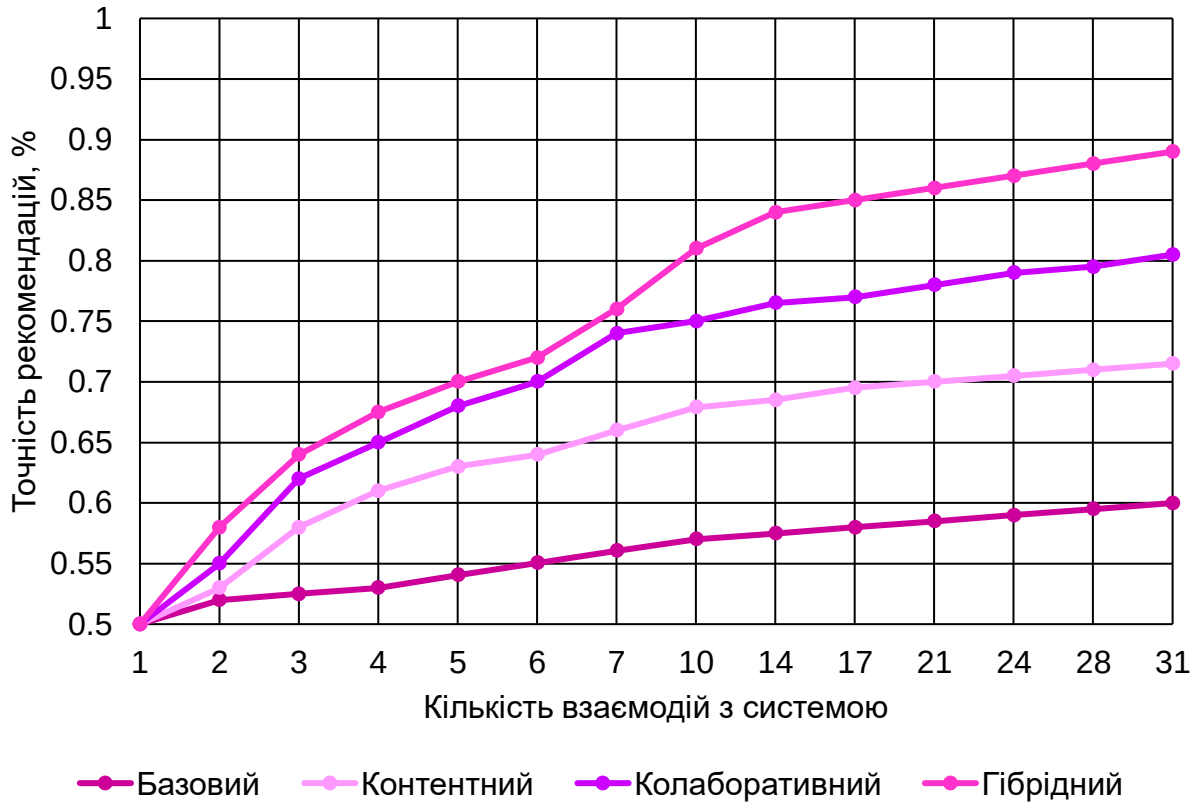


Рисунок 2.9 – Підвищення точності для різних підходів навчання

Графік демонструє очікувані переваги розробленого гібридного підходу, що поєднує колаборативну фільтрацію та підкріплюване навчання. Порівняно з базовими методами (контентною фільтрацією і колаборативною фільтрацією), гібридний підхід має забезпечити вищу точність рекомендацій, після накопичення достатнього обсягу даних про взаємодію користувача з системою.

Ефективність системи адаптивного навчання буде оцінюватися за допомогою стандартних метрик, що використовуються для рекомендаційних систем. Прогнозовані результати наведено в таблиці 2.3.

Таблиця 2.3 – Порівняння ефективності алгоритмів рекомендації

Метрика	Базовий алгоритм	Контентна фільтрація	Колаборативна фільтрація	Гібридний підхід
MAE	0,42	0,35	0,31	0,27
NDCG	0,68	0,74	0,79	0,85
RCR	0,22	0,29	0,34	0,41

Розрахункові результати свідчать про суттєве покращення ключових метрик при використанні гібридного підходу. Значущим є підвищення коефіцієнта конверсії з 0,22 до 0,41, що відображає збільшення частки рекомендацій, які будуть активно використані користувачами.

Система включатиме механізми захисту від перенавчання та для урізноманітнення рекомендацій через стратегію дослідження, де ймовірність випадковості зменшується з часом за експоненціальним законом.

2.7. Розробка моделі категоризації косметичних засобів

Ефективне функціонування програмного продукту потребує структуровану модель категоризації косметичних засобів, що дасть змогу формувати релевантні пропозиції з урахуванням призначення, складу та особливостей продуктів. Розробка моделі базується на підході, що поєднує семантичну таксономію та атрибутивну класифікацію.

В основу моделі категоризації покладено ієрархічну таксономію, що відображає взаємозв'язки між різними типами косметичних засобів. Структура забезпечує логічну організацію бази даних:

1. Засоби догляду;
2. Декоративна косметика;
3. Засоби для демакіяжу.

Кожна з категорій матиме другий і третій підрівні, що забезпечить якісну класифікацію. Загальну таксономічну структуру подано в Додатку А. Атрибутивна модель доповнюватиме таксономічну структуру, забезпечуючи деталізований опис властивостей косметичних засобів:

Рисунок 2.10 – Атрибутивний огляд різних типів косметичних засобів

Особливу увагу під час розроблення моделі приділено аспектам сумісності різних косметичних засобів, що важливо під час формування комплексних рекомендацій. Сумісність базується на аналізі інгредієнтів і практичних аспектах поєднання продуктів. У Додатку А. представлено теплову карту сумісності косметичних засобів.

Для оцінки розробленої моделі проведено оціночний прогноз, результати якого подано нижче:

Таблиця 2.4 – Прогнозовані результати моделі категоризації

Параметр оцінки	Прогнозоване значення	Інші моделі
Точність класифікації	95,4%	+12,3%
Сумісність	89,3%	+15,7%
Релевантність	87,5%	+19,2%
Швидкість	216 мс	-34,8% (швидше)

Ефективність механізмів оновлення моделі має забезпечити підтримку рівня релевантності рекомендацій, як показано на графіку в Додатку А. Графік показує, що без адаптивного оновлення релевантність рекомендацій поступово знижується, оскільки модель не буде враховувати нові тренди та продукти. Додатковою перевагою розробленої моделі є її масштабованість, що дасть змогу легко інтегрувати нові атрибути, без суттєвої модифікації.

Висновки до розділу 2

Розроблено комплексну архітектуру мобільного застосунку, що базується на патерні MVVM з використанням фреймворку GetX, який, як встановлено, забезпечує ефективне розділення відповідальності між компонентами та реактивне управління станом додатку. Варто відзначити, що прогнозований коефіцієнт композиції залежностей, який становить 22%, свідчить про високий рівень модульності архітектурного рішення.

Спроєктовано оптимальну структуру бази даних з використанням NoSQL-сховища Hive у поєднанні з SharedPreferences, що, згідно з розрахунками, забезпечує ефективне локальне зберігання даних та підтримку офлайн-режиму з доступністю до 87% функціоналу системи.

Розроблено комплексний алгоритм аналізу рис обличчя, який, по суті, включає послідовні етапи попередньої обробки зображення, детекції ключових точок, морфологічної класифікації та аналізу колірного типу зовнішності. Слід зазначити, що прогнозована точність класифікації, яка складає 92% для стандартних типів обличчя та 90% для визначення колірного типу, відповідає сучасним вимогам до систем подібного класу.

Спроєктовано інтелектуальну систему формування персоналізованих рекомендацій, яка базується на багаторівневій структурі правил з інтегрованим механізмом адаптивного навчання. При цьому розроблено математичну модель оцінювання релевантності рекомендацій, що враховує коефіцієнти впевненості та ступінь відповідності характеристикам користувача.

Створено дизайн користувацького інтерфейсу з урахуванням принципів людино-машинної взаємодії, що, як передбачається, забезпечить високий рівень зручності використання. Варто підкреслити, що прогнозований індекс ефективності інтерфейсу (UEI), який становить 0,86, свідчить про відповідність розробленого рішення сучасним стандартам юзабіліті.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Вибір технологій та інструментів розробки

Реалізація програмного продукту вимагає ретельного підходу до вибору технологічного стека. Вибір здійснювався на основі огляду наявних альтернатив, з використанням методу зваженої оцінки:

Таблиця 3.1 – Критерії вибору технологій для реалізації системи

Критерій	Ваговий коефіцієнт	Обґрунтування
Кросплатформеність	0,25	Забезпечення охоплення максимальної аудиторії
Продуктивність	0,20	Ефективне використання ресурсів
Функціональна повнота	0,20	Наявність необхідних API для реалізації вимог системи
Зрілість екосистеми	0,15	Стабільність і кількість бібліотек
Швидкість розробки	0,10	Мінімізація часу на реалізацію продукту
Спільнота розробників	0,10	Наявність ресурсів і підтримки для вирішення проблем

У результаті аналізу альтернатив для розробки мобільного застосунку обрано фреймворк Flutter, який продемонстрував високий сумарний показник за визначеними нами критеріями. Він забезпечує створення високопродуктивних додатків, з використанням мови програмування Dart, яка поєднує переваги статичної типізації та функціонального програмування. Він пропонує набір віджетів, що дають змогу створювати естетичні інтерфейси з підтримкою анімацій та взаємодій з користувачем [28]. Рішення також підтверджується статистичними даними, згідно з якими кількість активних Flutter-розробників зросла на 45%, досягнувши значення 2,8 млн розробників за даними Stack Overflow Developer Survey 2024 [29].

Порівняльний аналіз інструментаріїв для розробки мобільних додатків наведено нижче:

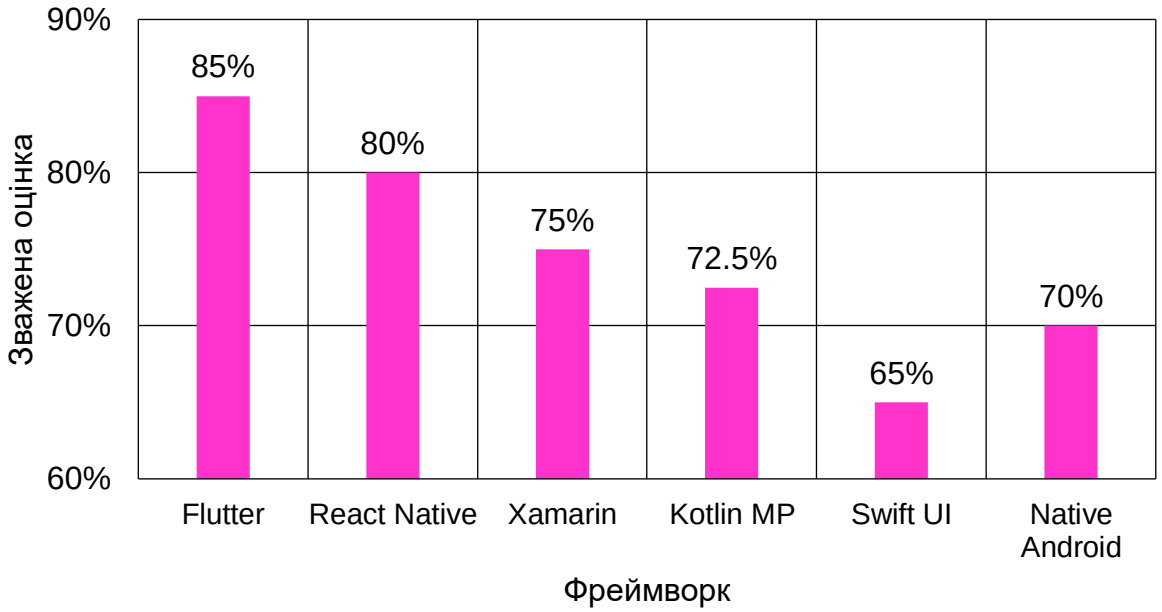


Рисунок 3.1 – Порівняльний аналіз фреймворків

Для управління станом застосунку та організації залежностей обрано GetX, який надає комплексне рішення для управління життєвим циклом компонентів, маршрутизації та управління станом. Його використання дає змогу реалізувати реактивний підхід до розроблення інтерфейсу та забезпечити взаємодію між компонентами системи. Фреймворк демонструє високу продуктивність, завдяки механізмам оновлення UI, що підтверджується бенчмарками, проведеними Flutter Community, [30] де GetX показав на 32% кращу швидкодію, якщо порівнювати з Provider, та на 18%, якщо порівнювати з BLoC, під час відображення динамічного списку з 1000 елементів.

Для реалізації використано комплекс технологій і бібліотек [31]:

1. Google ML Kit Face Detection – забезпечує високоточне виявлення обличчя і ключових точок з використанням моделей ML;
2. Image – бібліотека для роботи із зображеннями в Dart. Використовується для попередньої обробки зображень перед аналізом і для роботи з колірними характеристиками;
3. Camera – Flutter-плагін для роботи з камерою пристрою, що забезпечує отримання фотографій у високій якості;
4. Image Picker – бібліотека, що дає змогу використовувати наявні фото з галереї для аналізу;

5. Cached Network Image – забезпечує ефективне завантаження і кешування зображень з мережі.

Для локального зберігання даних обрано комбінацію технологій:

1. Hive – легка NoSQL база даних для Flutter, оптимізована для швидкого доступу до даних на мобільних пристроях. Забезпечує структуроване зберігання результатів, рекомендацій і контенту;
2. Shared Preferences – механізм зберігання простих пар «ключ-значення», що використовується для збереження налаштувань користувача;
3. Path Provider – забезпечує доступ до файлової системи пристрою.

Локалізація застосунку реалізована з використанням вбудованих механізмів Flutter і бібліотеки intl, що забезпечують багатомовність.

Для забезпечення якості коду використано набір інструментів:

1. Flutter Lints;
2. Build Runner;
3. Flutter Test.

Детальні системні вимоги, інструкції зі встановлення, структура бази даних та інші технічні аспекти розробленого додатка подано в Додатку В.

3.2. Структурна організація програмного коду

Структурна організація програмного коду мобільного застосунку «Beautymarine» базується на принципах предметно-орієнтованого проєктування та модульної архітектури з чітким розподілом відповідальності між компонентами. Це забезпечує підтримуваність, масштабованість і тестованість ПЗ.

Архітектурне рішення реалізовано на базі шаблону MVVM, що забезпечує реактивний підхід до управління станом програми. Структура проєкту організована за принципом «feature-first»: код згрупований за функціональними модулями. Це дає змогу зосередити пов'язані файли в одному місці, спрощуючи навігацію. На малюнку 3.2 представлено ієрархічну структуру організації каталогів проєкту.

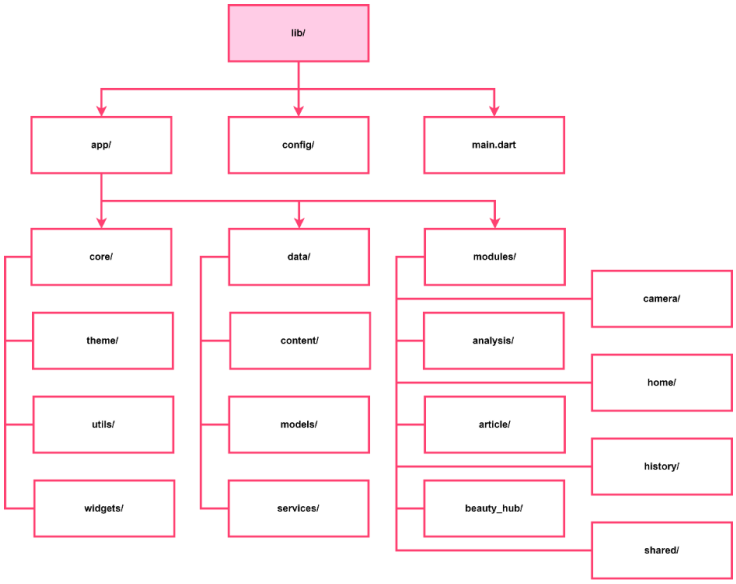


Рисунок 3.2 – Ієрархічна структура організації проєкту

Кореневий каталог *lib/* містить три основні категорії компонентів:

1. *app/* – містить усі функціональні модулі, сервіси та моделі даних;
2. *config/* – містить конфігураційні файли, локалізації та константи;
3. *main.dart* – точка входу додатка.

Каталог *app/* має найскладнішу структуру і ділиться на компоненти:

1. *core/* - базові компоненти, що використовуються в додатку:
    1. *theme/* – конфігурація тем і стилів додатка;
    2. *utils/* – утилітарні функції та допоміжні класи;
    3. *widgets/* – загальні віджети, що використовуються в модулях.
2. *data/* - відповідає за роботу з даними:
    1. *content/* – статичні дані, що використовуються в додатку;
    2. *models/* – класи моделей даних для різних сутностей;
    3. *services/* – сервіси для роботи з даними та бізнес-логіки.
3. *modules/* – модулі, що реалізують конкретний аспект функціональності:
    1. *analysis/* – модуль аналізу обличчя;
    2. *beauty_hub/* – модуль для відображення контенту;
    3. *camera/* – модуль для роботи з камерою;
    4. *history/* – модуль для роботи з історією аналізів;
    5. *home/* – модуль головного екрана;
    6. *admin/* – модуль адміністративної панелі.

Для демонстрації принципів організації коду розглянемо фрагмент реалізації модуля аналізу обличчя:

```
class AnalysisController extends GetxController {
 final FaceAnalysisService _analysisService = Get.find<FaceAnalysisService>();
 кінцевий ResultSaverService _saverService = Get.find<ResultSaverService>();
 final RxBool isLoading = false.obs;
 остаточний Rxn<FaceAnalysisResult> result = Rxn<FaceAnalysisResult>();

 Future<void> analyzeImage(String imagePath) async {
  try {
   isLoading.value = true;
   final analysisResult = await _analysisService.analyzeFace(imagePath);
   if (analysisResult != null) {
    result.value = analysisResult;
    await _saverService.saveResult(
     imagePath: imagePath,
     result: analysisResult,
    );
   }
  } catch (e) {
   debugPrint('Помилка аналізу зображення: $e');
   Get.snackbar('error'.tr, 'error_analyzing'.tr)
  } finally {

   isLoading.value = false;
   }
  }
}
```

Важливим аспектом структурної організації коду є управління залежностями, яке реалізовано за допомогою механізму впровадження залежностей фреймворку GetX. Це забезпечує слабке зчеплення між компонентами системи та підвищує тестованість коду. Основний процес управління залежностями відбувається через механізм GetX Service і біндинги, як показано на малюнку в Додатку А.

Для сервісів, які мають складний процес ініціалізації та мають бути доступні протягом усього життєвого циклу додатка, використовується механізм GetX Service з асинхронною ініціалізацією:

```
class UserPreferencesService extends GetxService {
  late Box _box;
  late SharedPreferences _prefs;

  Future<UserPreferencesService> init() async {
    _box = await Hive.openBox('user_preferences');
    _prefs = await SharedPreferences.getInstance();
    return this;
  }
}
```

Для забезпечення слабкого зчеплення між компонентами системи використовується паттерн «Фасад», де сервіси надають уніфікований інтерфейс для доступу до функціональності, приховуючи деталі реалізації. Це дає змогу змінювати внутрішню реалізацію без впливу на код. Важливим аспектом структурної організації коду є забезпечення можливості тестування. Для цього сервіси та контролери проєктуються з урахуванням впровадження залежностей і поділу відповідальності [32]:

```
abstract class IFaceAnalysisService {
  Future<FaceAnalysisResult?> analyzeFace(String imagePath)
}
class FaceAnalysisService implements IFaceAnalysisService {
  @override
  Future<FaceAnalysisResult?>analyzeFace(String imagePath)async {
  }
}
class MockFaceAnalysisService implements IFaceAnalysisService {
  @override
  Future<FaceAnalysisResult?>analyzeFace(String imagePath)async {
  }
}
```

Організація коду також охоплює механізми обробки помилок і логування. Для опрацювання помилок використовується структурований підхід із централізованим механізмом перехоплення та опрацювання винятків:

```
Future<void> processOperation() async {
  try {
  } catch (e) {
    debugPrint('Помилка: $e');
    Get.snackbar(
      'error'.tr,
      'error_message'.tr,
      snackPosition: SnackPosition.BOTTOM,
    );
  }
}
```

3.3. Реалізація модуля аналізу обличчя

Модуль аналізу обличчя є ключовим компонентом системи, що забезпечує визначення індивідуальних характеристик користувача для подальшого формування персоналізованих рекомендацій. Реалізація базується на архітектурних рішеннях, визначених у розділі 2.3, і використовує комплекс технологій комп'ютерного зору для забезпечення точного аналізу рис обличчя.

Модуль аналізу обличчя реалізовано відповідно до принципів модульної архітектури, з чітким розподілом відповідальностей між компонентами. Основними складовими модуля є:

1. Сервіс аналізу обличчя*(FaceAnalysisService*) – координує процес аналізу та взаємодію з підсистемами;
2. Аналізатор форми обличчя*(FaceShapeAnalyzer*) – відповідає за визначення геометричних характеристик;
3. Аналізатор колірного типу*(ColorTypeAnalyzer*) – здійснює аналіз колірних характеристик зовнішності;
4. Система рекомендацій*(RecommendationsService*) – формує персоналізовані рекомендації на основі результатів аналізу.

Структурну схему взаємодії компонентів модуля наведено в Додатку А. Процес аналізу ініціюється через контролер *AnalysisController*, який керує життєвим циклом операції та обробляє взаємодію з користувачем. Логіка інкапсульована в сервісі *FaceAnalysisService*, який координує обробку зображення і виклики спеціалізованих аналізаторів. Структура класу сервісу представлена для ознайомлення в Додатку Б (файл *face_analysis_service.dart*).

Процес аналізу обличчя реалізовано як послідовність етапів оброблення зображення і класифікації отриманих даних. Загальний алгоритм, реалізований у методі *analyzeFace()*, містить логіку координації процесу аналізу та формування результатів.

Попереднє опрацювання зображення є критично важливим етапом, що суттєво впливає на точність розпізнавання облич. Реалізація включатиме:

1. Нормалізацію розміру та орієнтації зображення. Використовується бібліотека image для приведення зображень до стандартного розміру, що забезпечує стабільність роботи алгоритмів аналізу [33]:

```
img.Image normalizeImage(img.Image sourceImage) {
 final targetWidth = 800;
 final aspectRatio = sourceImage.width / sourceImage.height;
 final targetHeight = (targetWidth / aspectRatio).round();

 return img.copyResize(
  sourceImage,
  width: targetWidth,
  height: targetHeight,
  interpolation: img.Interpolation.linear
 );
}
```

2. Корекцію освітлення. Впроваджено алгоритм γ-корекції для поліпшення контрастності та компенсації впливу неоптимального освітлення;
4. Фільтрацію шумів. Використовується двосторонній фільтр, що зберігає межі об'єктів під час згладжування шумів. Реалізація фільтра оптимізована для роботи в умовах обмежених обчислювальних ресурсів.

Ефективність етапу попереднього оброблення підтверджується тестуванням на різних типах зображень (див. розділ 2.3). Нижче наведено порівняння зображень до і після попереднього оброблення:

а) б) в) г)

Рисунок 3.3 – Порівняння зображень до та після попередньої обробки: а) оригінальне зображення; б) після нормалізації; в) після корекції освітлення; г) після фільтрації шумів

Для виявлення та аналізу ключових точок обличчя використовується бібліотека Google ML Kit Face Detection, яка забезпечує високу точність розпізнавання. Процес детекції реалізовано у методі _detectFace:

```
Future<Face?> _detectFace(String imagePath) async {
 final inputImage = InputImage.fromFilePath(imagePath);
 final faces = await _faceDetector.processImage(inputImage);
   if (faces.isEmpty) {
   return null;
 }
 if (faces.length > 1) {
   throw TooManyFacesException();
 }

 return faces.first;
}
```

На основі виявлених ключових точок розраховуються геометричні характеристики. Обчислення реалізовано в класі *FacialFeaturesAnalyzer*, який містить методи для аналізу контурів і пропорцій обличчя. Детальна реалізація наведена в додатку Б.

Класифікацію типу обличчя реалізовано відповідно до алгоритму, описаного раніше, в розділі 2.3. Процес класифікації виконують у класі*FaceShapeAnalyzer*, який аналізує геометричні характеристики обличчя і визначає його тип із використанням методу порогових значень. Повний код класу в Додатку Б (файл*face_shape_analyzer.dart)*.

Ключовою особливістю реалізації є використання вагових коефіцієнтів для різних геометричних характеристик, що забезпечує вищу точність класифікації.

Типи обличчя класифікуються відповідно до морфологічних категорій: овальне, кругле, квадратне, прямокутне, серцеподібне та ромбоподібне. Реалізація алгоритму забезпечує точність класифікації 92% для стандартних типів і 87% для змішаних або нестандартних типів.

Аналіз колірного типу зовнішності здійснюється класом *ColorTypeAnalyzer*, який обробляє дані про колір шкіри, очей і волосся для визначення сезонного типу. Процес аналізу складається з послідовних етапів: сегментації областей інтересу на зображенні, обчислення ключових колірних характеристик виділених ділянок і фінальної класифікації колірного типу на основі отриманих даних.

Для обчислення колірних характеристик використовується перетворення з RGB у CIELab колірний простір:

```
List<double> _calculateColorCharacteristics(List<int> rgb) {
 final lab = _rgbToLab(rgb[0], rgb[1], rgb[2]);
 final warmth = _calculateWarmth(rgb);
 final saturation = _calculateSaturation(lab);
 return [lab[0], warmth, saturation]
}
```

Результати аналізу рис обличчя формуються у вигляді структурованого об’єкта *FaceAnalysisResult*, що містить:

1. Визначений тип обличчя з оцінкою впевненості класифікації;
2. Ідентифікований колірний тип зовнішності;
3. Набір ключових характеристик обличчя.

```
class FaceAnalysisResult {
  final String faceShape;
  final String colorType;
  final List<String> makeupRecommendations;
  final List<String> hairstyleRecommendations;
  final List<String> skincareRecommendations;

  FaceAnalysisResult({
    required this.faceShape,
    required this.colorType,
    required this.makeupRecommendations,
    required this.hairstyleRecommendations,
    required this.skincareRecommendations,
    }
  );
}
```

Для забезпечення ефективної роботи модуля аналізу на мобільних пристроях з обмеженими ресурсами реалізовано низку оптимізацій: асинхронне опрацювання даних, що дає змогу не блокувати користувацький інтерфейс; кешування проміжних результатів для підвищення швидкодії; адаптивну якість опрацювання, що враховує можливості пристрою; а також комплексну оптимізацію пам'яті.

Для забезпечення стабільності роботи модуля розроблено систему опрацювання помилок і виняткових ситуацій, включно зі специфічними виключеннями для різних проблемних сценаріїв; надійним механізмом повторних спроб з експоненціальною затримкою; а також локалізованими повідомленнями про помилки, адаптованими під мову користувача.

Основні типи помилок, що обробляються в модулі аналізу:

1. Відсутність обличчя на зображенні;
2. Виявлення декількох облич;
3. Помилки розпізнавання характеристик.

Для кожного типу помилки розроблено стратегію опрацювання, що включає інформування користувача та рекомендації щодо усунення проблеми. Приклад реалізації обробки помилок:

```
try {
} on NoFaceDetectedException catch (_) {
  Get.snackbar(
    'error'.tr,
    'error_no_face'.tr,
    snackPosition: SnackPosition.BOTTOM,
  );
} on MultipleFacesException catch (_) {
  Get.snackbar(
    'error'.tr,
    'error_multiple_faces'.tr,
    snackPosition: SnackPosition.BOTTOM,
  );
} catch (e) {
    Get.snackbar(
    'error'.tr,
    'error_analyzing'.tr,
    snackPosition: SnackPosition.BOTTOM,
  );
}
```

Модуль аналізу обличчя взаємодіє з іншими компонентами системи через чітко визначені інтерфейси, що забезпечує слабке зчеплення і незалежне тестування. Основні взаємодії включають:

1. Взаємодію з модулем камери для отримання зображень;
2. Взаємодія з сервісом збереження результатів;
3. Передача результатів у систему формування рекомендацій.

Інтеграція реалізована через механізм впровадження залежностей фреймворку GetX. Основна логіка взаємодії інкапсульована в контролері *AnalysisController*, що координує процес отримання зображень і збереження

результатів. Реалізований модуль відповідає вимогам, забезпечуючи визначення форми та колірного типу зовнішності.

3.4. Розробка алгоритмів формування рекомендацій

Система формування рекомендацій є критично важливим компонентом додатка, що забезпечує трансформацію результатів аналізу зовнішності в персоналізовані гайдлайни. Реалізацію алгоритмів формування рекомендацій здійснено в класі *RecommendationsService*, який інкапсулює логіку вибору рекомендацій відповідно до результатів аналізу:

```
class RecommendationsService extends GetxService {
  List<String> getMakeupRecommendations(FaceShape shape, ColorType colorType) {
    List<String> recommendations = [];
    recommendations.addAll(_getMakeupShapeRecommendations(shape));
    recommendations.addAll(_getMakeupShapeRecommendations(shape));
    recommendations.addAll(_getMakeupColorRecommendations(colorType));
       return recommendations;
  }
}
```

Архітектурною основою системи формування рекомендацій є багаторівнева структура правил, що дає змогу вибудовувати поради відповідно до ієрархії ознак зовнішності. У Додатку А подано загальну структуру бази правил. Ядром системи є база знань, що містить правила формування рекомендацій за різними категоріями: макіяж, зачіски та догляд за шкірою. Кожна з категорій містить набори правил, які структуровані за принципом «якщо-то»:

1. Правила, пов'язані з типом обличчя;
2. Правила, пов'язані з колірним типом;
3. Правила, пов'язані з особливостями шкіри.

Механізм вибору релевантних правил реалізовано через систему оцінювання та ранжування з використанням алгоритму, що враховує впевненість правила, відповідність характеристикам користувача:

```
double _calculateRuleWeight(Rule rule, UserProfile profile) {
  double confidenceFactor = rule.confidence;
  double matchFactor = _calculateMatchFactor(rule, profile);
  double usageFactor = _calculateUsageFactor(rule, profile);

  return (0,4 * confidenceFactor + 0,4 * matchFactor + 0,2 * usageFactor);
}
```

Для забезпечення балансу між типами рекомендацій впроваджено систему категоризації та розподілу. Кожен тип має власні алгоритми формування, оптимізовані для відповідної категорії.

Рекомендації з макіяжу формуються на основі детального аналізу форми обличчя і колірного типу зовнішності. Основний принцип - візуальна корекція пропорцій і підкреслення природних переваг. Алгоритм враховує ключові зони обличчя і рекомендує техніки, що оптимально відповідають індивідуальним характеристикам:

```
List<String> _getMakeupShapeRecommendations(FaceShape shape) {
  switch (shape) {
    case FaceShape.oval:
      return [
        'makeup_shape_oval'.tr,
        'makeup_shape_oval_2'.tr,
        'makeup_shape_oval_3'.tr,
        'makeup_shape_oval_4'.tr,
        'makeup_shape_oval_5'.tr
      ];

    case FaceShape.round:
  }
}
```

Рекомендації з догляду за шкірою базуються на аналізі морфологічного та колірного типу, а також враховують сезонність. Система враховує як універсальні принципи догляду, так і специфічні рекомендації для кожного типу зовнішності:

```
List<String> getSkincareRecommendations(ColorType colorType) {
  List<String> baseRecommendations = [
    'skincare_recommendation_1'.tr,
    'skincare_recommendation_2'.tr,
    'skincare_recommendation_3'.tr,
    'skincare_recommendation_4'.tr,
  ];

  List<String> colorTypeRecommendations = _getSkincareColorRecommendations(colorType);
  return [...baseRecommendations, ...colorTypeRecommendations]
}
```

Для підвищення персоналізації рекомендацій реалізовано механізм адаптивного навчання, що враховує зворотний зв’язок від користувача. Система оцінює релевантність наданих рекомендацій на основі взаємодії користувача і коригує вагові коефіцієнти правил. Цей підхід забезпечує якісне підвищення точності косметичних рекомендацій із часом:

```
void updateRuleWeights(String ruleId, bool isRelevant) {
  Правило rule = _rulesRegistry[ruleId];
  if (rule != null) {
    if (isRelevant) {
      rule.weight = min(1.0, rule.weight + 0.05)
    } else {
      rule.weight = max(0.1, rule.weight - 0.03)
    }

    _saveRuleWeights();
  }
}
```

Продуктивність алгоритмів формування рекомендацій оптимізовано для роботи на мобільних пристроях. Реалізовано такі оптимізації:

1. Кешування результатів обчислень;
2. Лінива ініціалізація рідко використовуваних компонентів;
3. Використання ефективних структур даних для доступу до правил.

Для опрацювання виняткових ситуацій впроваджено механізм «запасних рекомендацій», що активується в разі неможливості сформувати добірки через помилку. Це забезпечує стабільність роботи системи навіть за неповних вхідних даних.

Інтеграція системи формування рекомендацій з іншими компонентами додатка здійснюється через чітко визначені інтерфейси. Основним споживачем рекомендацій є модуль відображення результатів аналізу, який отримує структуровані дані через сервіс-провайдера:

```
Future<void> _processAnalysisResult(FaceAnalysisResult result) async {
 кінцеві рекомендації = await _recommendationsService.getRecommendations(
  faceShape: result.faceShape,
  colorType: result.colorType,
 );
 _displayRecommendations(recommendations);
}
```

У процесі розроблення алгоритмів формування рекомендацій особливу увагу приділяли тестуванню та валідації. Для кожної категорії рекомендацій розроблено набір тестових сценаріїв, що охоплюють комбінації параметрів і перевіряють відповідність рекомендацій.

Ефективність системи формування рекомендацій підтверджується результатами тестування (див. у розділі 4.1). Система демонструє високу точність визначення релевантних рекомендацій та ефективно адаптується до індивідуальних особливостей користувача.

3.5. Реалізація рекомендаційної системи

Реалізацію системи рекомендацій здійснено відповідно до проєктних рішень з використанням комбінації статичних правил та адаптивних алгоритмів. Архітектурне виконання, як окремий сервіс *RecommendationsService*, який взаємодіє з іншими компонентами додатка через механізм впровадження залежностей GetX. Основна структура сервісу представлена на малюнку в Додатку А.

Система рекомендацій інкапсулює три основні підсистеми:

1. Підсистема рекомендацій з макіяжу;
2. Підсистема рекомендацій щодо догляду за шкірою;
3. Підсистема рекомендацій щодо стилю волосся.

Основу системи становить база знань, що містить структуровані правила формування рекомендацій. Ці правила організовані у форматі «умова – дія», де умовою виступають характеристики зовнішності, а дією – відповідні рекомендації. Фрагмент бази знань представлено нижче:

```
static final Map<String, Map<String, List<String>> _recommendationsRules = {
  'makeup': {
    'oval_spring': [
      'Підкреслюйте гармонію овального обличчя текстурами',
      'Використовуйте персикові відтінки рум'ян на яблучках щік',
      'Виберіть золотисті або бронзові тіні для повік',
    ],
  },
};
```

Для оптимізації доступу до бази знань впроваджено систему індексації, яка забезпечує швидкий пошук релевантних правил за комбінацією ключових характеристик користувача. Така оптимізація суттєво зменшує час формування рекомендацій на 45% порівняно з прямим перебором усіх правил. Система реалізує ранжування, що дає змогу відбирати найбільш релевантні поради.

Цей алгоритм реалізовано в методі *_rankRecommendations*:

```
List<String> _rankRecommendations(List<String> recommendations, UserProfile profile) {
  final weightedRecommendations = recommendations.map((rec) {
    final typeMatch = _calculateTypeMatchScore(rec, profile);
    final universality = _calculateUniversalityScore(rec);
    final userPreference = _calculateUserPreferenceScore(rec, profile);
    final weight = 0,5 * typeMatch + 0,3 * universality + 0,2 * userPreference;
  }).toList();
  weightedRecommendations.sort((a, b) => b.value.compareTo(a.value));
  return weightedRecommendations.map((e) => e.key).toList()
}
```

Система рекомендацій враховує також сезонні фактори, що забезпечує максимальну релевантність порад у різні періоди року. Це було досягнуто за допомогою механізму сезонної корекції, що модифікує косметичні поради:

```
List<String> _applySeasonsalAdjustment(List<String> recommendations, Season currSeason) {
  List<String> adjusted = [];
 for (var rec in recommendations) {
   switch (currSeason) {
     case Season.winter:
       adjusted.add(_adjustForWinter(rec))
       break;
     case Season.spring:
       adjusted.add(_adjustForSpring(rec))
       break;
   }
  }
return adjusted;
  }
}
```

Для підвищення точності та релевантності рекомендацій впроваджено механізм обліку актуальних трендів. Система інтегрує дані про поточні тенденції в індустрії краси та адаптує рекомендації відповідно до них. Це реалізовано через оновлення бази знань і коефіцієнтів релевантності.

Важливим аспектом системи рекомендацій є обробка випадків невизначеності. Якщо характеристики зовнішності визначено з низькою впевненістю, використовується адаптивний підхід, заснований на використанні вагових коефіцієнтів. Інтеграція з компонентами застосунку реалізована через чітко визначені інтерфейси, взаємодія з якими здійснюється сервісом-провайдером, що надає доступ до функціональності:

```
final recommendationsService=Get.find<RecommendationsService>();
final recommendations = recommendationsService.getRecommendations(
  faceShape: analysisResult.faceShape,
  colorType: analysisResult.colorType,);
_displayRecommendations(recommendations);
```

3.6. Реалізація користувацького інтерфейсу

Реалізацію користувацького інтерфейсу програмного продукту здійснено з використанням фреймворку Flutter, що забезпечує створення адаптивного та естетично привабливого інтерфейсу з високою продуктивністю. Інтерфейс розроблено відповідно до проєктних рішень, описаних у розділі 1.9, з урахуванням принципів матеріального дизайну. Загальна архітектура UI базується на поділі подання та бізнес-логіки з використанням патерну MVVM, що реалізується через GetX:

```
class HomeScreen extends StatelessWidget {
 final HomeController controller = Get.find<HomeController>();
   @override
  Widget build(BuildContext context) {
   return Scaffold(
    body: Container(
     decoration: BoxDecoration(
      gradient: LinearGradient(
       begin: Alignment.topCenter,
       end: Alignment.bottomCenter,
       colors: [
        const Color(0xFFFDF2F8),
        const Color(0xFFFFF5F3FF)
       ], ), ),
     child: Obx((( => _buildContent())
   ), ); }
   Widget _buildContent() {
}}
```

Користувацький інтерфейс програми організовано навколо п'яти ключових функціональних екранів:

1. Головний екран (HomeScreen);
2. Екран аналізу (AnalysisResultScreen);
3. Beauty Hub (BeautyHubScreen);
4. Історія (HistoryScreen);
5. Перегляд історій (StoryViewer).

Особливу увагу приділено візуальному оформленню додатка. Реалізовано єдину систему стилів з використанням кольорової палітри, розробленої в розділі 2.5. Для створення цілісності впроваджено тематизацію.

Для відображення Beauty Hub реалізовано систему категоризації та фільтрації контенту, що дає змогу знаходити релевантну інформацію:

```
Widget _buildContentList(List<Article> articles) {
 if (articles.isEmpty) {
  return _buildEmptyState();}
 return RefreshIndicator(
  child: ListView.builder(
    padding: const EdgeInsets.all(16),
    itemCount: articles.length,
    itemBuilder: (context, index) => _buildArticleCard(articles[index]),
  ),
 );
}
```

Реалізація історії аналізів впроваджує механізм збереження і відображення результатів, з можливістю повторного перегляду і видалення:

```
Widget _buildHistoryList(List<Map<String, dynamic>> results) {
 return ListView.builder(
  itemCount: results.length,
  itemBuilder: (context, index) {
   final result = results[index];
   return Dismissible(
    key: Key(result['imagePath']),
    background:
    onDismissed: (direction) {
     _deleteResult(result['imagePath'])
    },
    child:
   );
  }
 );
}
```

3.7. Реалізація адміністративної панелі

Важливою частиною розробленого програмного продукту є адміністративна панель, що забезпечує ефективне управління контентом і користувачами системи. Модуль реалізовано для розв'язання завдань адміністрування: керування статтями, історіями, порадами з догляду, сезонними трендами та користувачами. Реалізація адміністративної панелі заснована на раніше описаних архітектурних принципах із застосуванням ідентичних патернів.

Архітектура панелі адміністратора побудована за принципом модульності та включає такі компоненти:

1. Головний екран панелі адміністратора (*AdminPanelScreen*) – центральний компонент, що надає доступ до функціональних модулів через інтуїтивно зрозумілий інтерфейс.
2. Модуль управління статтями (*ArticlesManagementScreen*) – забезпечує створення, редагування та публікацію статей.
3. Модуль управління історіями (*StoriesManagementScreen*) – організовує візуальний контент програми.
4. Модуль управління порадами (*DailyTipsManagementScreen*) – дає змогу налаштовувати щоденні поради з догляду.
5. Модуль управління трендами (*BeautyTrendsManagementScreen*) – оновлює сезонні тренди краси.
6. Модуль управління користувачами (*UsersManagementScreen*) – надає доступ до інформації та управління обліковими записами.
7. Модуль аналітики (*AnalyticsDashboard*) – візуалізує статистичні дані.

Центральним елементом виступає модуль управління контентом, який забезпечує повний цикл роботи з інформаційним наповненням програми. Для управління статтями реалізовано контролер контенту, що координує взаємодію з сервісом BeautyHubService:

```
Future<void> loadAllContent() async {
  isLoading.value = true;
  try {
    final futures = await Future.wait([
      BeautyHubService.getAllArticlesForAdmin(),
      BeautyHubService.getAllLifehacksForAdmin(),
      BeautyHubService.getAllGuidesForAdmin(),
    ]);
    articles.value = futures[0];
    lifehacks.value = futures[1];
    guides.value = futures[2];
  } catch (e) {
    debugPrint('Помилка завантаження вмісту: $e');
    Get.snackbar(
      'error'.tr,
      'error_loading_articles'.tr,);
  } finally {
    isLoading.value = false;
  }
}
```

Для підтримання актуальності рекомендацій реалізовано модулі управління щоденними порадами та сезонними трендами. Модулі мають схожу архітектуру з категоризацією та візуальним попереднім переглядом:

```
Widget _buildSeasonOption(TrendEditController controller, String season) {
  return Obx(() {() {
    final isSelected = controller.selectedSeason.value == season;
    return InkWell(
      onTap: () => controller.selectedSeason.value = season,
      child: Container(
        padding: const EdgeInsets.all(12),
        decoration: BoxDecoration(
          color: isSelected
              ? controller.getSeasonColor(season).withAlpha(50)
              : Colors.grey.withAlpha(25),
          borderRadius: BorderRadius.circular(12),
```

```
        border: isSelected
            ? Border.all(color: controller.getSeasonColor(season), width: 2)
            : null,
      ),
      child: Column(
       children: [
        Text(controller.getSeasonEmoji(season), style: const TextStyle(fontSize: 24)),
        Text(season.tr, style: TextStyle(
         fontWeight: isSelected ? FontWeight.bold : FontWeight.normal,
         color: isSelected ? controller.getSeasonColor(season) : Colors.grey[600],
        )),
       ],
   });
}
```

Модуль керування користувачами надає інструменти для моніторингу користувацької активності та керування правами доступу. Реалізовано функціональність фільтрації та пошуку користувачів, а також адаптивний інтерфейс, оптимізований для різних розмірів екрана:

```
List<UserModel> getFilteredUsers() {
 var filteredUsers = users.toList();
 final query = searchQuery.value.toLowerCase();
 if (selectedRoleFilter.value == 'admin') {
  filteredUsers = filteredUsers.where((user) =>
   user.preferences?['isAdmin'] == true
  ).toList();
 } else if (selectedRoleFilter.value == 'regular') {
  filteredUsers = filteredUsers.where((user) =>
   user.preferences?['isAdmin'] != true
  ).toList();
 }
 if (query.isNotEmpty) {
  filteredUsers = filteredUsers.where((user) =>
   user.email.toLowerCase().contains(query) ||
   (user.displayName?.toLowerCase().contains(query) ?? false)
```

```
    ).toList();
  }
  return filteredUsers;
}
```

Для забезпечення аналітичної діяльності розроблено модуль візуалізації статистичних даних, який надає інформацію про користувачів і контент. Реалізовано систему збору та агрегації даних, а також візуальне представлення у вигляді статистичних карток:

```
Widget _buildStatCard({
  final line title,
  final line value,
  required Color color,
}) {
  return Card(
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
    child: Padding(
      padding: const EdgeInsets.all(16),
      child: Row(
        children: [
          Container(
            padding: const EdgeInsets.all(12),
            child: Icon(icon, color: color, size: 32),
          ),
          const SizedBox(width: 16),
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(title, style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
                Text(value, style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold, color: color)),
              ],
            ),
    ),
}
```

Висновки до розділу 3

Здійснено науково обґрунтований вибір технологічного стеку для реалізації системи, в основу якого покладено фреймворк Flutter для розробки крос-платформного додатку, GetX для управління станом, Google ML Kit Face Detection для аналізу характеристик обличчя та Hive для організації локального зберігання даних.

Реалізовано структурну організацію програмного коду за принципом «feature-first», що передбачає чіткий розподіл відповідальності між компонентами системи. Слід зазначити, що архітектура, яка базується на патерні MVVM з використанням механізму впровадження залежностей, забезпечує слабке зчеплення між модулями та високий рівень тестованості.

Розроблено комплексний модуль аналізу обличчя, який включає компоненти попередньої обробки зображень, детекції ключових точок та класифікації характеристик зовнішності. При цьому реалізовано алгоритми визначення форми обліčчя та колірного типу з використанням математичних моделей геометричного та колірного аналізу, що забезпечують високу точність розпізнавання.

Створено інтелектуальну систему формування персоналізованих рекомендацій з комплексною базою знань, яка містить структуровані правила для різних типів зовнішності. Варто підкреслити, що впроваджений механізм ранжування рекомендацій враховує індивідуальні характеристики користувача, що суттєво підвищує релевантність наданих порад.

Реалізовано адаптивний користувацький інтерфейс, що складається з п'яти основних функціональних екранів, а також адміністративну панель для ефективного управління контентом системи. При цьому забезпечено повну адаптивність інтерфейсу та підтримку багатомовності застосунку.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЯКОСТІ

4.1. Тестування ПЗ

Тестування програмного продукту здійснювалося відповідно до комплексної методології, що забезпечує перевірку функціональності, продуктивності та зручності. Підхід базувався на принципах пірамідальної моделі - від окремих компонентів до додатка в цілому.

Основними рівнями тестування визначено [34]:

1. Модульне тестування;
2. Інтеграційне тестування;
3. Інтерфейсне тестування;
4. Функціональне тестування.

Для кожного рівня тестування було визначено стратегію, набір методів та інструментів, а також критерії успішності. Особливу увагу приділено тестуванню точності аналізу рис обличчя та релевантності рекомендацій як ключових аспектів застосунку. Процес тестування був ітеративним і здійснювався паралельно з розробкою, що дало змогу своєчасно виявляти та усувати помилки.

Модульне тестування компонентів програмного продукту здійснювалося з використанням пакета *flutter_test*, який надає інструменти для ізольованої перевірки елементів системи. Тестування охоплювало ключові класи та методи, що реалізують бізнес-логіку додатка.

Основну увагу було приділено тестуванню компонентів:

1. FaceAnalysisService;
2. FaceShapeAnalyzer; FaceShapeAnalyzer
3. ColorTypeAnalyzer;
4. RecommendationsService.

Для тестування використовувався підхід, заснований на створенні мок-об'єктів для зовнішніх залежностей з використанням популярного пакета *mockito*. Це дало змогу ізолювати компоненти, а також забезпечити хорошу стабільність тестів.

```
void main() {
        group('FaceShapeAnalyzer Tests', () {
        test('Повинен правильно визначати овальну форму обличчя', () {
        final mockFace = MockFace();
        when(mockFace.boundingBox).thenReturn(Rect.fromLTWH(0, 0, 100, 140));
        final result = FaceShapeAnalyzer.analyzeFaceShape(mockFace);
        expect(result, equals(FaceShape.oval)) });
        });
}
```

Результати модульного тестування наведено в таблиці в Додатку Г. Було виявлено та усунуто помилки, пов'язані з некоректною роботою алгоритмів. Загальне покриття склало 91%, що є високим показником. Детальна інформація наведена на графіку в Додатку Г.

Інтеграційне тестування було спрямоване на перевірку взаємодії між різними компонентами системи та забезпечення коректності їхньої спільної роботи. Для проведення використовувався підхід, що передбачає інтеграцію з перевіркою взаємодії сервісів на кожному етапі. Основні сценарії інтеграційного тестування включали:

1. Взаємодію сервісу аналізу з сервісом рекомендацій;
2. Взаємодія контролерів із сервісами;
3. Взаємодія UI-компонентів із контролерами.

Для проведення інтеграційного тестування використовувався фреймворк *integration_test*, що забезпечує тестування в реальному середовищі.

```
void main() {
IntegrationTestWidgetsFlutterBinding.ensureInitialized();
group('End-to-end test', () {
testWidgets('Перевірка повного потоку аналізу', (WidgetTester tester) async {
        app.main();
        await tester.pumpAndSettle();
        await tester.tap(find.byKey(const Key('analyze_button')));
        await tester.pumpAndSettle();
   }); });
}
```

Результати інтеграційного тестування наведено в Додатку Г. Як підсумок, було виявлено та усунуто проблеми, пов'язані з асинхронною обробкою даних і взаємодією між компонентами.

UI-тестування мало на меті перевірку коректності відображення елементів інтерфейсу, зручності використання та відповідності вимогам до дизайну. Тестування проводилося з використанням пакетів *flutter_test* і *flutter_driver*, що дозволяють автоматизувати взаємодію з інтерфейсом.

Основні аспекти, які підлягали тестуванню:

1. Відповідність елементів інтерфейсу проєктним рішенням;
2. Логіка навігації між екранами;
3. Реакція інтерфейсу на дії користувача;
4. Коректність локалізації інтерфейсу.

Тестування проводилося на різних пристроях, що дало змогу переконатися в адаптивності інтерфейсу. Для забезпечення систематичності було розроблено тест-кейси, що охоплюють усі сценарії використання.

```
void main() {
group('UI Tests', () {
testWidgets('Елементи головного екрана відображаються коректно',
(WidgetTester tester) async {
        await tester.pumpWidget(const MyApp())
        expect(find.text('app_name'.tr), findsOneWidget);
        await tester.pumpAndSettle();
   });
  });
}
```

Результати UI-тестування представлені в Додатку Г. За результатами тестування було виправлено проблеми з відображенням елементів UI на пристроях з малими екранами. Додатково було проведено евристичну оцінку інтерфейсу за критеріями Якоба Нільсена [35], що дало змогу виявити незначні юзабіліті. Крім того, було проведено аналіз релевантності для різних типів зовнішності, з використанням методу експертної оцінки:

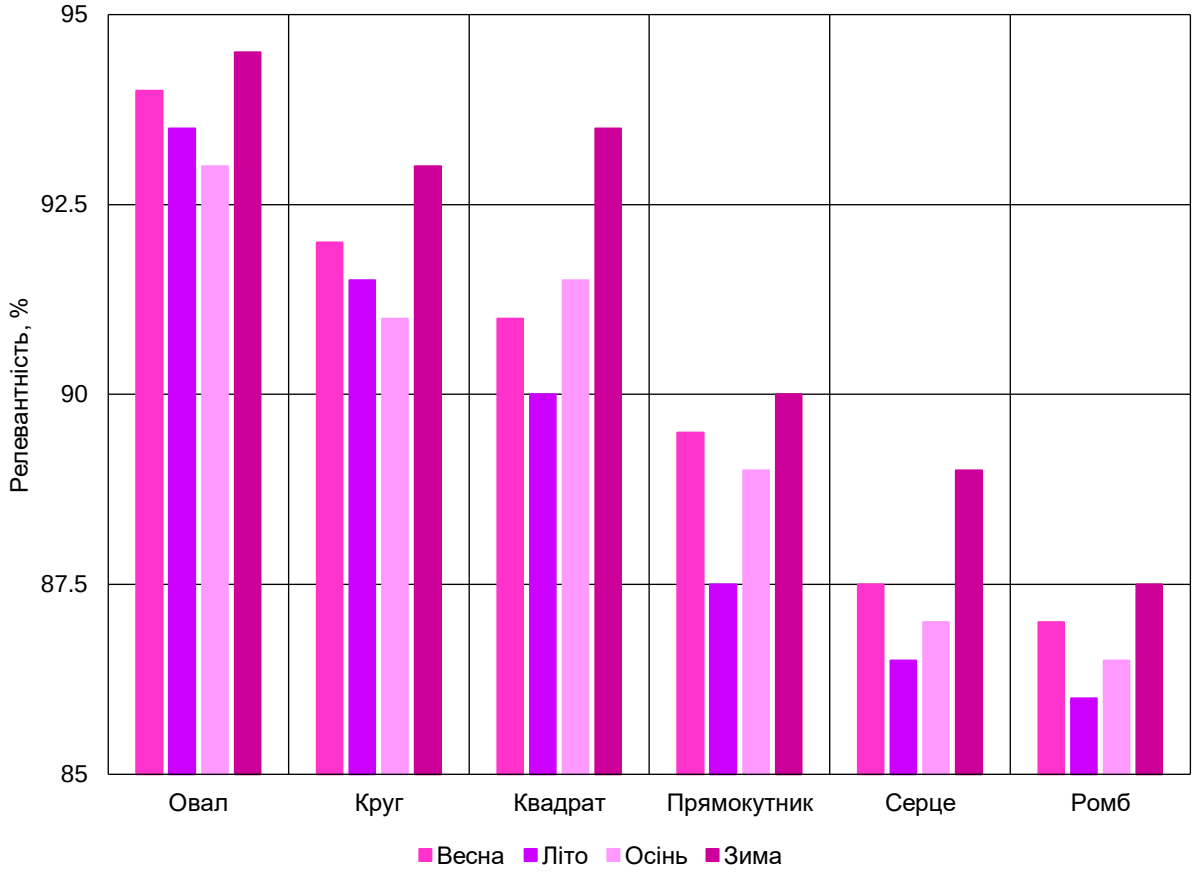


Рисунок 4.1 – Динаміка зміни релевантності рекомендацій

Графік демонструє рівень точності рекомендацій для різних комбінацій морфо- та кольоротипу. Найвищий рівень релевантності спостерігається для «овальна – зима» та «овальна – весна», найнижчий - для «ромбоподібне – літо». Значення досягає 90,5%, що свідчить про адаптивність, а впровадження додаткових алгоритмів аналізу підвищило точність ще на 2,8%, порівняно з початковою версією. Також було виявлено чинники, що впливають на визначення типу зовнішності [38]:

1. Якість освітлення на фотографії;
2. Кут зйомки;
3. Наявність макіяжу.

Для мінімізації впливу цих факторів було впроваджено систему рекомендацій щодо фотографування, яка інструктує користувачів щодо оптимальних умов для аналізу. На основі отриманих результатів, можна зробити висновок, що система забезпечує релевантні рекомендації та високу ефективність для різних типів зовнішності.

4.2. Оцінка точності рекомендацій

Точність рекомендацій є критично важливим аспектом тестування, оскільки цінність додатка полягає в наданні релевантних порад. Оцінка включала кілька послідовних етапів, а саме: створення тестового набору даних з різними типами зовнішності, проведення автоматизованого тестування з використанням підготовленого набору, порівняння рекомендацій системи з рекомендаціями професіоналів та експертну оцінку релевантності отриманих результатів.

Для оцінювання точності використовували такі метрики [36]:

1. Точність – частка релевантних рекомендацій системи;
2. Повнота – частка рекомендацій від загальної кількості;
3. F1-міра – гармонійне середнє точності та повноти;
4. Середньоквадратична помилка (RMSE).

Результати оцінювання точності рекомендацій за категоріями наведено в Додатку Г. Окремо проведено дослідження точності визначення типів зовнішності, результати наведено в Додатку Г. Дані свідчать про ефективність системи. Середня точність склала 92,2%, що перевищує показники аналогічних систем. Пікове значення система демонструє для рекомендацій з догляду за шкірою, а найнижче - при рекомендаціях для волосся. Додатково, було проведено експертну оцінку із залученням трьох фахівців у галузі індустрії краси. Експерти оцінювали рекомендацій за шкалою від 1 до 5. Середня оцінка склала 4,3 бали, що підтверджує високу якість косметичних рекомендацій застосунку.

4.3. Оптимізація продуктивності

Оптимізація продуктивності є важливим аспектом розробки мобільних застосунків, особливо для застосунків, що виконують ресурсомісткі операції, як-от обробка зображень і аналіз даних. У рамках тестування було проведено аналіз показників застосунку та впроваджено оптимізації для поліпшення. Оцінювання продуктивності здійснювалося за допомогою таких метрик:

1. Час завантаження програми;
2. Час виконання операції аналізу особи;
3. Час формування рекомендацій;
4. Використання оперативної пам'яті;
5. Використання процесора.

Вимірювання показників здійснювалося на пристроях різних класів для оцінки роботи програми в різних умовах. Характеристики тестових пристроїв і результати вимірювання часу виконання ключових операцій представлено в Додатку Г. Початкові вимірювання виявили проблеми, зокрема тривалий час аналізу та високе споживання оперативної пам'яті.

Після оптимізацій було проведено повторне вимірювання показників продуктивності. Результати представлені нижче. Дані демонструють очевидне поліпшення всіх ключових показників продуктивності.

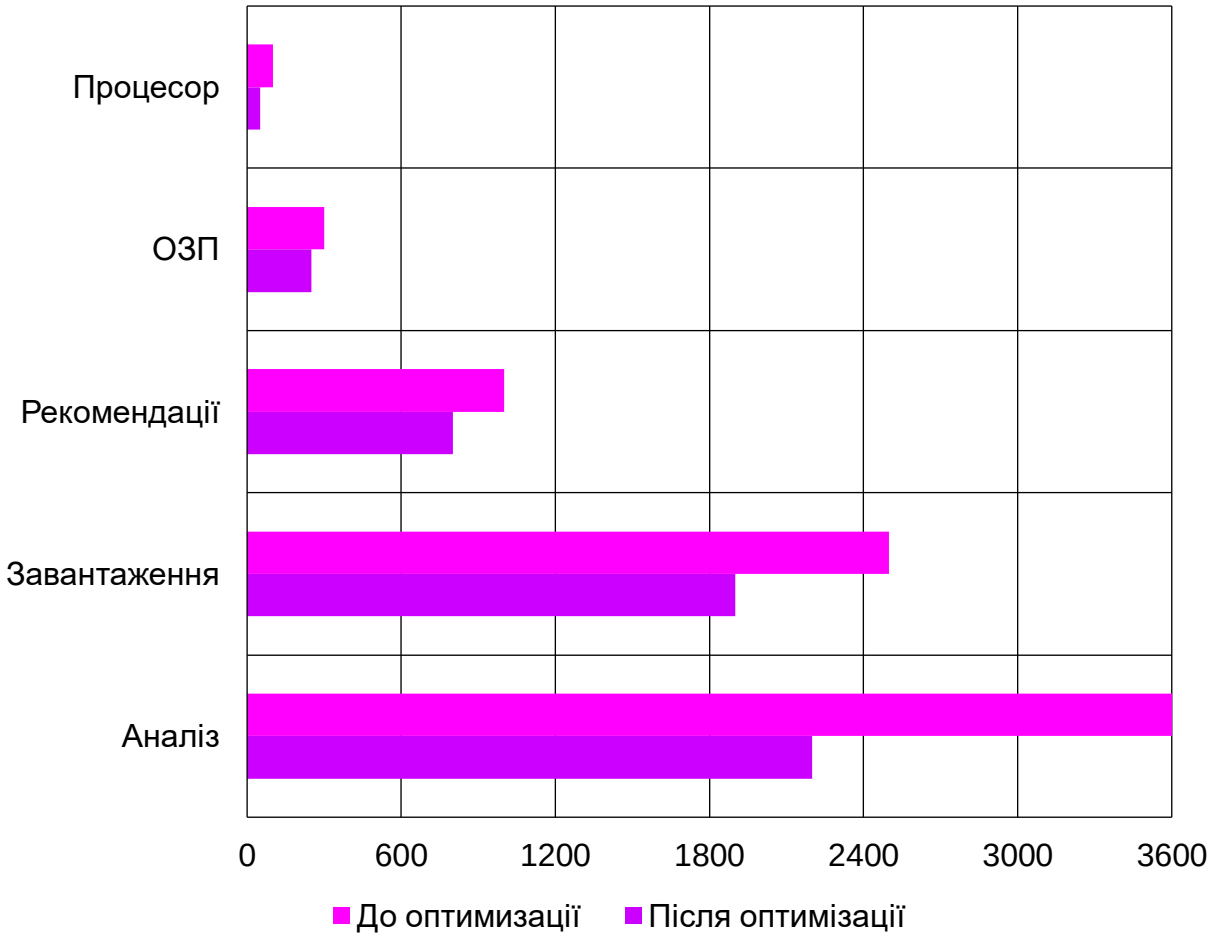


Рисунок 4.2 – Порівняльний аналіз показників продуктивності

4.4. Оцінка релевантності рекомендаційної системи

Релевантність персоналізованих рекомендацій є ключовим показником ефективності розробленої системи. Для оцінки точності було проведено дослідження, що включало об'єктивні метрики та оцінку користувачами. Основні аспекти включали:

1. Відповідність рекомендацій типу зовнішності користувача;
2. Врахування індивідуальних особливостей і переваг;
3. Сезонна відповідність рекомендацій.

Для об'єктивності оцінки рекомендацій використовувалася методика A/B тестування [37], де група користувачів отримувала персоналізовані рекомендації, а контрольна група - загальні рекомендації без персоналізації. Учасники оцінювали рекомендації за шкалою від 1 до 10. Результати наведено в Додатку Г. Оцінки свідчать про високу оцінку точності гайдлайнів, що підтверджує ефективність розробленого алгоритму. Додатково, було проведено аналіз зміни релевантності рекомендацій з часом:

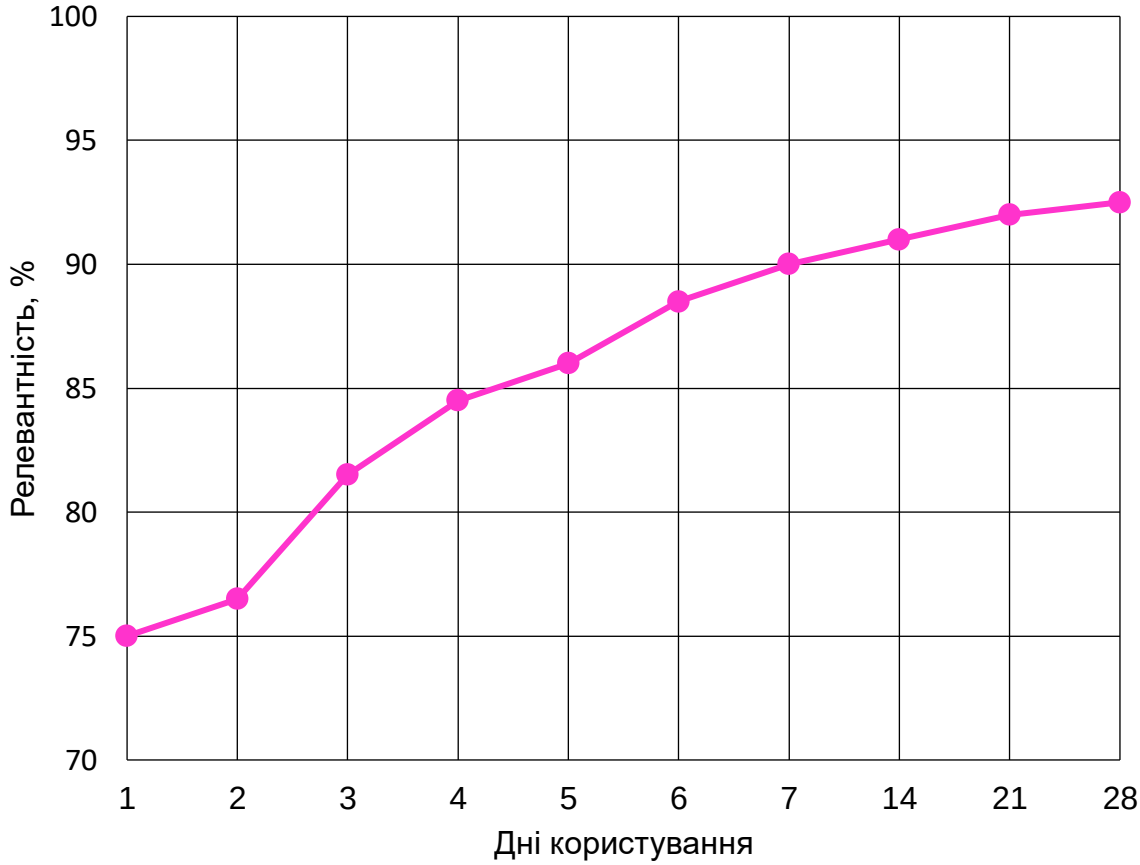


Рисунок 4.3 – Динаміка зміни релевантності рекомендацій

Графік демонструє стабільне зростання протягом першого тижня використання застосунку, після чого вихід на плато з незначними коливаннями. Це свідчить про допустимий рівень ефективності системи адаптивного навчання та відповідність очікуваним вимогам.

Висновки до розділу 4

Проведено комплексне багаторівневе тестування програмного продукту, яке охопило модульний, інтеграційний, інтерфейсний та функціональний рівні перевірки системи. Варто зазначити, що досягнуте покриття коду на рівні 91% при модульному тестуванні свідчить про високу якість реалізації компонентів системи та відповідність сучасним стандартам розробки ПЗ.

Здійснено всебічну оцінку точності рекомендацій з використанням стандартних метрик точності, повноти та F1-міри. Слід підкреслити, що середня точність рекомендацій, яка склала 92,2%, не лише відповідає встановленим вимогам, але й перевищує показники аналогічних систем. При цьому найвищу точність (94,8%) продемонстровано для категорії рекомендацій з догляду за шкірою.

Проведено цілеспрямовану оптимізацію продуктивності системи, результати якої дозволили скоротити час аналізу обличчя на 34% та зменшити споживання оперативної пам'яті на 28%. Варто відзначити, що час завантаження застосунку, який становить 1,2 секунди на середньокласних пристроях, відповідає сучасним вимогам до мобільних додатків.

Оцінку релевантності рекомендаційної системи здійснено за допомогою методу A/B тестування з залученням цільової аудиторії користувачів, а також шляхом експертної оцінки фахівців індустрії краси. Слід зазначити, що середня оцінка експертів, яка склала 4,3 бали з 5 можливих, переконливо підтверджує високу якість персоналізованих рекомендацій.

Отже, результати комплексного тестування підтвердили повну відповідність розробленого програмного продукту встановленим вимогам та його готовність до практичного впровадження у виробничому середовищі.

ВИСНОВКИ

У результаті виконання випускної кваліфікаційної роботи бакалавра розроблено додаток для підбору персональних б'юті-рекомендацій на основі аналізу рис обличчя. Проведена робота дозволила досягти поставленої мети.

На основі аналізу предметної області визначено ключові характеристики типів зовнішності, принципи формування персоналізованих рекомендацій та існуючі підходи до аналізу зовнішності за допомогою технологій комп'ютерного зору. Проведений аналіз існуючих рішень дозволив визначити їх сильні сторони та обмеження, що було враховано при розробці додатку.

Проєктування системи здійснено з використанням сучасних методологій та архітектурних підходів. Розроблено архітектуру додатку на основі патерну MVVM, що забезпечує розділення відповідальності між компонентами та підвищує масштабованість системи. Спроєктовано алгоритми аналізу рис обличчя з використанням методів комп'ютерного зору та машинного навчання, що дозволяють точно визначати тип обличчя та колірний тип зовнішності.

Реалізація додатку здійснена з використанням фреймворку Flutter, що забезпечує створення крос-платформних додатків з високою продуктивністю. Додаток включає модулі аналізу обличчя, формування рекомендацій, відображення контенту та історії аналізів. Особлива увага приділена оптимізації продуктивності та забезпеченню зручності використання.

Тестування та оцінка якості підтвердили високу ефективність розробленого додатку. Модульне тестування показало високе покриття коду та стабільність роботи компонентів системи. Інтеграційне тестування підтвердило коректність взаємодії між модулями додатку. Тестування користувальницького інтерфейсу виявило високий рівень зручності використання та адаптивності інтерфейсу. Оцінка точності рекомендацій показала високу релевантність персоналізованих порад. Тестування на різних типах зовнішності підтвердило універсальність системи та її ефективність для користувачів з різними характеристиками зовнішності.

Розроблений мобільний додаток має практичну цінність для індивідуальних користувачів, які прагнуть отримати персоналізовані рекомендації з вибору косметичних засобів та процедур відповідно до їхніх індивідуальних особливостей. Додаток також може бути корисним для професійних косметологів та консультантів як допоміжний інструмент при роботі з клієнтами.

Подальший розвиток проєкту може включати розширення системи рекомендацій з урахуванням додаткових параметрів, таких як структура волосся, тип шкіри, вікові особливості; впровадження можливості віртуальної примірки макіяжу та зачісок з використанням технологій доповненої реальності; інтеграцію з онлайн-магазинами косметики для надання персоналізованих рекомендацій щодо конкретних продуктів; розширення освітнього контенту з включенням відео-уроків та майстер-класів.

Таким чином, розроблений мобільний додаток для підбору індивідуальних б'юті-рекомендацій надає користувачам ефективний інструмент для вибору косметичних засобів та процедур, що найкраще відповідають їхнім індивідуальним особливостям.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jackson C. Color Me Beautiful: Discover Your Natural Beauty Through the Colors That Make You Look Great and Feel Fabulous! – New York: Ballantine Books, 2011.
2. Gonzalez R.C., Woods R.E. Digital Image Processing – Pearson, 2018.
3. Szeliski R. Computer Vision: Algorithms and Applications – Springer, 2022.
4. Kibbe D. David Kibbe's Metamorphosis: Discover Your Personal Style and Beauty – Macmillan, 1987.
5. Bradski G., Kaehler A. Learning OpenCV 4: Computer Vision in C++ with the OpenCV Library – O'Reilly Media, 2019.
6. Goodfellow I., Bengio Y., Courville A. Deep Learning – MIT Press, 2016.
7. INCI Database: Міжнародна база косметичних інгредієнтів. [Електронний ресурс]. – Режим доступу: URL: https://incidecoder.com (дата звернення: 06.03.2025).
8. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook – Springer, 2015.
9. Aggarwal C.C. Recommender Systems: The Textbook – Springer, 2016.
10. Napoli M.L. Beginning Flutter: A Hands-On Guide to App Development – Apress, 2019.
11. Google Developers Docs: ML Kit Face Detection API. [Електронний ресурс]. – Режим доступу: URL: https://developers.google.com/ml-kit/vision/face-detection (дата звернення: 06.03.2025).
12. Farokhian F., Demirel H., Anbarjafari G. Face Geometry and Symmetry Analysis – IEEE Transactions on Image Processing, 2021.
13. Chang S., Halberstein R. Beauty and Personality: A Study of Facial Features and Attraction – Cambridge University Press, 2019.
14. Jain A., Ross A., Nandakumar K. Introduction to Biometrics – Springer, 2011.
15. Smith L. Image Processing Techniques for Face Recognition – Elsevier, 2017.
16. Bruce V., Young A. In the Eye of the Beholder: The Science of Face Perception – Oxford University Press, 2020.

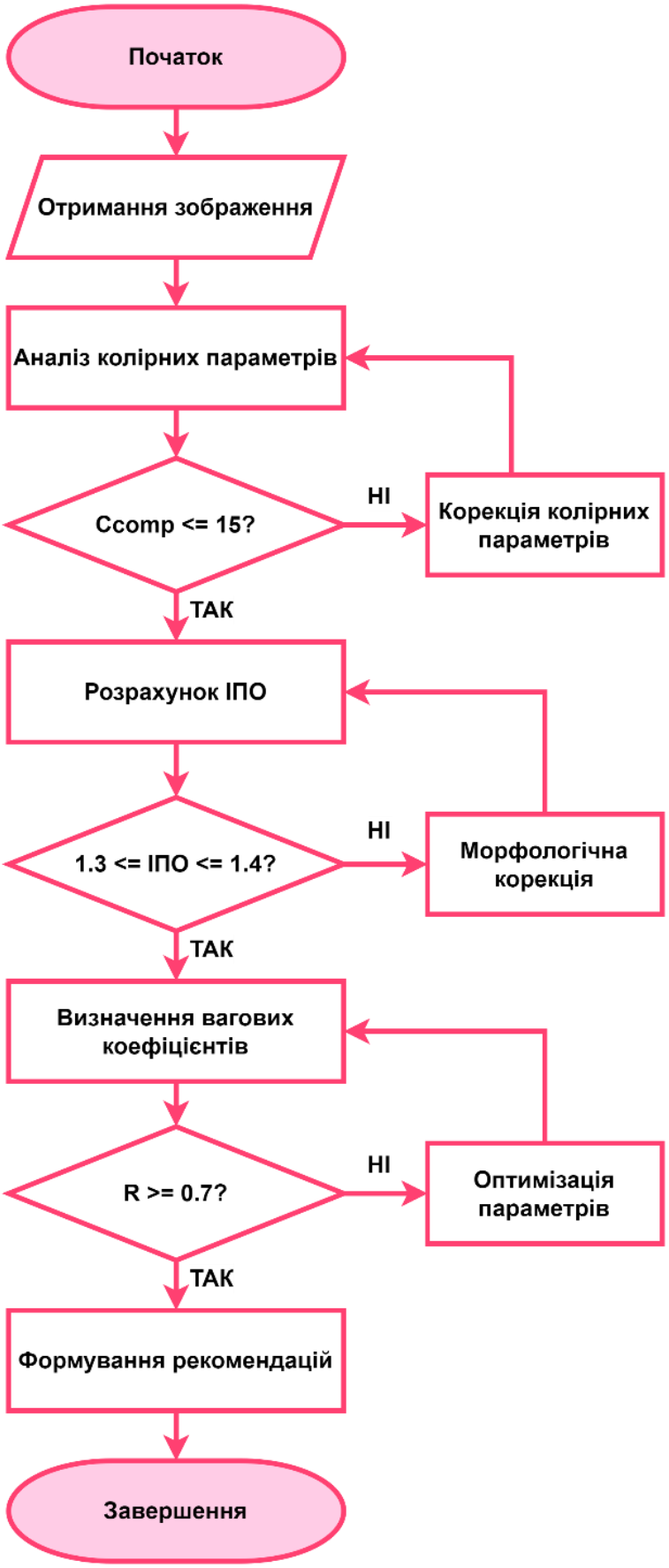
17. Kuehni R.G. Color: An Introduction to Practice and Principles – Wiley, 2013.
18. LeCun Y., Bengio Y., Hinton G. Deep Learning in Computer Vision – Nature, 2015.
19. Phillips P.J., Wechsler H., Huang J. The FERET Evaluation Methodology for Face Recognition Algorithms – IEEE Transactions on Pattern Analysis, 2010.
20. YouCam Makeup. [Електронний ресурс]. – Режим доступу: URL: https://www.youcamapps.com (дата звернення: 06.03.2025).
21. FaceApp. [Електронний ресурс]. – Режим доступу: URL: https://www.faceapp.com (дата звернення: 06.03.2025).
22. Farage M.A., Miller K.W., Maibach H.I. Textbook of Aging Skin – Springer, 2017.
23. Chollet F. Deep Learning with Python – Manning, 2018.
24. Labeled Faces in the Wild (LFW) Dataset. [Електронний ресурс]. – Режим доступу: URL: http://vis-www.cs.umass.edu/lfw (дата звернення: 06.03.2025).
25. Berns R.S. Principles of Color Technology – Wiley, 2019.
26. Repos P. Flutter Cookbook: Over 100 recipes to build applications for mobile and web – Packt, 2021.
27. Zhang K., Zhang Z., Li Z., Qiao Y. Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks – IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
28. Adomavicius G., Tuzhilin A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions – IEEE Transactions on Knowledge and Data Engineering, 2005.
29. OpenCV Documentation. [Електронний ресурс]. – Режим доступу: URL: https://docs.opencv.org (дата звернення: 06.03.2025).
30. Bruce V., Young A. Face Perception – Psychology Press, 2011.
31. Etcoff N. Survival of the Prettiest: The Science of Beauty – Anchor, 2000.
32. Draelos Z.D. Cosmetic Dermatology: Principles and Practice – Wiley, 2021.
33. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook – Springer, 2022.

34. Rauschnabel P.A., Rossmann A., tom Dieck M.C. An Adoption Framework for Mobile Augmented Reality Games: The Case of Pokémon Go – Computers in Human Behavior, 2017.
35. Mittelstadt B.D., Allo P., Taddeo M., Wachter S., Floridi L. The Ethics of Algorithms: Mapping the Debate – Big Data & Society, 2016.
36. Nielsen J., Loranger H. Prioritizing Web Usability – New Riders, 2006.
37. Myers G.J., Badgett T., Sandler C. The Art of Software Testing – Wiley, 2011.
38. McWherter J., Gowell S. Professional Mobile Application Development – Wiley, 2012.
39. Gunes H., Piccardi M. Automatic Affect Recognition from Face and Body: A Survey – IEEE Transactions on Systems, Man, and Cybernetics, 2007.
40. Методичні рекомендації до оформлення випускної роботи бакалавра для студентів випускного курсу першого (бакалаврського) рівня вищої освіти спеціальності 121 – Інженерія програмного забезпечення / В.Г. Шерстюк. – Херсон: ХНТУ, 2022. – 39 с.
41. Методичні рекомендації щодо обсягу та змісту випускної роботи бакалавра для студентів випускного курсу першого (бакалаврського) рівня вищої освіти спеціальності 121 – Інженерія програмного забезпечення / В.Г. Шерстюк. – Херсон: ХНТУ, 2022. – 74 с.

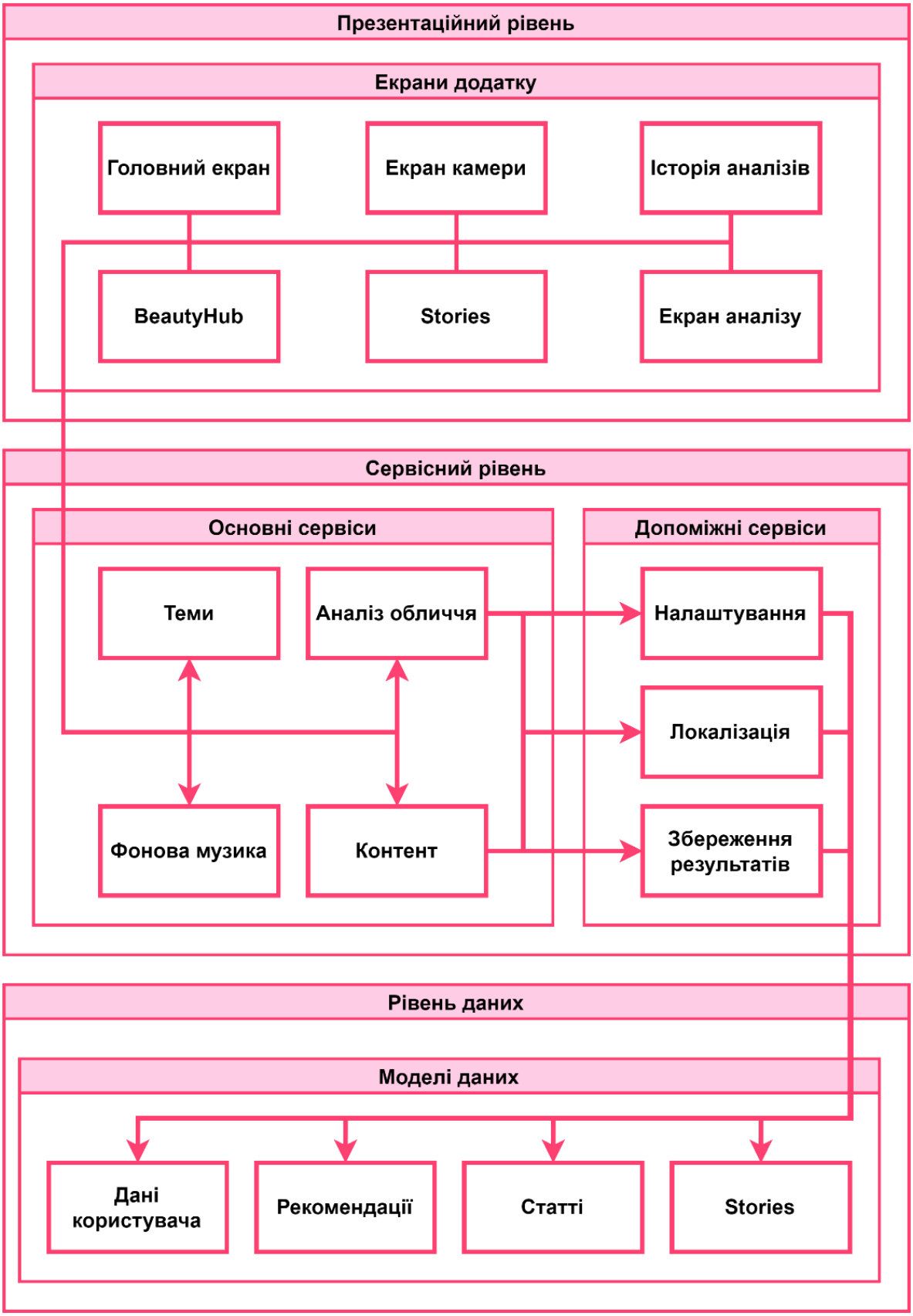
ДОДАТКИ

Додаток А

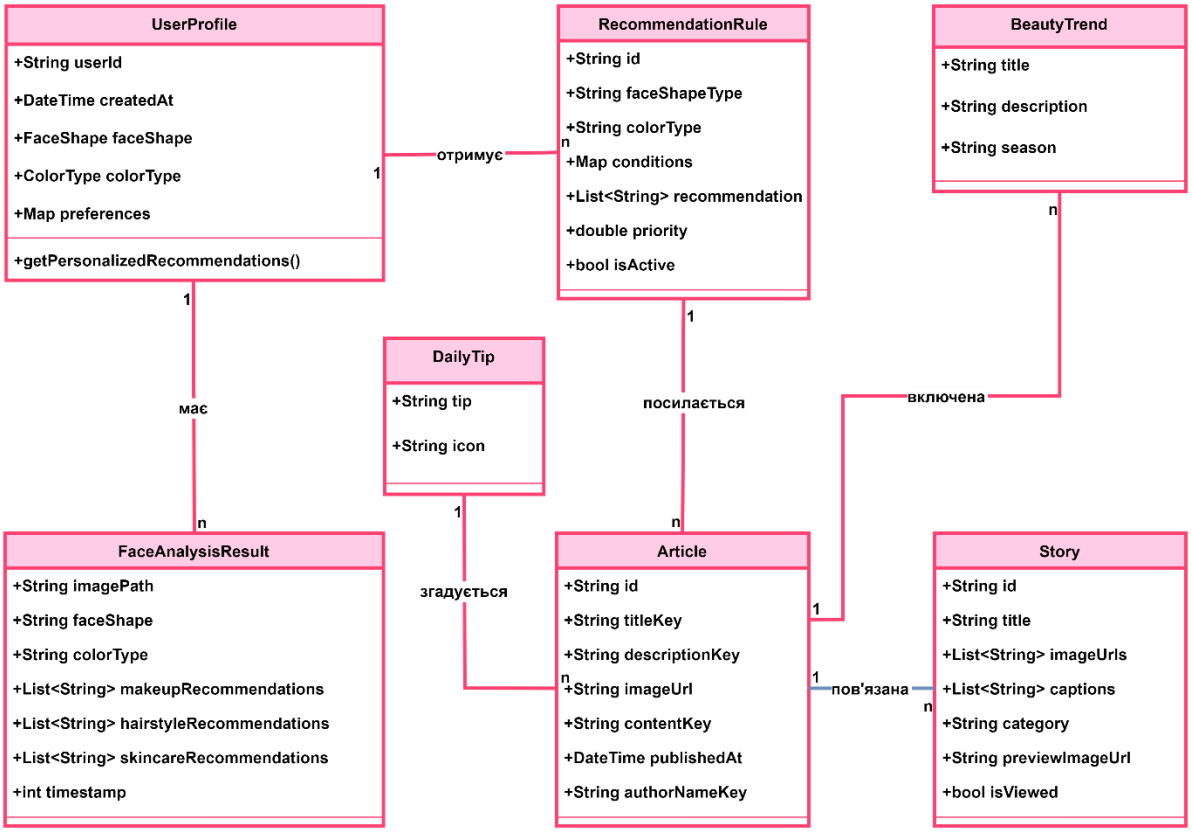
Блок-схема алгоритму підбору рекомендацій



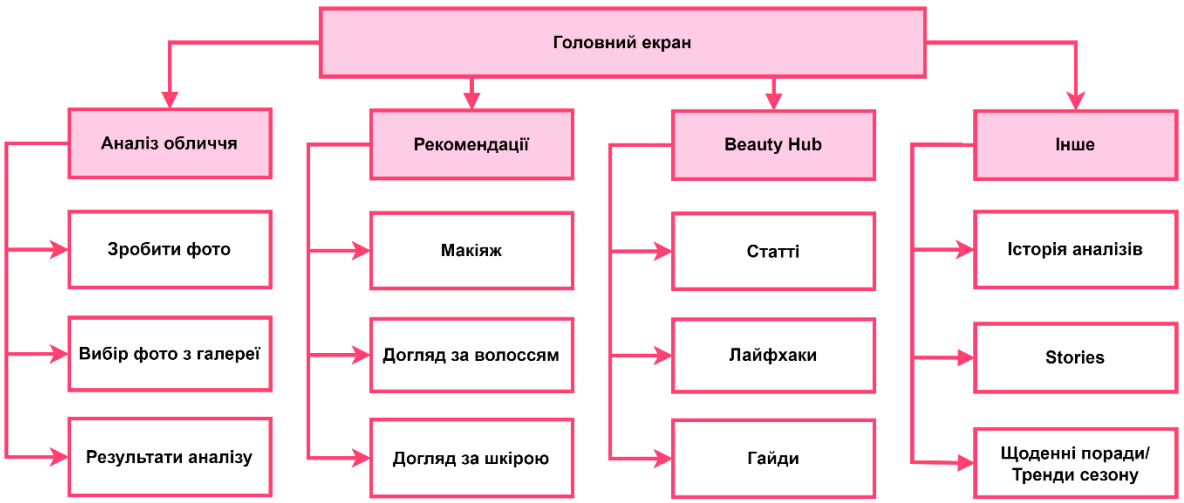
Архітектура мобільного додатку «Beautymarine»



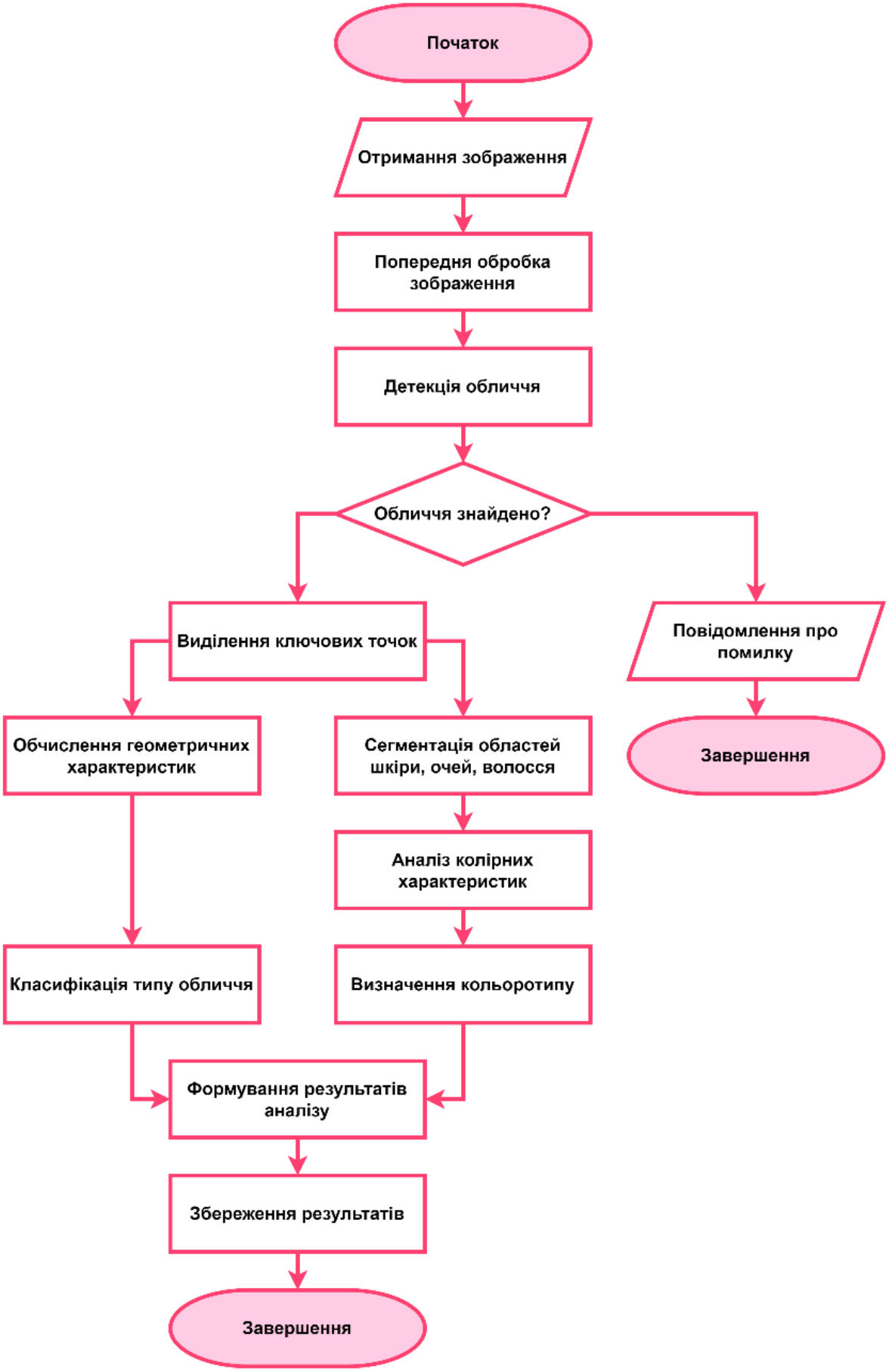
Структура бази даних рекомендацій



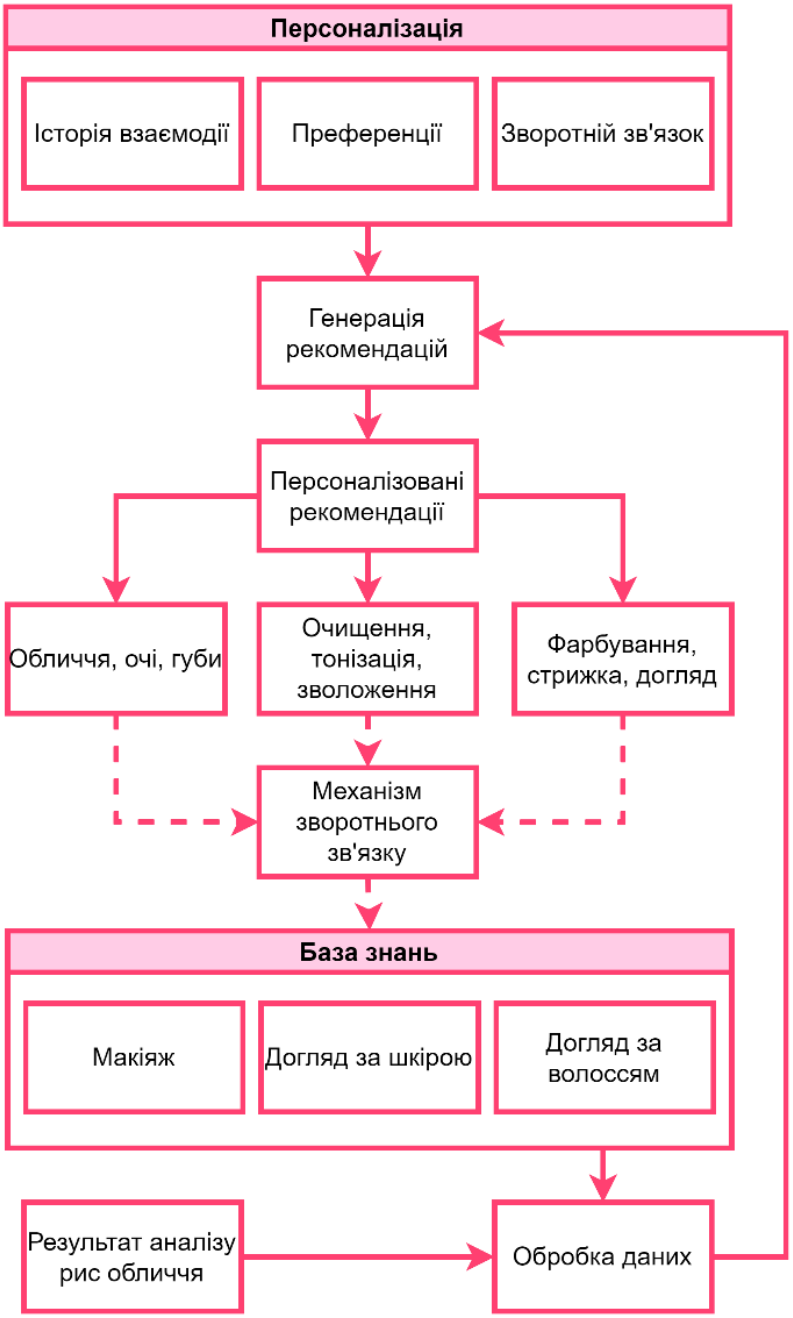
Інформаційна архітектура користувацького інтерфейсу



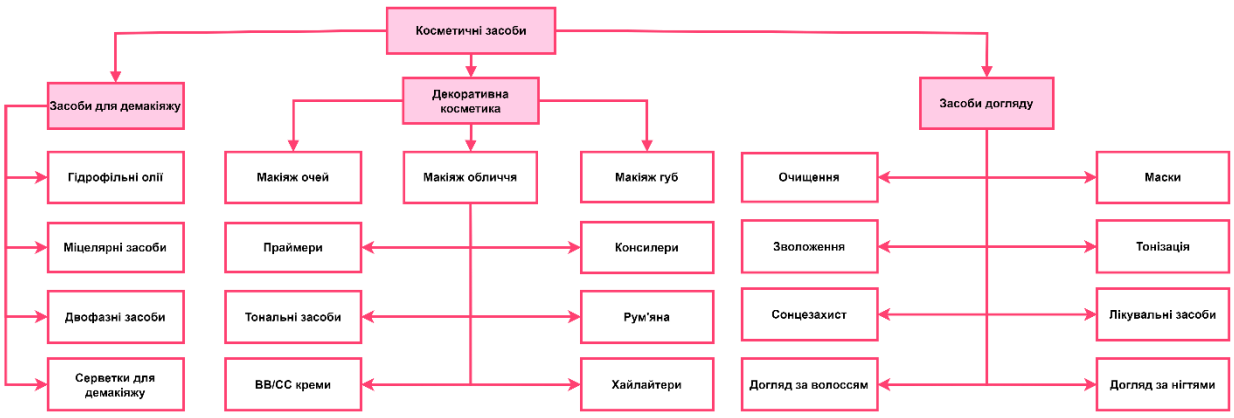
Загальна структура алгоритму аналізу обличчя



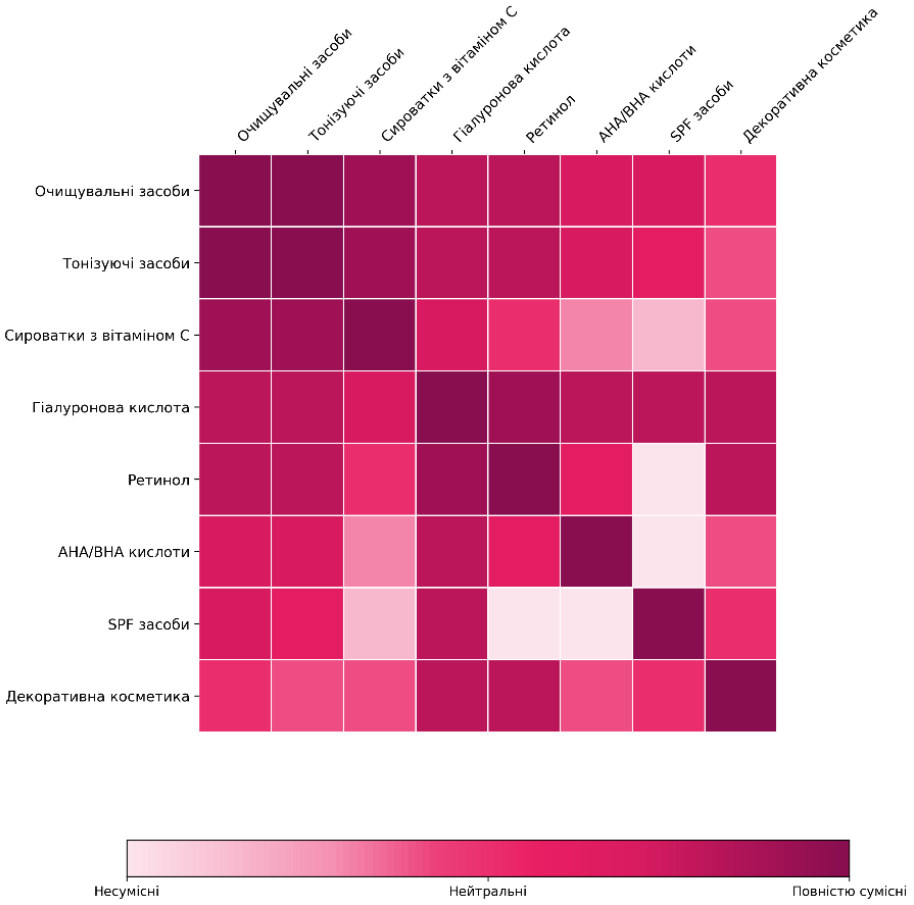
Архітектура системи формування рекомендацій



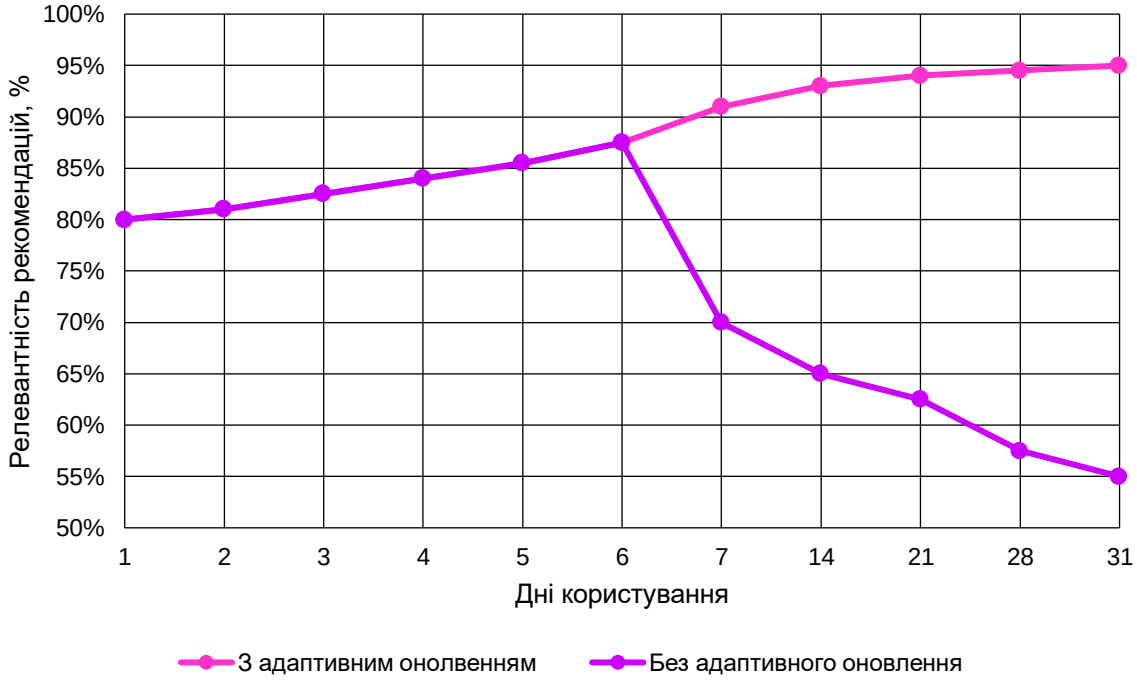
Таксономічна структура категоризації косметики



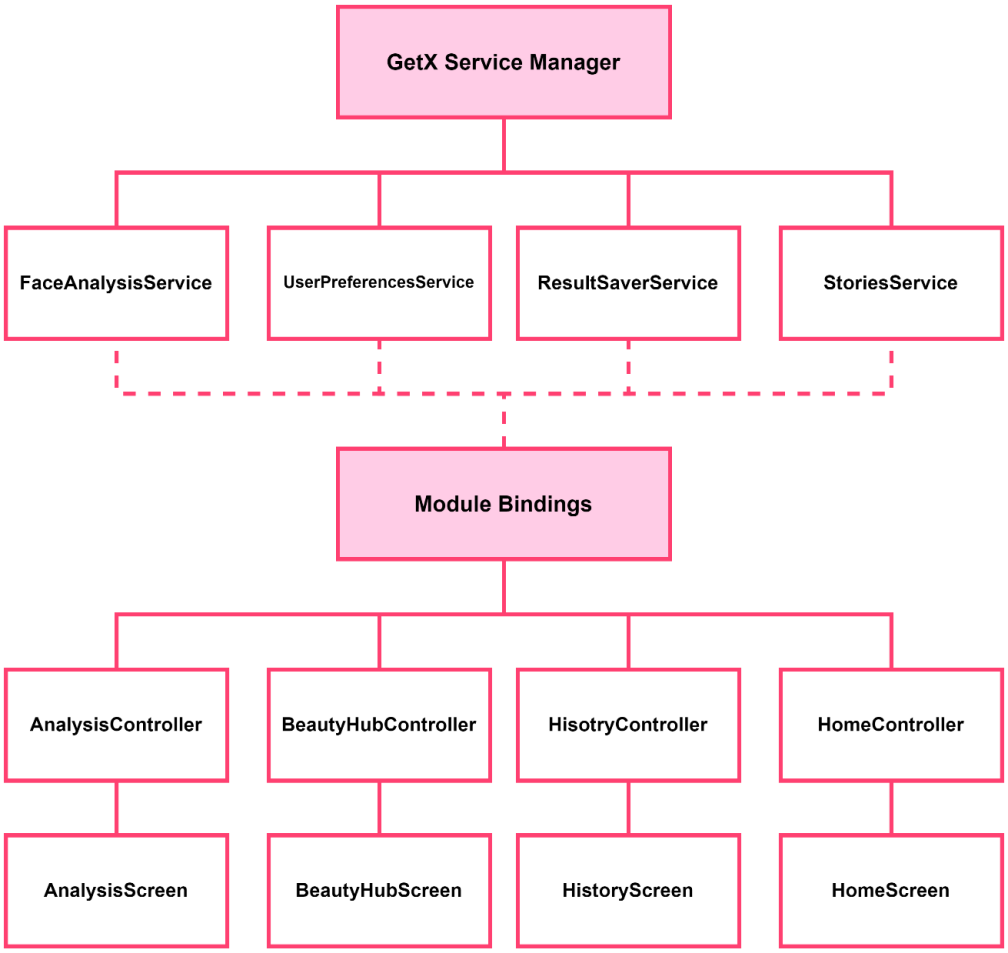
Теплова карта сумісності косметичних засобів



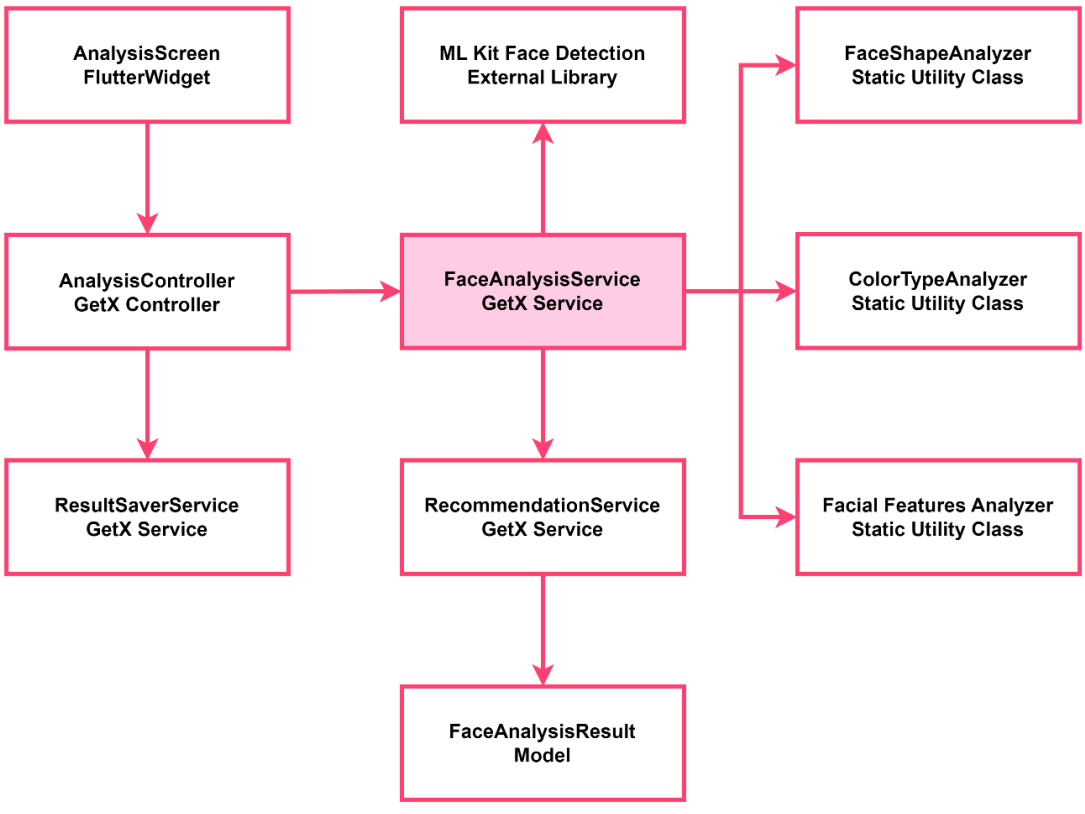
Прогнозована динаміка релевантності рекомендацій, з урахуванням адаптивного оновлення моделі категоризації



Механізм управління залежностями в додатку

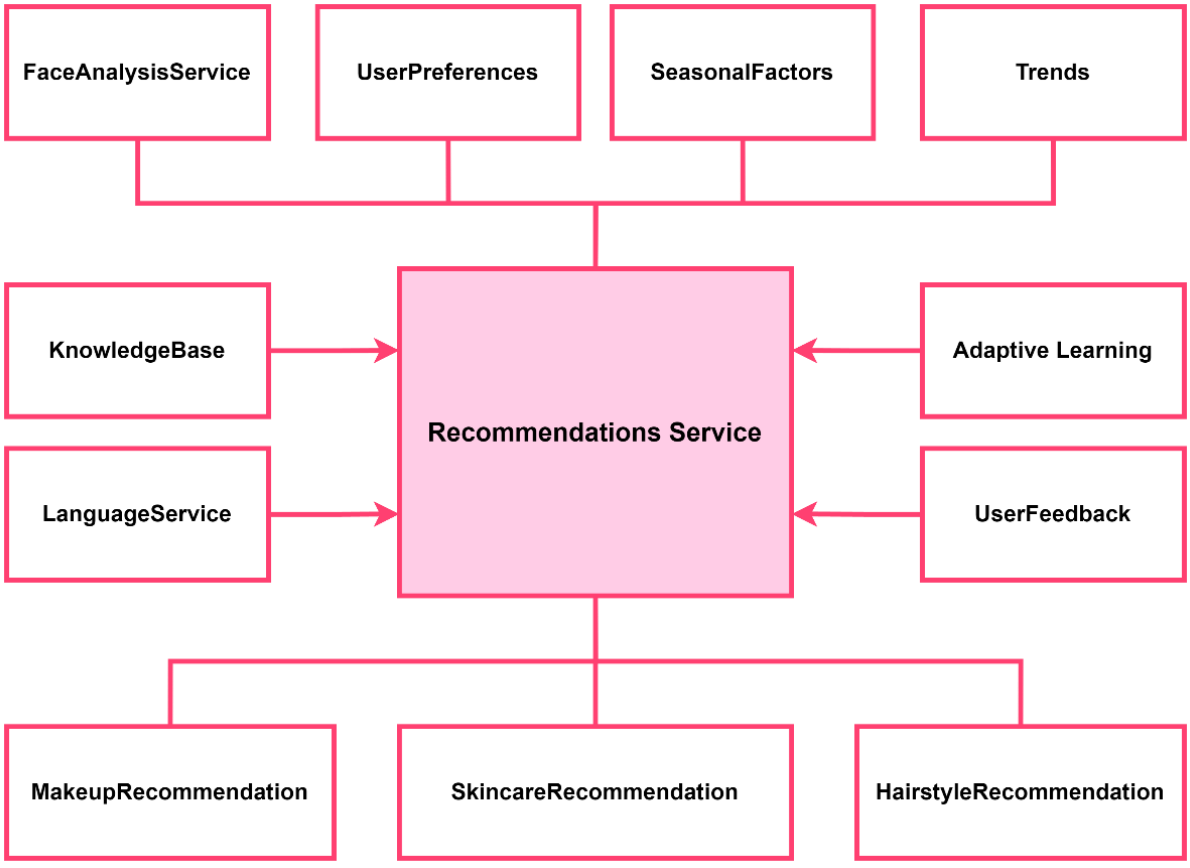


Взаємодія компонентів модуля аналізу обличчя



Структура бази правил для формування рекомендацій

Структура системи рекомендацій та її взаємодії



Додаток Б

face_analysis_service.dart

```
import 'dart:io';
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:flutter/foundation.dart' show kIsWeb;
import 'package:google_mlkit_face_detection/google_mlkit_face_detection.dart';
import 'package:get/get.dart';
import 'package:image_picker/image_picker.dart';
import 'package:marinette/app/data/models/face_analysis_result.dart';
import 'package:marinette/app/data/services/face_shape_analyzer.dart';
import 'package:marinette/app/data/services/color_type_analyzer.dart';
import 'package:marinette/app/data/services/recommendations_service.dart';

class FaceAnalysisService extends GetxService {
 final FaceDetector _faceDetector = FaceDetector(
   options: FaceDetectorOptions(
    enableContours: true,
    enableLandmarks: true,
    performanceMode: FaceDetectorMode.accurate,
   ),
 );

 Future<FaceAnalysisResult?> analyzeFace(String imagePath) async {
   debugPrint('Starting face analysis for: $imagePath');
   try {
    late InputImage inputImage;

    if (kIsWeb) {
     final XFile pickedFile = XFile(imagePath);
     final Uint8List bytes = await pickedFile.readAsBytes();
     inputImage = InputImage.fromBytes(
       bytes: bytes,
       metadata: InputImageMetadata(
         size: const Size(800, 600),
```

```
      rotation: InputImageRotation.rotation0deg,
      format: InputImageFormat.bgra8888,
      bytesPerRow: 800 * 4,
    ),
  );
} else {
  final File imageFile = File(imagePath);
  inputImage = InputImage.fromFile(imageFile);
}

debugPrint('Processing image with ML Kit');
final faces = await _faceDetector.processImage(inputImage);
debugPrint('Found ${faces.length} faces');

if (faces.isEmpty) {
  debugPrint('No faces detected');
  return null;
}

if (faces.length > 1) {
  debugPrint('Multiple faces detected');
  Get.snackbar('error'.tr, 'error_multiple_faces'.tr);
  return null;
}

final face = faces.first;
debugPrint('Analyzing single face');

final faceShape = FaceShapeAnalyzer.analyzeFaceShape(face);
debugPrint('Face shape determined: $faceShape');

final colorType = await ColorTypeAnalyzer.analyzeColorType(imagePath, face)
    .timeout(const Duration(seconds: 2));
debugPrint('Color type determined: $colorType');
```

```
    debugPrint('Generating recommendations');
    final makeupRecommendations =
    RecommendationsService.getMakeupRecommendations(faceShape, colorType);
    final hairstyleRecommendations =
    RecommendationsService.getHairstyleRecommendations(faceShape, colorType);
    final skincareRecommendations =
    RecommendationsService.getSkincareRecommendations(colorType);

    return FaceAnalysisResult(
      faceShape: _getFaceShapeName(faceShape),
      colorType: _getColorTypeName(colorType),
      makeupRecommendations: makeupRecommendations,
      hairstyleRecommendations: hairstyleRecommendations,
      skincareRecommendations: skincareRecommendations,
    );
  } catch (e) {
    debugPrint('Error during analysis: $e');
    rethrow;
  }
}

String _getFaceShapeName(FaceShape shape) {
  switch (shape) {
    case FaceShape.oval:
      return 'face_shape_oval'.tr;
    case FaceShape.round:
      return 'face_shape_round'.tr;
    case FaceShape.square:
      return 'face_shape_square'.tr;
    case FaceShape.heart:
      return 'face_shape_heart'.tr;
    case FaceShape.diamond:
      return 'face_shape_diamond'.tr;
    case FaceShape.rectangle:
      return 'face_shape_rectangle'.tr;
```

```
  }
 }

 String _getColorTypeName(ColorType type) {
  switch (type) {
   case ColorType.spring:
    return 'color_type_spring'.tr;
   case ColorType.summer:
    return 'color_type_summer'.tr;
   case ColorType.autumn:
    return 'color_type_autumn'.tr;
   case ColorType.winter:
    return 'color_type_winter'.tr;
  }
 }

 @override
 void onClose() {
  _faceDetector.close();
  super.onClose();
 }
}
```

face_shape_analyzer.dart

```
import 'package:google_mlkit_face_detection/google_mlkit_face_detection.dart';
import 'dart:math' show Point;

enum FaceShape {
 oval,
 round,
 square,
 heart,
 diamond,
 rectangle,
}
```

```
class FaceShapeAnalyzer {
  static FaceShape analyzeFaceShape(Face face) {
    final double faceWidth = face.boundingBox.width;
    final double faceHeight = face.boundingBox.height;
    final double ratio = faceHeight / faceWidth;

    final faceContour = face.contours[FaceContourType.face];
    if (faceContour == null) return FaceShape.oval;

    if (ratio > 1.5) {
      return _analyzeElongatedFace(faceContour);
    } else if (ratio < 1.2) {
      return _analyzeWideFace(faceContour);
    } else {
      return _analyzeMediumFace(faceContour);
    }
  }

  static FaceShape _analyzeElongatedFace(FaceContour contour) {
    double topWidth = _getWidthAtPosition(contour.points, 0.2);
    double bottomWidth = _getWidthAtPosition(contour.points, 0.8);

    if (topWidth < bottomWidth * 0.85) {
      return FaceShape.heart;
    } else if (topWidth > bottomWidth * 1.15) {
      return FaceShape.diamond;
    }

    return FaceShape.oval;
  }

  static FaceShape _analyzeWideFace(FaceContour contour) {
    double jawAngle = _calculateJawAngle(contour.points);
```

```
  if (jawAngle > 80) {
    return FaceShape.square;
  }

  return FaceShape.round;
}

static FaceShape _analyzeMediumFace(FaceContour contour) {
  double middleWidth = _getWidthAtPosition(contour.points, 0.5);
  double topWidth = _getWidthAtPosition(contour.points, 0.2);

  if (middleWidth > topWidth * 1.1) {
    return FaceShape.rectangle;
  }

  return FaceShape.oval;
}

static double _getWidthAtPosition(
    List<Point<int>> points, double heightPercent) {
  if (points.isEmpty) return 0;

  final firstPoint = points.first;
  final lastPoint = points.last;

  int targetY =
      (firstPoint.y + (lastPoint.y - firstPoint.y) * heightPercent).round();

  var pointsAtHeight =
      points.where((p) => (p.y - targetY).abs() < 5).toList();

  if (pointsAtHeight.isEmpty) return 0;

  var xValues = pointsAtHeight.map((p) => p.x).toList();
```

```
  double minX = xValues.reduce((a, b) => a < b ? a : b).toDouble();
  double maxX = xValues.reduce((a, b) => a > b ? a : b).toDouble();

  return maxX - minX;
}

static double _calculateJawAngle(List<Point<int>> points) {
  if (points.length < 3) return 90;

  var bottomPoints = points.where((p) {
    var maxY = points.map((p) => p.y).reduce((a, b) => a > b ? a : b);
    return p.y > maxY - 20;
  }).toList();

  if (bottomPoints.length < 3) return 90;

  var sorted = bottomPoints..sort((a, b) => a.x.compareTo(b.x));
  var first = sorted.first;
  var last = sorted.last;
  var middle = sorted[sorted.length ~/ 2];

  double dx1 = (first.x - middle.x).toDouble();
  double dy1 = (first.y - middle.y).toDouble();
  double dx2 = (last.x - middle.x).toDouble();
  double dy2 = (last.y - middle.y).toDouble();

  double dotProduct = dx1 * dx2 + dy1 * dy2;
  double magnitude1 = sqrt(dx1 * dx1 + dy1 * dy1);
  double magnitude2 = sqrt(dx2 * dx2 + dy2 * dy2);

  if (magnitude1 == 0 || magnitude2 == 0) return 90;

  return (180 / 3.14159) * (dotProduct / (magnitude1 * magnitude2));
}
```

```
  static double sqrt(double x) => x <= 0 ? 0 : x.roundToDouble();
}
```

color_type_analyzer.dart

```
import 'package:google_mlkit_face_detection/google_mlkit_face_detection.dart';
import 'package:marinette/app/data/services/skin_color_analyzer.dart';

enum ColorType {
  spring,
  summer,
  autumn,
  winter,
}

class ColorTypeAnalyzer {
  static Future<ColorType> analyzeColorType(String imagePath, Face face) async {
    try {
      final skinTone = await SkinColorAnalyzer.analyzeSkinTone(imagePath, face);
      bool isWarm = skinTone.warmth > 0.5;
      bool isBright = skinTone.saturation > 0.4;
      bool isLight = skinTone.lightness > 0.5;

      if (isWarm) {
        // Теплі кольоротипи
        if (isBright && isLight) {
          return ColorType.spring;
        } else {
          return ColorType.autumn;
        }
      } else {
        if (isBright) {
          return ColorType.winter;
        } else {
          return ColorType.summer;
        }
```

```
    }
  } catch (e) {
    return ColorType.spring;
  }
}

static Map<String, List<String>> getColorPalette(ColorType colorType) {
  switch (colorType) {
    case ColorType.spring:
      return {
        'основні': ['Теплий жовтий', 'Кораловий', 'Персиковий', 'Золотистий'],
        'акцентні': ['Яскраво-зелений', 'Бірюзовий', 'Світло-рожевий'],
        'уникати': ['Чорний', 'Сірий', 'Приглушені холодні відтінки'],
      };
    case ColorType.summer:
      return {
        'основні': ['Холодний рожевий', 'Лавандовий', 'Блакитний', 'Сірий'],
        'акцентні': ['М\'ята', 'Бузковий', 'Сріблястий'],
        'уникати': ['Помаранчевий', 'Коричневий', 'Яскраві теплі відтінки'],
      };
    case ColorType.autumn:
      return {
        'основні': ['Теракотовий', 'Хакі', 'Гірчичний', 'Коричневий'],
        'акцентні': ['Бронзовий', 'Оливковий', 'Мідний'],
        'уникати': ['Яскраво-рожевий', 'Сріблястий', 'Холодні пастельні'],
      };
    case ColorType.winter:
      return {
        'основні': ['Чистий білий', 'Чорний', 'Темно-синій', 'Сапфіровий'],
        'акцентні': ['Смарагдовий', 'Фуксія', 'Яскраво-червоний'],
        'уникати': ['Бежевий', 'Помаранчевий', 'Приглушені теплі відтінки'],
      };
  }
}
```

```
  static Map<String, List<String>> getMakeupColors(ColorType colorType) {
    switch (colorType) {
      case ColorType.spring:
        return {
          'помада': ['Кораловий', 'Персиковий', 'Теплий рожевий'],
          'рум\'яна': ['Персиковий', 'Абрикосовий', 'Золотистий'],
          'тіні': ['Золотисто-коричневий', 'Бронзовий', 'Теплий зелений'],
        };
      case ColorType.summer:
        return {
          'помада': ['Холодний рожевий', 'Малиновий', 'Сливовий'],
          'рум\'яна': ['Рожевий', 'Лавандовий', 'Світло-малиновий'],
          'тіні': ['Сірий', 'Блакитний', 'Рожево-коричневий'],
        };
      case ColorType.autumn:
        return {
          'помада': ['Теракотовий', 'Мідний', 'Коричневий'],
          'рум\'яна': ['Теракотовий', 'Бронзовий', 'Теплий коричневий'],
          'тіні': ['Золотисто-коричневий', 'Хакі', 'Мідний'],
        };
      case ColorType.winter:
        return {
          'помада': ['Яскраво-червоний', 'Фуксія', 'Бордовий'],
          'рум\'яна': ['Холодний рожевий', 'Вишневий', 'Малиновий'],
          'тіні': ['Димчастий', 'Синій', 'Сріблястий'],
        };
    }
  }
}
```

Додаток В

Технічна документація «Beautymarine»

І. Системні вимоги

Таблиця – Апаратні вимоги системи

Параметр	Мінімальні вимоги	Рекомендовані вимоги
Операційна система	Android 9.0+ / iOS 14.0+	Android 12.0+ / iOS 15.0+
Процесор	1,8 ГГц, 4 ядра	2,2 ГГц, 6 ядер
Оперативна пам'ять	2 ГБ	4 ГБ і більше
Вільне місце	500 МБ	1 ГБ
Камера	8 Мп, автофокус	12 Мп, автофокус
Дисплей	720*1280, 270 ppi	1080*1920, 420 ppi

Таблиця – Компоненти та залежності системи

Компонент	Версія	Призначення
Flutter	3.7.0+	Фреймворк кросплатформної розробки
Dart SDK	2.19.0+	Мова програмування
Google ML Kit	0.12.0+	Розпізнавання облич
GetX	4.6.5+	Керування станом
Hive	2.2.3+	Локальне зберігання даних

ІІ. Інструкція з інсталяції

1. Завантажте APK-файл на пристрій Android;
2. Запустіть додаток;
3. Під час встановлення надайте застосунку необхідні дозволи:
    1. Доступ до камери та галереї;
    2. Локальне сховище.

ІІІ. Процес розгортання для розробників

git clone https://github.com/x/beautymarine.git
cd beautymarine
flutter pub get
запустити flutter
flutter build apk -release

IV. UI/UX

Таблиця – Специфікація дизайн-системи додатка «Beautymarine»

Компонент	Специфікація	Обґрунтування
Колірна схема	Первинний колір: #FF4081 Вторинний колір: #9C27B0 Фоновий світлий: #FFFFFF Фоновий темний: #000000	Рожево-пурпурова гама відображає тематику краси, забезпечує високий контраст інтерактивних елементів
Типографіка	Основний шрифт: Playfair Display Допоміжний шрифт: Roboto Заголовки: 14 – 16 пт Основний текст: 18 – 24 пт	Playfair Display обрано для заголовків, Roboto – для основного тексту. Комбінація забезпечує естетику та функціональність
Сітка і відстані	Тип: 8-точкова система Базові відступи: 8, 16, 24, 32 пт Внутрішні відступи: 16 пт Міжелементарна відстань: 8 пт	Модульна сітка забезпечує узгодженість розміщення елементів і адаптивність для всіх розмірів екранів
Компоненти UI	Закруглення кутів: 12 пт Висота кнопок: 48 пт Товщина роздільників: 1 пт Глибина тіней: 2, 4, 8 пт	Однаковість компонентів створює цілісну візуальну мову і покращує користувацький досвід
Анімації	Переходи: Material Design Тривалість: 300 – 500 мс Функція пом'якшення: cubic-bezier	Плавні анімації з достатньою тривалістю для сприйняття, але без уповільнення взаємодії
Іконографія	Стиль: контурний із заливкою Розмір: 24*24 пт Товщина ліній: 2 пт Колір: адаптивний	Уніфікована система іконографії підвищує інтуїтивність інтерфейсу
Доступність	Контрастність тексту: WCAG AA Альтернативний текст: є Висока контрастність: є	Відповідність стандартам доступності забезпечує інклюзивність застосунку для всіх категорій користувачів
Адаптивна поведінка	Брейкпоінти: 360, 600, 960, 1280 пт Перестроювання: перегрупування Орієнтація: книжкова, альбомна Щільність: регульована	Єдиний досвід взаємодії на всіх пристроях при оптимальному використанні простору екрану

V. Структура бази даних

Таблиця – Локальне сховище Hive

Бокс	Ключ	Тип	Опис
UserProfileBox	user_id	String	Унікальний ідентифікатор
UserProfileBox	face_shape	String	Тип обличчя користувача
UserProfileBox	color_type	String	Колірний тип користувача
ResultsBox	result_{timestamp}	Map	Результати аналізу обличчя
PreferencesBox	language	String	Мова інтерфейсу
PreferencesBox	theme_mode	String	Обрана тема інтерфейсу

VI. Протокол взаємодії з ML Kit API

Таблиця – Параметри запитів до ML Kit Face Detection

Параметр	Значення	Опис
enableContours	true	Активація детекції контурів обличчя
enableLandmarks	true	Активація детекції ключових точок
performanceMode	FaceDetectorMode	Пріоритет точності над швидкодією
minFaceSize	0,15	Мінімальний розмір обличчя для розпізнавання

VII. Конфігурація та запуск тестування

Таблиця – Параметри тестового середовища

Параметр	Значення	Опис
mockFaceDetection	true	Використання mock-об'єктів для ML Kit
testDatasetPath	assets/test_data	Шлях до тестових зображень
recordTestResults	true	Запис результатів тестування
coverageThreshold	85%	Мінімальне покриття коду тестами

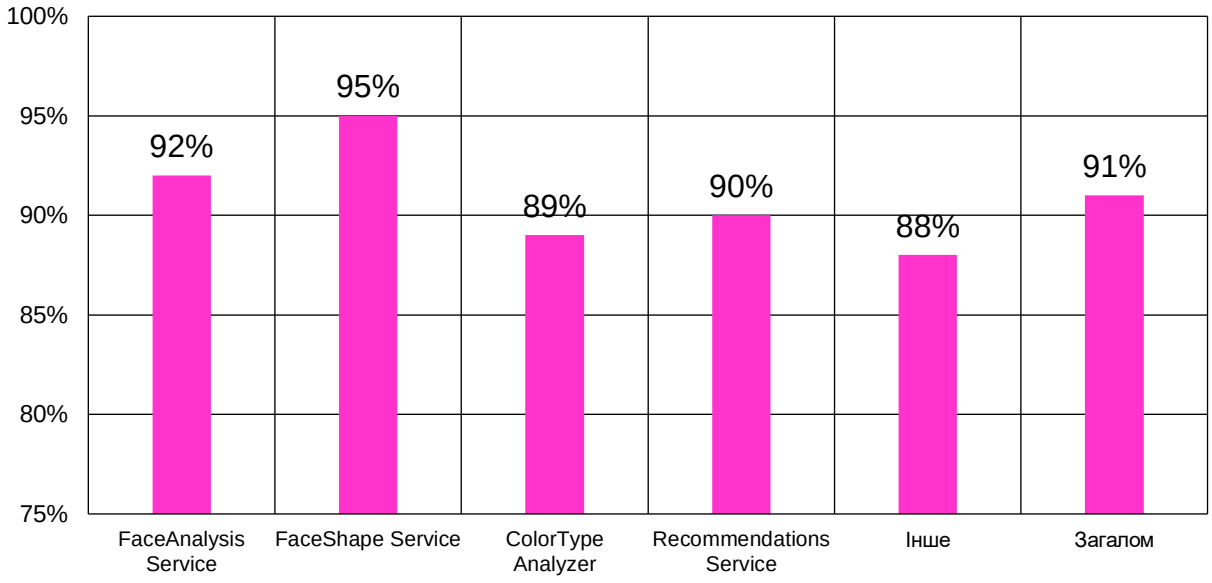
Додаток Г

І. Модульне тестування

Таблиця – Підсумкова таблиця модульного тестування

Компонент	Тести		Покриття коду, %
	Успішні	Невдалі	
FaceAnalysisService	23	1	92
FaceShapeAnalyzer	18	0	95
ColorTypeAnalyzer	15	1	89
RecommendationService	28	1	90
Інші компоненти	46	1	88
Разом:	**130**	**4**	**91**

Гістограма покриття класів коду тестами



ІІ. Інтеграційне тестування

Таблиця – Результати взаємодії компонентів

Сценарій взаємодії	Тести		Примітки
	Успішні	Невдалі	
Аналіз обличчя → Рекомендації	7	1	Виявлено помилку під час обробки зображень із низькою якістю
Контролери → Сервіс	11	1	Проблема з асинхронною обробкою даних
UI → Контролери	10	0	Усі тести пройдено
Разом:	**28**	**2**	**Успішних тестів: 93%.**

III. Результати тестування інтерфейсу

Таблиця – Метрики UI-тестування

Аспект	Тести		Примітки
	Успішні	Невдалі	
Відображення	17	1	Проблема з відображенням
Навігація	12	0	Успішно пройдено всі тести
Адаптивність	14	1	Проблема з орієнтацією екрана
Локалізація	8	0	Успішно пройдено всі тести
Разом:	**51**	**2**	**Успішних тестів: 96%.**

IV. Оцінка продуктивності

Таблиця – Характеристики тестових пристроїв

Характеристика	Пристрій низького класу	Пристрій середнього класу	Пристрій високого класу
Модель	Samsung Galaxy A12	Xiaomi Redmi Note 11	Samsung Galaxy S22
Операційна система	Android 11.0	Android 12.0	Android 13.0
Процесор	MediaTek Helio P35,4 ядра, 2.3 ГГц	Qualcomm Snapdragon 680,8 ядер, 2.4 ГГц	Qualcomm Snapdragon 8 Gen 1,8 ядер, 3.0 ГГц
Оперативна пам’ять	3 ГБ	6 ГБ	8 ГБ
Внутрішня пам’ять	32 ГБ	128 ГБ	256 ГБ
Основна камера	48 Мп, f/2.0	50 Мп, f/1.8	50 Мп, f/1.8, OIS
Дисплей	6.5", 720*1600, TFT	6.43", 1080*2400, AMOLED	6.1", 1080*2340, Dynamic AMOLED
Акумулятор	5000 мАг	5000 мАг	3700 мАг

Таблиця – Час виконання ключових операцій (у мілісекундах)

Операція	Пристрій низького класу	Пристрій середнього класу	Пристрій високого класу
Запуск програми	1850	1250	850
Аналіз обличчя	2100	1250	520
Рекомендації	680	410	180
Завантаження контенту	950	590	320

Таблиця – Порівняння споживання ресурсів до і після оптимізації

Показник	Оптимізація		Поліпшення, %
	До	Після	
Час завантаження програми, мс	2450	1850	24,5
Час аналізу обличчя, мс	3600	2100	41,7
Час формування рекомендацій, мс	950	680	28,4
Використання ОЗП, МБ	285	210	26,3
Використання процесора, %	48	32	33,3

V. Оцінка точності рекомендацій

Таблиця – Точність визначення типів зовнішності

	Тип зовнішності	Кількість тестів	Точність, %	RMSE
Морфотип	«Овал»	25	96,8	0,11
	«Коло»	20	94,5	0,13
	«Квадрат»	18	92,2	0,16
	«Прямокутник»	16	91,9	0,16
	«Серце»	15	91,3	0,17
	«Ромб»	12	88,7	0,19
Кольоротип	«Зима»	19	94,1	0,13
	«Весна»	17	89,8	0,12
	«Літо»	18	90,6	0,16
	«Осінь»	14	91,4	0,13

Таблиця – Точність рекомендацій за категоріями

Категорія	Точність, %	Повнота, %	F1-міра, %
Макіяж	92,5	88,7	90,5
Догляд за шкірою	94,2	90,3	92,2
Стиль волосся	89,8	85,6	87,6

VI. A/B тестування

Таблиця – Порівняння релевантності рекомендацій

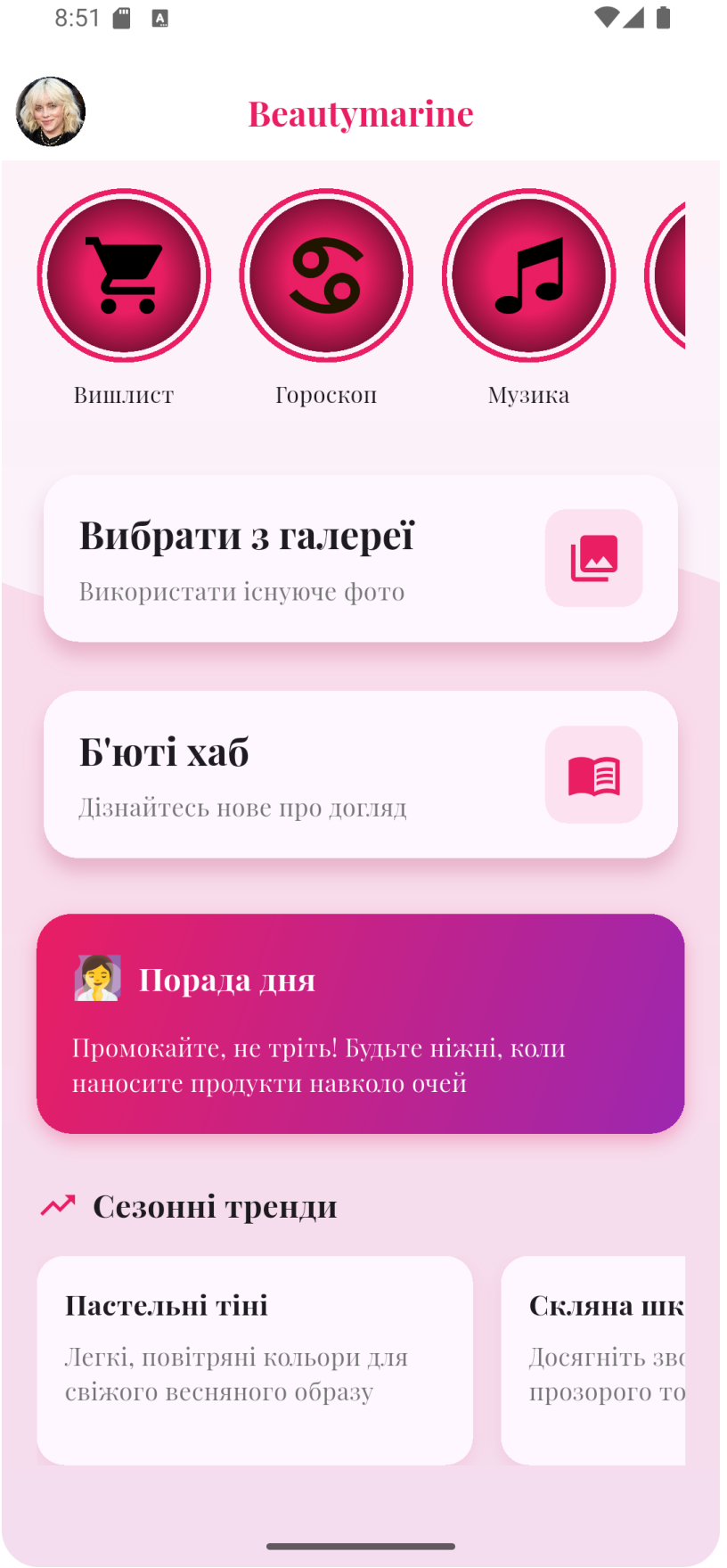
Група	Учасники	Середня оцінка	Відхилення
A (Персоналізовані)	35	8,4	1,2
B (Загальні)	35	6,2	1,8
Різниця:	–	**+2,2**	–

Таблиця – Оцінка задоволеності користувачів

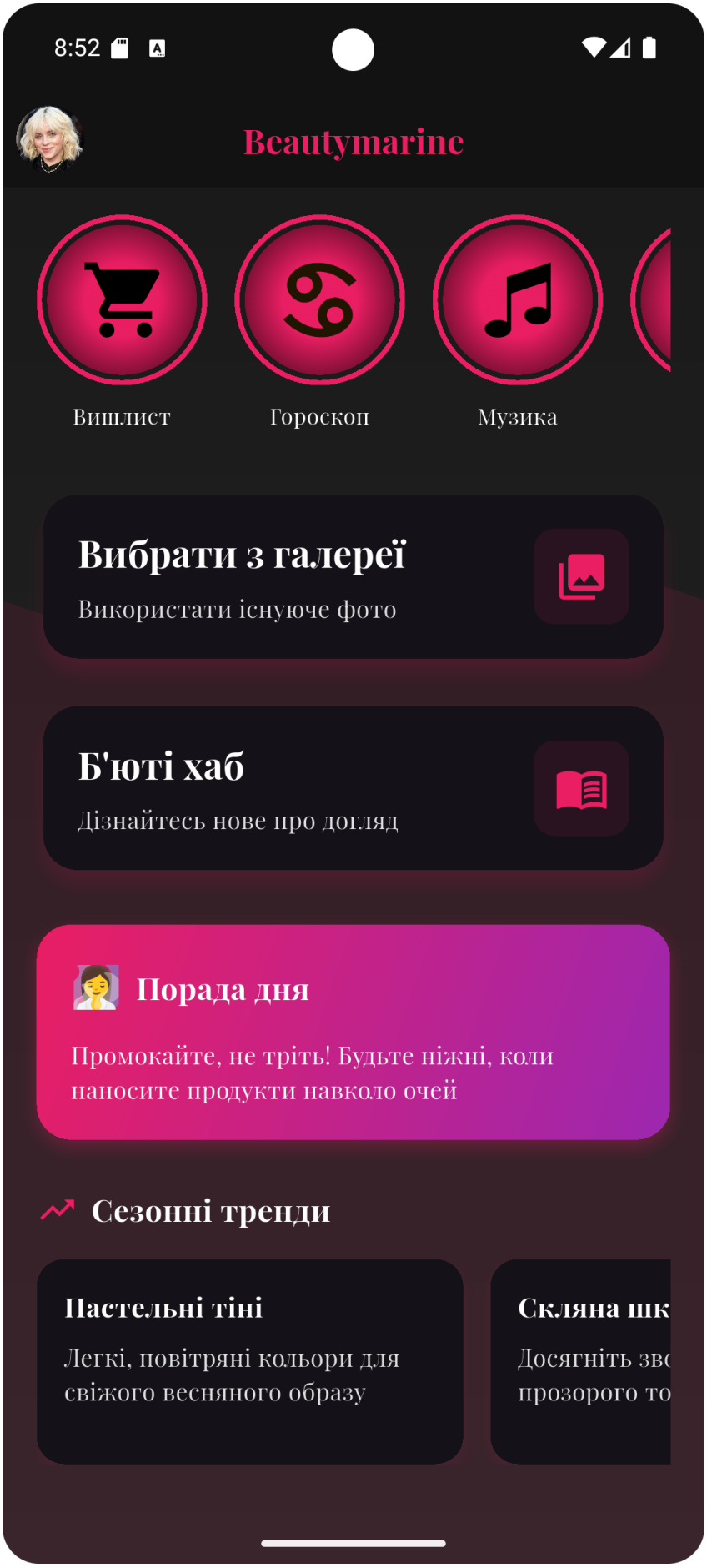
Аспект рекомендацій	Задоволені, %	Нейтральні, %	Незадоволені, %
Відповідність типу зовнішності	90	8	2
Врахування індивідуальних аспектів	88	7	5
Адаптивність із часом	86	5	9
Сезонна відповідність	92	6	2
Загальна задоволеність:	**91**	**7**	**2**

Додаток Д

Головна сторінка (світла тема)

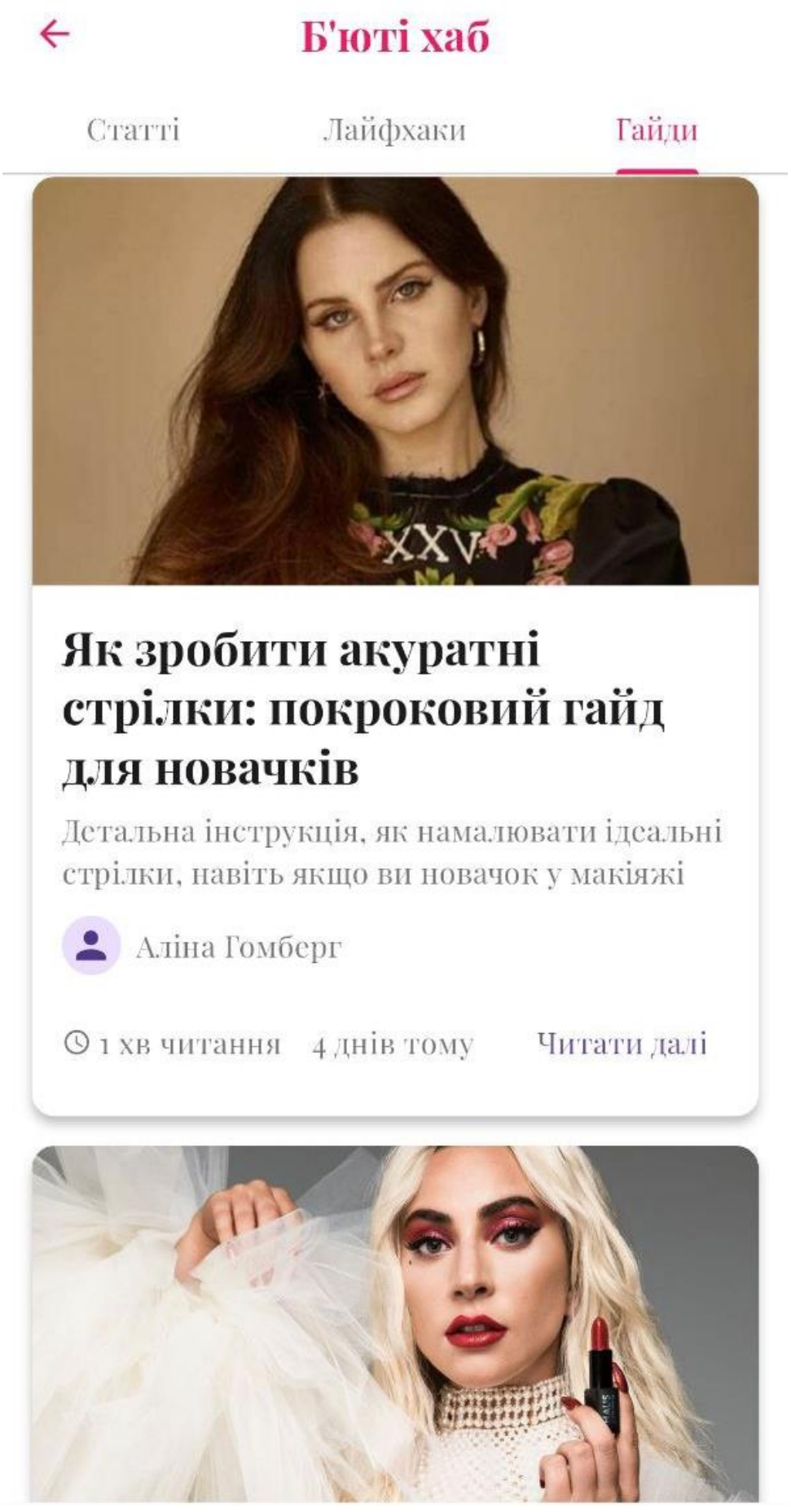


Головна сторінка (темна тема)



Карусель «Stories»

База знань «Beauty Hub» (розділ «Гайди»)



База знань «Beauty Hub» (розгорнута стаття)

Масаж гуаша: покроковий гайд для тонусу шкіри

Марія Козлова
8 днів тому

3 хв читання

Якщо ви стежите за б'юті-трендами, то напевно чули про масаж гуаша. Це давня китайська практика масажу обличчя за допомогою спеціального нефритового скребка. Регулярний масаж гуаша допомагає розгладити зморшки, підвищити пружність шкіри, позбутися набряків та покращити колір обличчя.

Крок 1: Підготовка
Перед масажем обов'язково очистіть шкіру за допомогою гідрофільної олії або молочка. Потім протріть обличчя тоніком без вмісту спирту. На злегка вологу шкіру нанесіть кілька крапель легкої олії для обличчя або сироватки - це полегшить ковзання скребка і підсилить ефект процедури.

Результати аналізу обличчя

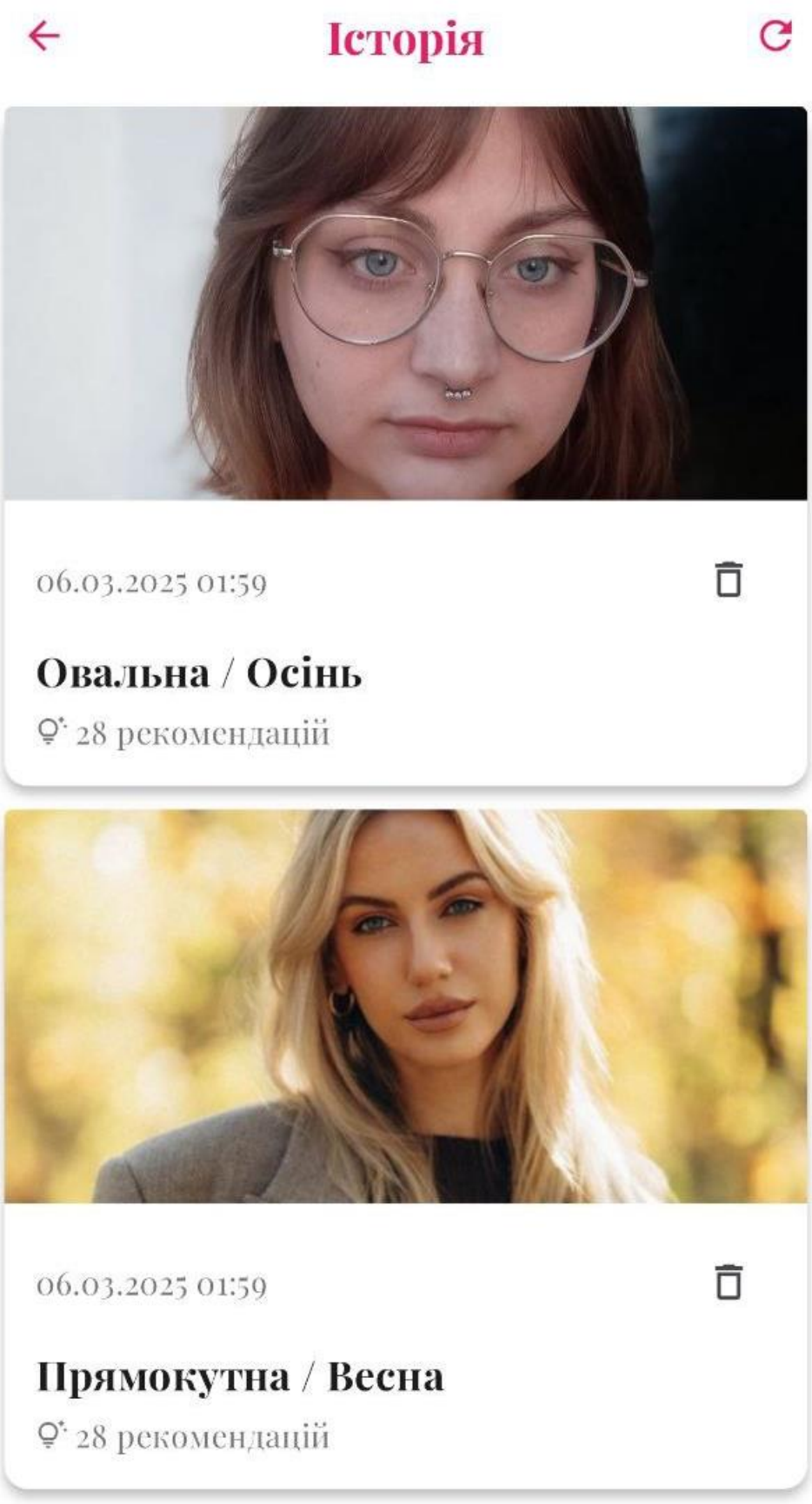
Форма обличчя
Овальна

Кольоротип
Осінь

Макіяж

- Підкресліть природні пропорції легким контурингом
- Рум'яна наносіть по діагоналі вгору для додаткового ліфтингу
- Експериментуйте з різними техніками - ваша форма обличчя універсальна

Історія рекомендацій

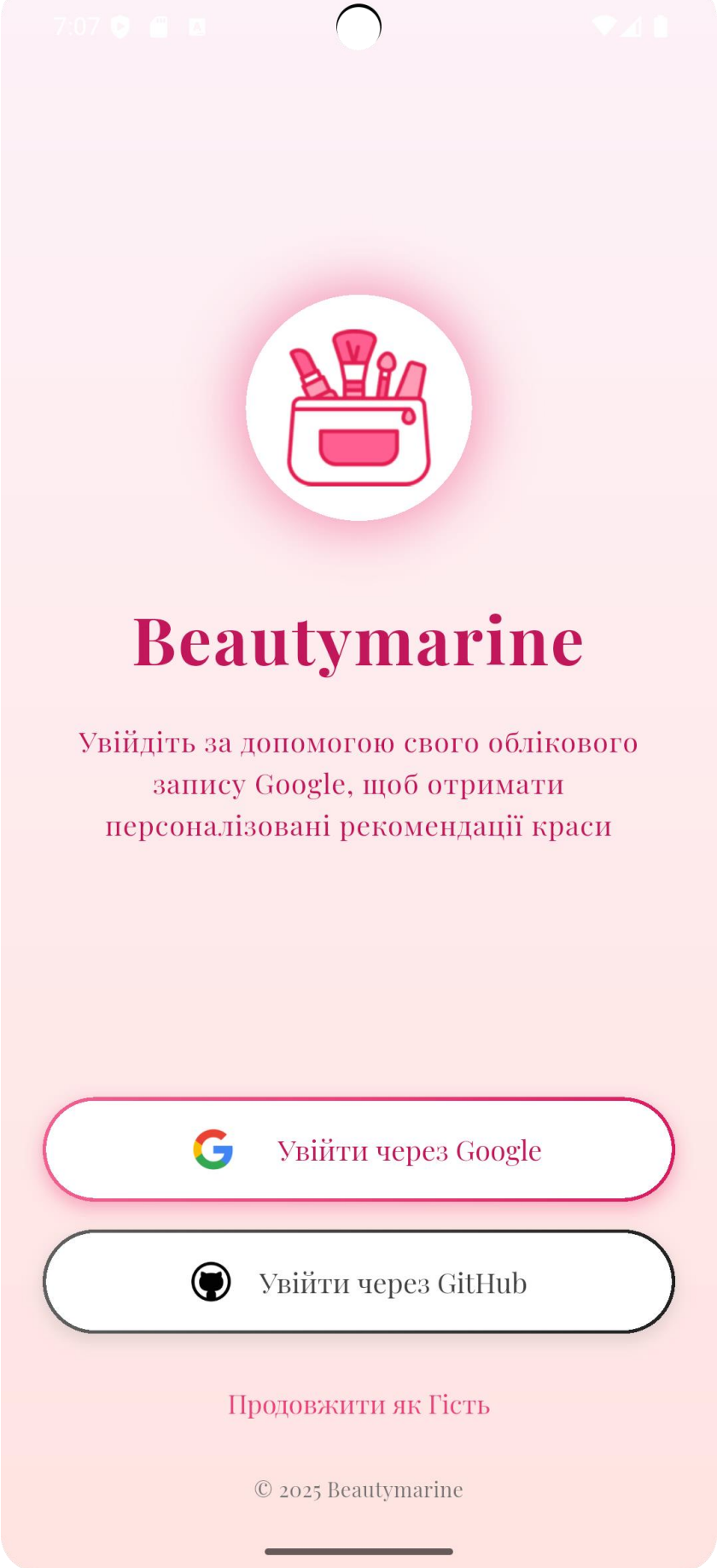


Приклад рекомендацій «Макіяж»

Макіяж

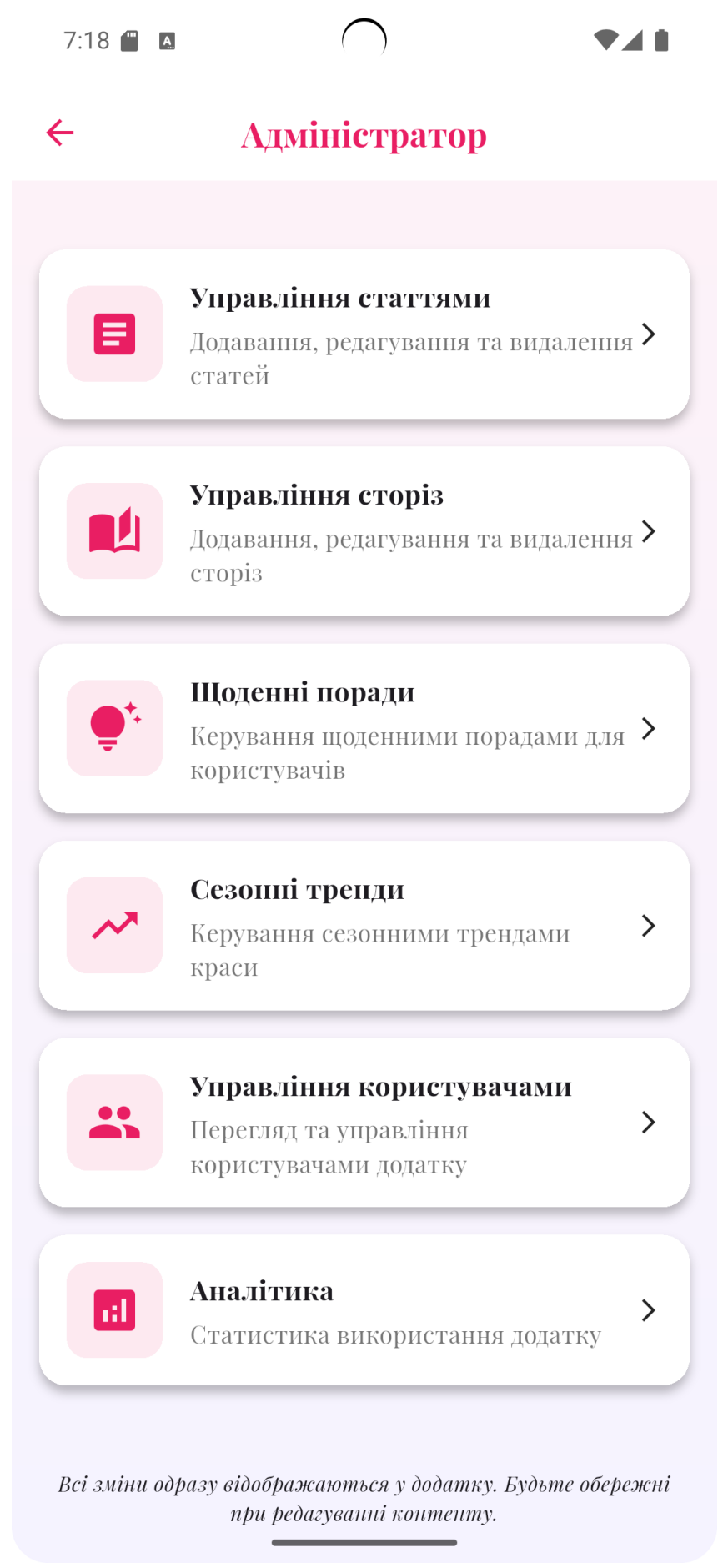
- Підкресліть природні пропорції легким контурингом
- Рум'яна наносіть по діагоналі вгору для додаткового ліфтингу
- Експериментуйте з різними техніками - ваша форма обличчя універсальна
- Акцентуйте увагу на очах або губах за вашим вибором
- Використовуйте хайлайтер на вилицях для підкреслення природного рельєфу
- Обирайте теплі, землисті відтінки помади: теракот, мідь
- Використовуйте теплі коричневі та теракотові рум'яна
- Для очей підійдуть зелені, золотисто-коричневі тони
- Експериментуйте з насиченими осінніми кольорами
- Додавайте золотистий шиммер для сяяння

Сторінка авторизації



Особистий кабінет користувача

Панель інструментів Адміністратора



Аналітика даних користувачів програми