

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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка web-застосунку для формування персоналізованих
стилістичних образів»

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

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

(підпис) Вероніка КАЗАК

Виконав: здобувачка вищої освіти групи ПД-44

Вероніка КАЗАК

Керівник: _____
Олександр ЯРОШЕВСЬКИЙ
асист.

Рецензент: _____

Київ 2026

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

« ____ » _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

_____ Казак Вероніці Юріївні

1. Тема кваліфікаційної роботи: «Розробка web-застосунку для формування персоналізованих стилістичних образів»
керівник кваліфікаційної роботи асистент кафедри Олександр ЯРОШЕВСЬКИЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.
2. Строк подання кваліфікаційної роботи «29» травня 2026 року.
3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи підбору одягу для створення персоналізованого образу, опис процесу навчання згорткової нейронної мережі, технічна документація з описом бібліотек для створення веб-застосунків та роботи з моделями розпізнавання зображень.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
 1. Огляд та аналіз існуючих методів та технологій розпізнавання зображень одягу в контексті задачі підбору персоналізованого стилістичного образу.
 2. Огляд та аналіз способів та принципів створення образів для автоматизації процесу формування персоналізованих образів.

3. Вибір засобів програмної реалізації веб-застосунку та засобів створення моделі розпізнавання зображень одягу.
4. Створення моделі розпізнавання зображень одягу для використання в веб-застосунку.
5. Проектування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».
6. Програмна реалізація та тестування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма діяльності підбору образу.
6. Діаграма класів бази даних.
7. Діаграма структури сторінок.
8. Екранні форми.
9. Апробація результатів дослідження.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	
3	Огляд та аналіз існуючих методів та технологій розпізнавання зображень одягу в контексті задачі підбору персоналізованого стилістичного образу	08.03-15.03.2026	
4	Огляд та аналіз способів та принципів створення образів для автоматизації процесу формування персоналізованих образів.	16.03-23.03.2026	
5	Вибір засобів програмної реалізації веб-застосунку та засобів створення моделі розпізнавання зображень одягу.	24.03-29.03.2026	
6	Створення моделі розпізнавання зображень одягу для використання в веб-застосунку.	30.03-09.04.2026	
7	Проектування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».	10.04-17.04.2026	
8	Програмна реалізація та тестування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».	18.04-05.05.2026	

9	Оформлення роботи: вступ, висновки, реферат	01.05-11.05.2026	
10	Розробка демонстраційних матеріалів	12.05-18.05.2026	
11	Попередній захист роботи	11.05-22.05.2026	

Здобувач(ка) вищої освіти

(підпис)

Вероніка КАЗАК

Керівник
кваліфікаційної роботи

(підпис)

Олександр ЯРОШЕВСЬКИЙ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 1 табл., 20 рис., 20 джерел.

Мета роботи – спрощення процесу формування персоналізованих стилістичних образів за рахунок автоматизації підбору одягу.

Об'єкт дослідження – рекомендаційні алгоритми для формування персоналізованих стилістичних образів та робота з моделями розпізнавання зображень і згортковими нейронними мережами.

Предмет дослідження – технології розробки веб-додатків та алгоритми формування персоналізованих стилістичних образів.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для підбору елементів одягу в контексті задачі формування персоналізованого стилістичного образу. Проаналізовано способи роботи зі згортковими нейронними мережами та моделями розпізнавання зображень. Розроблено алгоритми підбору одягу за стилем і на основі одного базового елементу, програмно реалізовані ключові функціональні можливості, зокрема: завантаження користувачем фото, розпізнавання фотографій одягу, підбір образу, CRUD операції з одягом, збереження та перегляд стилістичних образів. В роботі використано бібліотеку Pytorch для навчання згорткової нейронної мережі та експорту її в формат ONNX моделі, платформи .NET та Razor Pages для створення інтерфейсу та функціоналу веб-застосунку.

Сферою використання застосунку є автоматизація процесу створення стилістичних образів для людей зацікавлених в моді.

КЛЮЧОВІ СЛОВА: МАШИННИЙ ЗІР, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ONNX, PYTHON, PYTORCH, .NET, RAZOR PAGES, ВЕБ-ЗАСТОСУНОК, МОДА, ПІДБІР ОБРАЗУ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	10
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ.....	13
1.1 Аналіз алгоритмів та методів підбору одягу	13
1.2 Аналіз існуючих програмних рішень	15
1.2.1 AI Closet.....	15
1.2.2 Closet OS	18
1.2.3 Aura.....	21
1.2.4 Результати аналізу	25
1.3 Визначення вимог до застосунку	25
2 ЗАСОБИ РЕАЛІЗАЦІЇ	28
2.1 Середовище розробки	28
2.1.1 Visual Studio	28
2.1.2 Google Collaboratory	29
2.2 Мови програмування	30
2.2.1 C#	30
2.2.2 Python.....	31
2.2.3 HTML та CSS.....	32
2.2.4 JavaScript.....	32
2.3 Фреймворки.....	33
2.3.1 Razor Pages	33
2.3.2 Identity	34
2.3.3 Entity Framework Core	35
2.3.4 ONNX Runtime	35
2.4 Бібліотеки	36
2.4.1 Pytorch.....	37
2.4.2 Pandas	37
2.4.3 OpenCV	38

2.5 Система контролю версій.....	39
2.6 Інструменти дизайну інтерфейсу.....	40
2.7 База даних та СУБД.....	40
3 ПРОЄКТУВАННЯ.....	42
3.1 Проєктування архітектури застосунку.....	42
3.2 Архітектура клієнтської частини.....	44
3.3 Архітектура серверної частини.....	47
4 РЕАЛІЗАЦІЯ.....	51
4.1 Створення моделі розпізнавання зображень.....	51
4.2 Реалізація серверної частини.....	53
4.3 Реалізація клієнтської частини.....	57
4.4 Інтеграція та взаємодія компонентів.....	59
ВИСНОВКИ.....	61
ПЕРЕЛІК ПОСИЛАНЬ.....	63
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ.....	66
ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ.....	74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JWT – JSON Web Token

JSON – JavaScript Object Notation

HTTP – Hypertext Transfer Protocol

ПЗ – Програмне забезпечення

IDE – Integrated Development Environment

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

SQL – Structured Query Language

MSSQL - Microsoft SQL

ER - Entity Relationship

ВСТУП

Актуальність: застосунки для створення стилістичних образів є не надто великою, але швидко зростаючою нішею. Створення додатку для створення персоналізованих стилістичних образів з використанням технологій розпізнавання зображень для автоматичного заповнення даних про елементи гардеробу та можливістю створити образ зі збереженого одягу в один клік може допомогти користувачам, які вже зацікавлені в моді, або хочуть спробувати стильно одягатися.

Створення сервісу для формування персоналізованих стилістичних образів саме у форматі веб-застосунку, який буде доступний як і на комп'ютерах та ноутбуках, так й на мобільних пристроях допоможе закрити прогалину у ринку заповненому в основному мобільними додатками, адже забезпечить доступ до сервісу на різних пристроях збільшуючи можливу кількість користувачів.

Об'єктом дослідження є рекомендаційні алгоритми для формування персоналізованих стилістичних образів та робота з моделями розпізнавання зображень та зготковим нейронними мережами.

Предметом дослідження є технології розробки веб-додатків та алгоритми формування персоналізованих стилістичних образів.

Метою роботи є спрощення процесу формування персоналізованих стилістичних образів за рахунок автоматизації підбору одягу.

Методи дослідження: першим етапом було проведення аналізу інших веб-застосунків для створення персоналізованих стилістичних образів та способи роботи з моделями розпізнавання зображень. Також було проаналізовано способи підбору одягу та виділено основні правила, які допомагають створити образи. З отриманих під час аналізу даних було створено вимоги до розроблюваного веб-застосунку.

Наступним етапом був вибір засобів технічної реалізації та визначення архітектури застосунку. Першим серед засобів реалізації було обрано спосіб роботи з моделлю розпізнаванням зображень, а саме використання ONNX моделі створеної на основі згорткової нейронної мережі. Далі для веб-застосунку було обрано клієнт-серверну архітектуру, яка є звичною для більшості веб-застосунків. Потім було обрано платформу та середовище розробки веб-застосунку, а саме .NET Razor Pages з використанням реляційних баз даних MSSQL та середовище розробки Visual Studio. Крім того було обрано систему контролю версій - GitHub та застосунок для дизайну інтерфейсу - Figma.

Наукова новизна роботи полягає в реалізації веб-застосунку з розпізнаванням і класифікацією зображень одягу та створення образів з цих елементів одягу за допомогою рекомендаційного алгоритму та системи правил. Особливістю є використання ONNX моделі для розпізнавання зображень одягу, значення від якої використовуються для заповнення інформації про елементи одягу.

Практична значущість результатів полягає у створенні веб-застосунку, який може бути використаний користувачами для створення персоналізованих стилістичних образів з елементів одягу завантажених у веб-застосунок, що допомагає користувачам створити гармонійні образи за короткий час.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз алгоритмів та методів підбору одягу

Існує багато різних способів підбору одягу, тому для створення веб-застосунку для формування персоналізованих стилістичних образів потрібно проаналізувати ці способи та перетворити їх в автоматизований алгоритм. Підбір одягу для створення певного образу має багато деталей, які впливають на те, яким вийде кінцевий образ, цими деталями є: основні та вторинні кольори, крій та матеріал одягу, сучасні модні тенденції тощо. Задля створення алгоритму для формування образів з урахуванням цих факторів спочатку потрібно обрати за якою саме системою буде працювати алгоритм. Для роботи було обрано рекомендаційний алгоритм на основі системи правил.

Серед інших можливих для використання алгоритмів та способів підбору образів є такі, як: підбір за допомогою ШІ, алгоритм на основі уподобань користувача, підбір на основі схожих користувачів. Кожен із цих підходів має власні переваги та недоліки, тому їх необхідно враховувати під час вибору архітектури системи.

Підбір образів за допомогою ШІ є одним з найсучасніших підходів у галузі формування стилістичних образів. здатні аналізувати фотографії одягу, визначати стиль, сезонність, типи елементів одягу та навіть прогнозувати відповідність речей між собою. Проте використання готових моделей штучного інтелекту або сторонніх сервісів має певні недоліки. В основі створюваного веб-застосунку є використання лише наявних у системи елементів одягу, а при використанні сторонніх ШІ ця вимога може не завжди виконуватися.

Окрім готового ШІ також є варіант використання машинного навчання. Подібні алгоритми дозволяють системі самостійно знаходити закономірності у поєднанні елементів одягу та покращувати результати рекомендацій з часом. Наприклад, модель може аналізувати успішні образи, оцінки користувачів або

частоту використання певних комбінацій. Недоліком такого підходу є те, що створення всіх правил та навчання може зайняти велику кількість часу, хоч і може мати доволі гарні результати [1].

Алгоритм на основі уподобань користувача передбачає підбір образів із врахуванням найбільш часто використовуваних елементів одягу. Такий підхід забезпечує високий рівень персоналізації, оскільки система поступово адаптується до смаків конкретного користувача. Однак для реалізації такого підходу необхідно створити механізм відстеження історії використання одягу, збереження статистики. Крім того, орієнтація лише на вже використані елементи одягу може зменшити різноманітність образів та обмежити рекомендації лише улюбленими речами користувача.

Схожим підходом є підбір на основі схожих користувачів. Цей спосіб потребує великої кількості користувачів та великої кількості збережених даних про їх поведінку. Також веб-застосунок повинен мати можливість створення образів вручну або відстеження частоти використання елементів одягу в образах, що додає ускладнює структуру створюваного веб-застосунку.

Для створення веб-застосунку для формування персоналізованих стилістичних образів було обрано саме рекомендаційний алгоритм на основі системи правил, адже він має не надто складну структуру та буде працювати незалежно від кількості користувачів та забезпечує різноманіття образів [2]. Мінусами даного підходу є те що всі правила необхідно писати вручну і це неможливо автоматизувати, що може збільшити час на реалізацію. Крім того, необхідне розуміння правил підбору образів та теорії кольору задля створення правил та оцінок за відповідність та невідповідність цим правилам.

У веб-застосунку «PerVi Closet» є різні правила, які загально можна поділити на: комбінації кольорів, комбінації типів одягу та вподобання стилів. Під комбінацією кольорів мається на увазі перевірка чи є два кольори частиною наявних комбінацій кольорів та оцінка комбінації.

У веб-застосунку використовуються основні гармонійні схеми з теорії кольорів, зокрема аналогові кольори, компліментарні кольори та тріадні

комбінації. Аналогові кольори є сусідніми на кольоровому крузі та створюють м'які й гармонійні образи. Компліментарні кольори розташовані навпроти один одного та створюють контрастні поєднання. Тріадні кольорові схеми складаються з трьох рівновіддалених кольорів на кольоровому крузі та дозволяють створювати яскраві, але збалансовані образи.

Серед комбінацій типів одягу є правила для покращення комбінування елементів, наприклад комбінація блузи та спортивних штанів оцінюється від'ємним значенням, адже ці елементи одягу зовсім не підходять один одному. Під вподобанням стилів мається на увазі та які елементи одягу підходять під стиль, а яких краще уникати, наприклад формальному стилю підходять сорочки, блузки, штани та нейтральні кольори, а уникати краще насичених кольорів та худі та спортивних штанів.

1.2 Аналіз існуючих програмних рішень

На ринку існує багато засобів для створення персоналізованих стилістичних образів, вони дозволяють користувачам зберігати елементи одягу у віртуальній шафі, створювати готові образи та в деяких випадках отримувати рекомендації щодо комбінування речей. Хоча подібні застосунки й поширені більшість із них орієнтовані лише на мобільні пристрої та не мають веб або десктопної версії. Серед існуючих програмних рішень, які є веб-додатками доступними на ПК та телефонах розглянемо три різних варіанти.

1.2.1 AI Closet

Першим є AI Closet, цей сервіс як і більшість аналогів має обов'язкову авторизацію користувача, після якої стають доступні усі функції.

Add Item



Add a new item to your closet.

Item Image



Item Details



Name

Colorful Athletic Sneakers

Description

Eye-catching sneakers in vibrant hues with a cushioned sole for

Category

Shoes



Add Item

Рис. 1.1 Додавання елементу одягу

При завантаженні зображень можна використати як локально збережені на пристрої файли так й посилання на інтернет ресурс. При завантаженні зображень відбувається їх аналіз за допомогою ШІ, це генерує назву, опис та категорію елементу. Підтримувані категорії: tops (футболки, блузи, сорочки тощо), bottoms (штани, джинси, спідниці), outerwear (верхній одяг), accessories (аксесуари), dresses (сукні), shoes (взуття).

Створення образів у додатку може відбуватися двома способами: самостійним створенням або запитом до ШІ. При самостійному створенні образу можна додавати будь-які кількість елементів, назву образу та його опис. Це непоганий варіант роботи, але через необмеженість образ може складатися як з одного елементу, наприклад лише чоботів, так і з всіх збережених елементів одягу. Хоча б деякі обмеження в мінімальній та максимальній кількості елементів, були б доречні.

Другий варіант - запит до ШІ, може видавати як теоретичні рекомендації до стилізації одягу, так і готові образи які можна швидко зберегти. Головним недоліком є обмеження користування ШІ, а саме п'ять запитів на день та дуже різна якість готових образів. Наприклад при запиті створити костюм для прогулянки було створено образ приведений нижче.

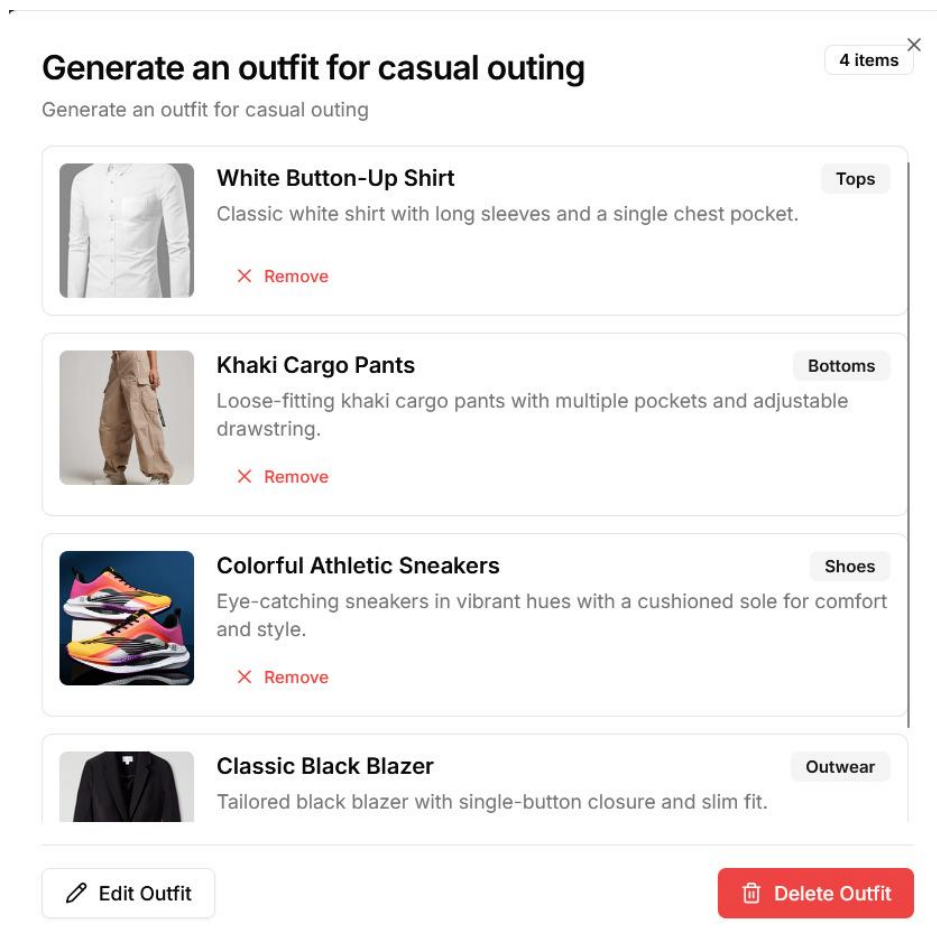


Рис. 1.2 Створений ШІ образ для прогулянки

Загалом додаток працює і виконує свої основні функції, але має деякі проблеми зі створенням гармонійних образів.

1.2.2 Closet OS

У цьому додатку також можна завантажувати як файли так і посилання на них, крім того можна обрати чи ви вже маєте додану річ чи хочете її придбати. Загалом при додаванні речей є назва, колір, бренд та категорія, але є один незручний нюанс, при додаванні одягу необхідно натиснути кнопку “analyze item” яка аналізує елемент одягу і заповнює всі описані вище поля, окрім бренду. Незручність в основному полягає у тому, що всі написані до натискання кнопки дані видаляються. Також ця кнопка видає фон зображення, що працює з більшістю елементів, але зовсім нічого не змінює або лише частково видає фон для інших. До того ж при створенні опису одягу не відображається його зображення та зміна назви кольору при створенні нічого не міняє його відображення.



Рис. 1.3 Додавання зображення одягу

Do you own this item?

☒ I own this ☐ I want this

Item Name

Women's Suede Ankle Boots

Color

Dark Olive Green

Brand

Brand name

Category

Shoes

Add Item

Рис. 1.4 Додавання опису елементу одягу

Для того, щоб повністю як назву та і кольорове відображення кольору одягу необхідно після створення зайти в деталі та вже там змінити назву кольору та вибрати його у вікні вибору кольору.

Колір

Основні кольори:

Додаткові кольори:

Визначити колір >>

OK Скасувати

Колір(Заливка)

Відтінок: 236 Червоний: 165

Контраст: 240 Зелений: 0

Яскравість: 78 Синій: 18

Додати до набору

Women's Red Single-breasted Coat

Cancel Save

Brand

Color

ТЕСТ

Material

Category

Outerwear

Status

I own this

Рис. 1.5 Зміна кольору одягу

Образи у цьому додатку як і у минулому можна створити як за допомогою ШІ так і самостійно. При самостійному створенні знову немає ніяких обмежень. При створенні користувача може додавати та перетягувати фотографій по власному бажанню на заданій ділянці. Є проблема з доступом до деяких категорій, адже немає прокрутки у фільтрі категорій, тому використати елементи з деяких категорій неможливо.

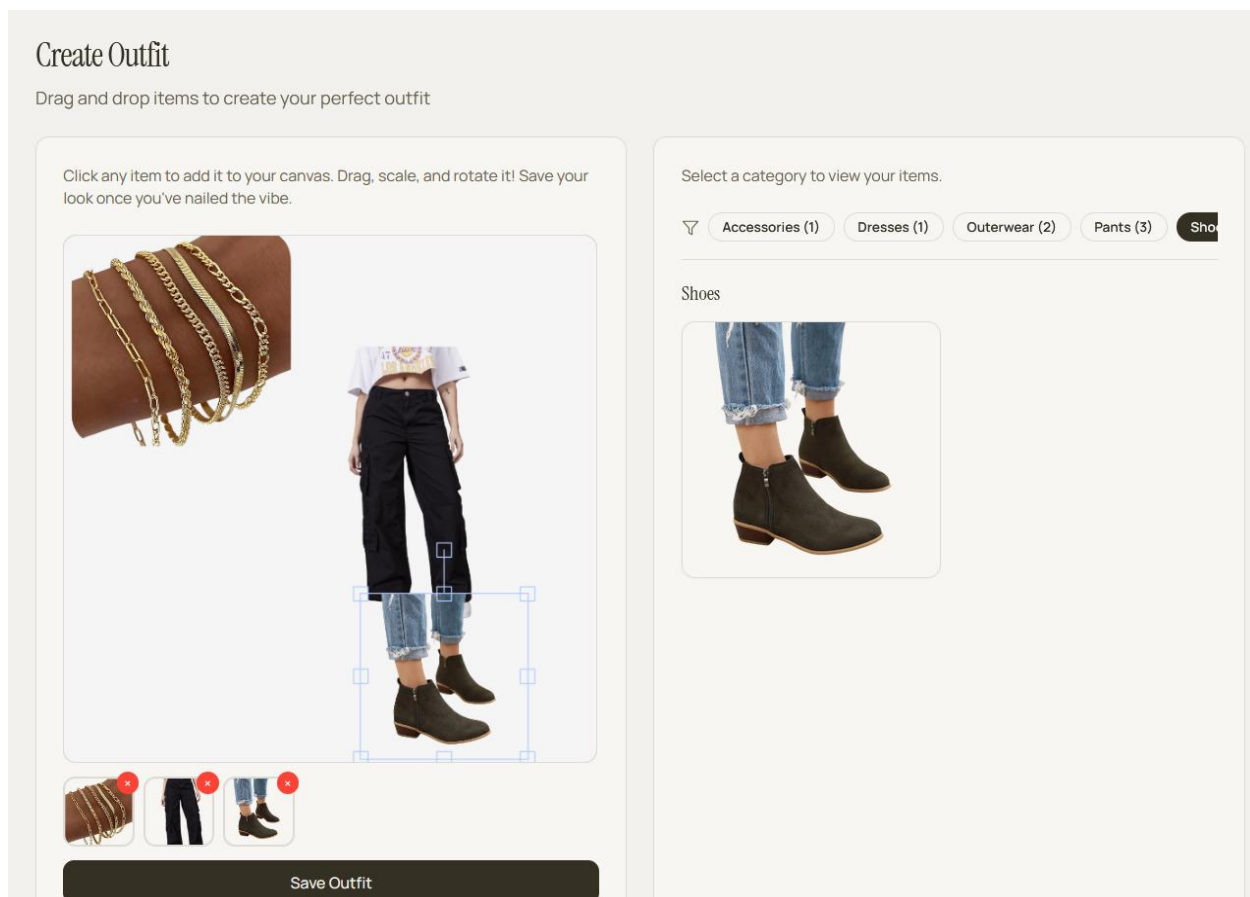


Рис. 1.6 Створення образу

При створенні образу за допомогою ШІ, користувачу дається опис та зменшені фото елементів одягу, які можна зберегти в бібліотеку образів. Образи загалом більш гармонійні, ніж у минулому сервісі, але все одно є невеликі проблеми. Наприклад при запиті про допомогу з створенням образу з використанням червоного пальто, яке є у списку одягу створений образ поперше не має цього пальто, а по-друге кольори інших елементів виглядають погано разом.

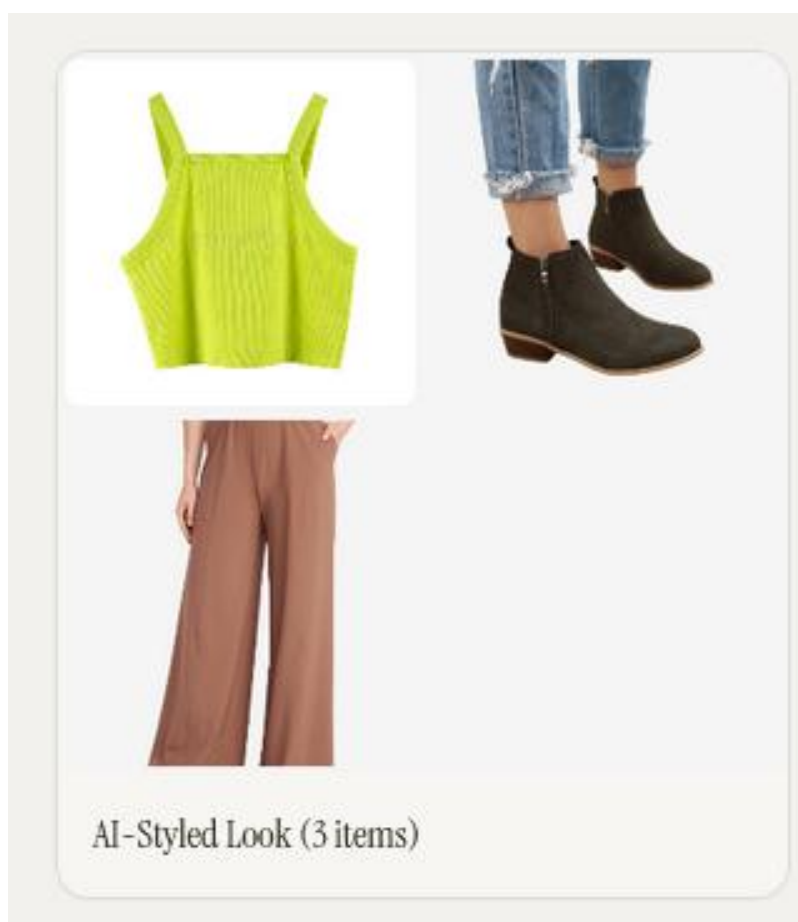


Рис. 1.7 Створений образ

1.2.3 Aura

У даному додатку одяг можна додати лише файлом, а заповнювані характеристики: назва, тип (те саме що категорія у інших), колір, бренд, сезон, випадки та нотатки. Бренд та назва заповнюються користувачем, але лише назва обов'язкова, що робить додавання одягу більш складним, адже кожному елементу одягу потрібно самотійно давати назву.

Item Photo



 **Add Photo**

Aura styles best when your pieces are laid out clearly on a white background 💡
Accepts HEIC, JPEG, PNG. Auto-converts and compresses.

Item Name *

e.g. Blue silk blouse

Type *

Shoes

Color

vibrant pink, orange, yellow, and purple

Brand

e.g. Zara

Season

All Seasons

Occasions

☐ Casual
 ☐ Work
☐ Date
 ☐ Workout
☐ Party
 ☐ Beach

Notes

Style detected: sporty.

Рис. 1.8 Додавання одягу

Для створення образів є декілька варіантів: III стиліст, візуалізація образу (Mix & Match) та ідеї стилізації (Rewear Ideas). Спочатку про III стиліста, він дає загальні поради та ідеї, але навіть при прямому запиті створити образ з того що є у бібліотеці він відповідає лише теоретично та лише словами. Крім того іноді він може загалом не відповідати.

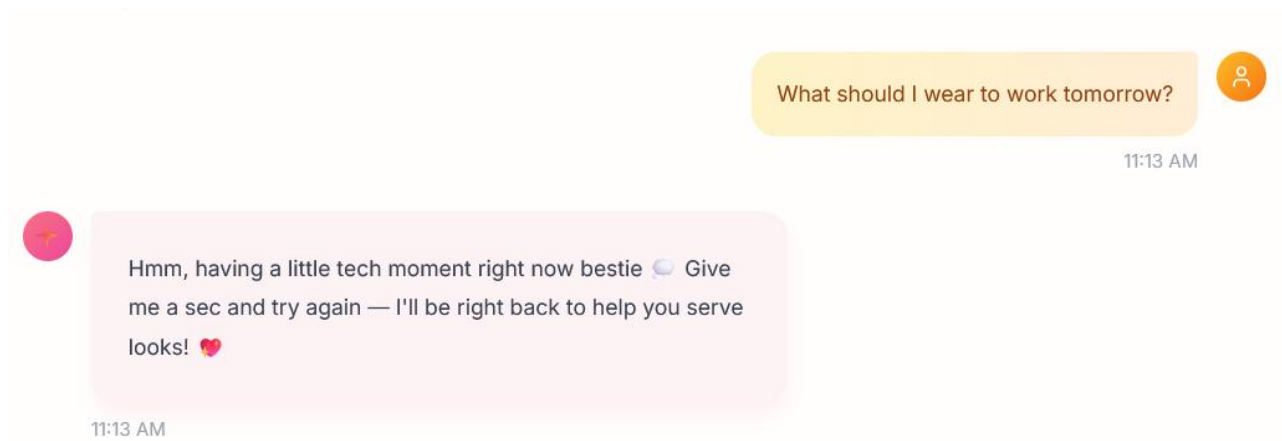


Рис. 1.9 Відсутня відповідь ІІІ стиліста

Візуалізація образу (Mix & Match) рекомендує що можна підібрати для обраного елементу одягу та робить колаж з цих елементів. Ця функція працює, але список елементів однаковий для майже всіх елементів різниця є лише, якщо між верхніми та нижніми частина образу. До того ж деякі картинки абсолютно не відповідають опису.

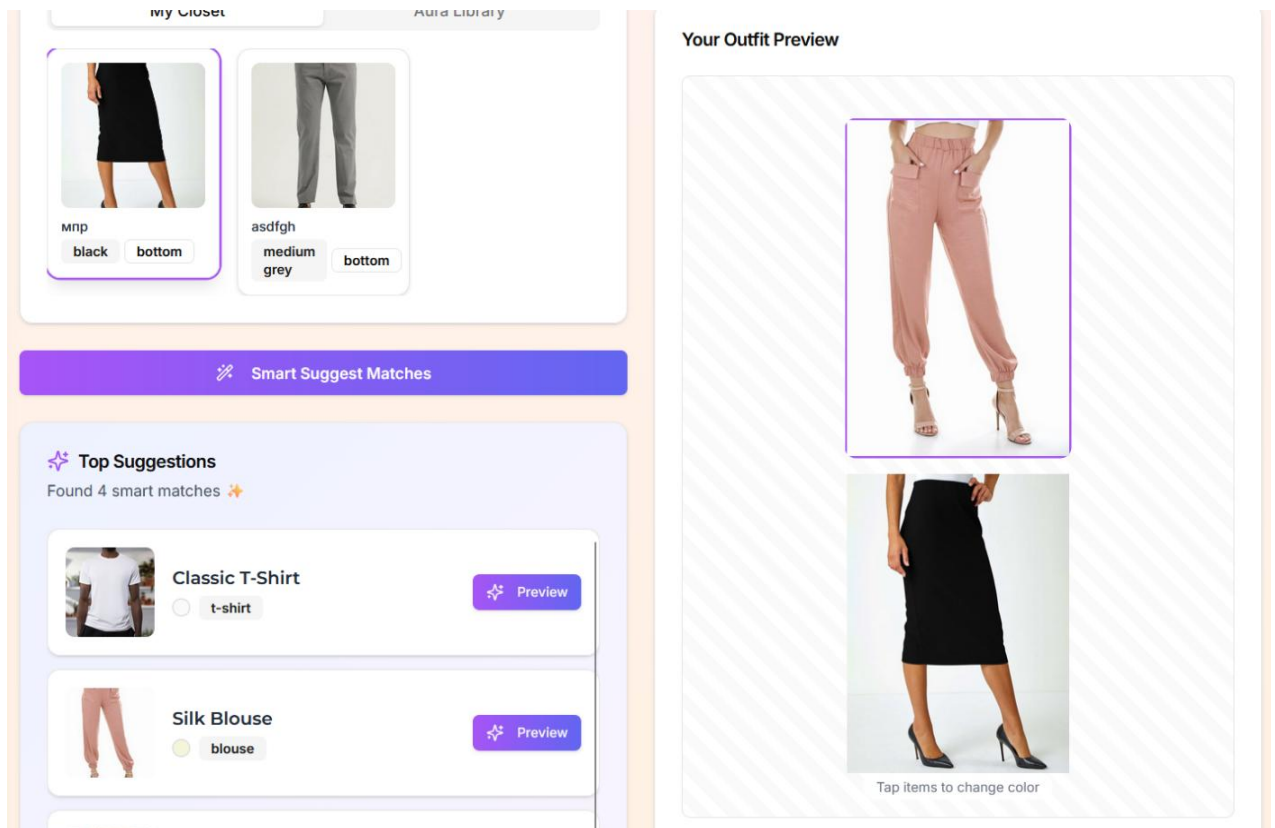


Рис. 1.10 Створений образ

При створенні ідей стилізації (Rewear Ideas) є три варіанти: створення образу за елементом (Aura Outfit Generator), створення образу з того що є в бібліотеці (Rewear Ideas) та створення образу з новими елементами (Shop New Fits). Перший варіант генерує п'ять зображень образів за вибраним елементом, але через те що це ШІ генерація елемент одягу не завжди виглядає так як повинен, наприклад при створенні образу з оливковими чоботами, на зображеннях вони ставали як балетками так й кедами.

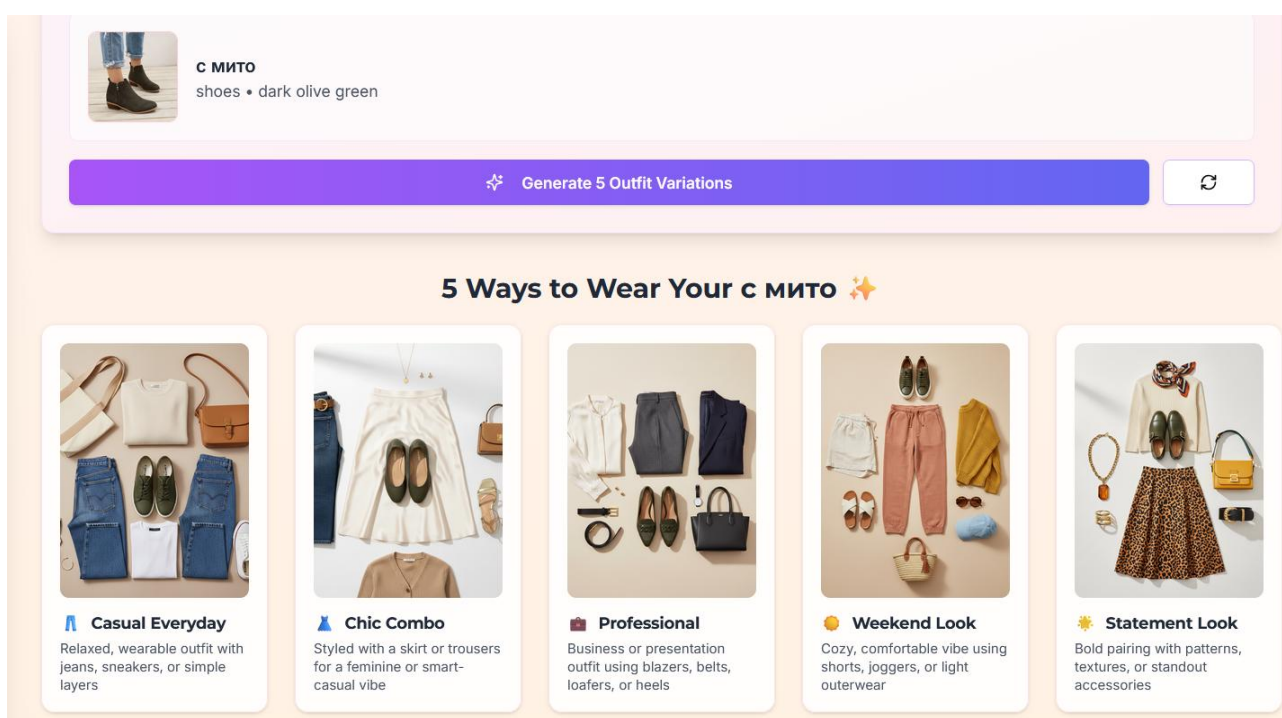


Рис. 1.11 Образи за елементом одягу

Інші два варіанти генерують список елементів одягу без будь-якого візуального відображення, лише з назвами одягу, тому важко оцінити гармонійність образів.

Загалом додаток може створити образи, але їх якість важко оцінити, адже більшість з них не мають ніякого візуального виду, крім того зберегти створені образи неможливо.

1.2.4 Результати аналізу

Аналіз показав, що основним недоліком багатьох веб-застосунків для формування персоналізованих стилістичних образів є нестабільна якість створюваних образів. В основному ці сервіси використовують ІІІ запити для створення образів, тому результати можуть виглядати як гарно, так і погано. У більшості застосунків наявні додаткові необов'язкові характеристики для одягу, які можуть бути корисними для користувача. Також у деяких з оглянутих веб-додатків є можливість ручного створення образів користувачем, що надає можливість користувачу спробувати себе в ролі стиліста.

Таблиця 1.1

Порівняння існуючих програм-аналогів

	AI Closet	Closet OS	Aura	PerVi Closet
Можливість збереження образів	Так	Так	Ні	Так
Наявність візуального представлення створених образів	Так	Так	Частково	Так
Додавання фото одягу за посиланням	Так	Так	Ні	Ні
Створення образу вручну користувачем	Так	Так	Ні	Ні
Основний спосіб створення образу	ІІІ запит	ІІІ запит	ІІІ запит	Рекомендаційна система правил

1.3 Визначення вимог до застосунку

Цільовою аудиторією даного продукту є люди, які цікавляться модою та стилем, але яким бракує знань або досвіду у цій області. Орієнтація на недосвідчених у галузі моди користувачів висуває свої вимоги до веб-застосунку для формування персоналізованих стилістичних образів.

По-перше, важливою вимогою є простота розуміння. В даному випадку це означає зменшення необхідних знань для роботи з додатком та обмеження використання специфічних для моди висловлювань та термінології. Іншим можливим варіантом є пояснення незрозумілих термінів простими словами, за допомогою підказок. Під час розробки вимог до ПЗ було обрано варіант зменшення кількості незрозумілої термінології.

По-друге, іншою важливою вимогою є зручність роботи. Акцент був на зручності для звичайного користувача, тому були виставлені вимоги до швидкості відповіді, а саме то, що час відгуку програми має не перебільшувати п'яти секунд, або мати індикатор прогресу [3, 4].

Останньою серед вимог, які залежать від цільової аудиторії є мобільна підтримка. Більшість людей користуються мобільними пристроями частіше, аніж комп'ютерами та ноутбуками, тому підтримка мобільних користувачів є необхідністю. Тим паче на ринку в якому мобільні пристрої є домінуючим способом використання ПЗ.

Цільова аудиторія диктує свої обмеження при розробці такі як: зрозумілість використаних термінів, швидкість роботи програми та кросплатформність, але крім наведених вище вимог, є й більш конкретні та пов'язані з роботою додатку, серед таких є безпека даних.

Для захисту даних облікових записів використовується хешування паролів за алгоритмом PBKDF2, який є частиною ASP.NET Core Identity, а для автентифікації та авторизації користувачів застосовуються JWT токени.

Функціональні вимоги до веб-застосунку для формування персоналізованих стилістичних образів включають в себе:

1. Реєстрацію та авторизацію користувачів.
2. Можливість завантаження зображень одягу.
3. Розпізнавання зображень одягу та додавання інформації про елементи одягу за зображеннями.
4. Перегляд списку завантаженого користувачем одягу.
5. Створення, редагування, видалення і перегляд елементів одягу.

6. Створення образу за стилем або за певним елементом одягу.
7. Збереження створених образів.
8. Перегляд та видалення та редагування створених образів.

Нефункціональні вимоги до веб-застосунку для формування персоналізованих стилістичних образів включають в себе:

1. Час відгуку системи не більше 3 сек.
2. Підтримка браузерів: Chrome, Firefox та Safari.
3. Адаптація інтерфейсу для розширень екранів 1920×1080 та 1366×768 , 360×800 та 390×844 [5].
4. Захист персональних даних використанням PBKDF2 хешування та JWT токенів.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

Для розробки ПЗ було використано два різних інтегрованих середовища розробки (IDE). IDE (Integrated development environment) - це програмне забезпечення, яке об'єднує в собі основні інструменти для створення, тестування, запуску та налагоджування програмного коду. Зазвичай IDE включає редактор коду, автоматичного компілювання та збору проєкту, інструменти для налагодження, менеджери бібліотек та засоби контролю версій.

Саме збір усіх основних інструментів для роботи з кодом та додаткові функції IDE спрощують та пришвидшують роботу розробника. Для розробки веб-застосунку «PerVi Closet» було використано середовище Visual Studio для створення серверної та клієнтської частини застосунку, а також Google Collaboratory для навчання згорткової нейронної мережі та експорту її як моделі ONNX для подальшого використання у веб-застосунку.

2.1.1 Visual Studio

Visual Studio - IDE від компанії Microsoft, яке підтримує створення програмного забезпечення різних типів: настільних програм, веб-застосунків, мобільних застосунків та серверних систем. Основою причиною вибору даного середовища розробки є гарна підтримка створення веб-додатків та зручність додавання необхідних при роботі фреймворків та бібліотек.

Visual Studio має велику кількість функцій та плагінів, які використовувались у роботі. Такими є: Server Explorer для роботи із серверними підключеннями та базами даних, Solution Explorer для управління структурою проєкту, Error List для перегляду помилок та попереджень, NuGet Package Manager для встановлення й оновлення бібліотек і фреймворків [6].

Іншою корисною функцією є інтелектуальне автодоповнення IntelliSense, яке аналізує написаний код і може пропонувати варіанти завершення недописаних команд чи автоматичне написання часто використовуваних фрагментів, наприклад клас із полями Id, Name, Description або створення базових конструкторів.

Також важливою перевагою Visual Studio є наявність інструментів для налагодження програмного забезпечення. Серед цих інструментів можна виділити: можливість покроково виконувати код, переглядати значення змінних при роботі коду, знаходити помилки та аналізувати роботу програми.

Загалом Visual Studio була незамінною програмою при розробці, адже використовувалася для створення клієнтського інтерфейсу, функціоналу та серверної частини веб-додатку.

2.1.2 Google Collaboratory

Google Collaboratory - хмарне IDE від компанії Google, яке підтримує мову програмування Python та більшість її бібліотек. Сервіс працює у браузері та не потребує встановлення додаткового програмного забезпечення на комп'ютер користувача.

Google Collaboratory доволі часто використовується для роботи з машинним навчанням, ШІ та комп'ютерним зором, адже сервіс має підтримку хмарних обчислювальних ресурсів та є простим у використанні

Однією з головних переваг Google Collaboratory є можливість використання графічних процесорів (GPU) та тензорних процесорів (TPU), які значно пришвидшують процес навчання нейронних мереж. Використання цих обчислювальних ресурсів є важливим при роботі зі згортковими нейронними мережами та великими наборами зображень. Крім того, середовище має підтримку більшості популярних Python-бібліотек, зокрема TensorFlow, Keras, NumPy, Pandas та OpenCV.

В даному проєкті Google Collaboratory використовувалася для створення моделі розпізнавання зображень одягу і тренування згорткової нейронної

мережі. Для виконання даних завдань можна було використати й Visual Studio, але можливість швидкого завантаження популярних Python бібліотек без використання термінала та загальна простота роботи з сервісом були вирішальними факторами для вибору саме цього середовища.

2.2 Мови програмування

У розробці було використано декілька мов програмування, адже кожна з них має свої способи використання та можливості.

В даному веб-застосунку було використано 5 різних мов програмування: C# для створення серверної частини та функціоналу додатку, Python для створення моделі розпізнавання одягу та тренування згорткової нейронної мережі, HTML та CSS для створення інтерфейсу користувача та стилів, JS для створення анімацій та додаткових функцій інтерфейсу.

Використання декількох мов програмування є типовим підходом у сучасній веб-розробці, оскільки створення повноцінного веб-застосунку потребує реалізації серверної логіки, роботи з базою даних, створення клієнтського інтерфейсу та підтримки динамічної взаємодії із користувачем.

2.2.1 C#

C# - об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Ця мова має широкий спектр використань, які в основному включають в себе розробку різних видів як мобільного так й десктопного ПЗ, ігор та веб-додатків [7].

Ця мова програмування була обрана для роботи через те, що вона має широкий спектр корисних фреймворків та бібліотек, а зокрема ASP.NET Core, який може допомогти створити додатки корпоративного рівня. Цей фреймворк забезпечує підтримку маршрутизації, роботи з базами даних, авторизації користувачів, створення API що є важливим для створення веб-застосунку. Також фреймворк ASP.NET Core має підтримку кросплатформних застосунків.

Іншою важливою перевагою C# є підтримка асинхронного програмування. Асинхронні методи та функції зменшують час очікування користувача під час довготривалих операцій, наприклад роботи з базою даних або завантаження зображень.

У межах розробки веб-застосунку «PerVi Closet» мова програмування C# використовувалася для реалізації основного функціоналу веб-застосунку, серверної логіки, роботи з базою даних, створення системи авторизації користувачів та інтеграції моделі розпізнавання одягу. Саме код на C# складає найбільшу частину загального обсягу програмного коду.

2.2.2 Python

Python - високорівнева мова програмування, яка є однією з найпопулярніших мов для роботи зі штучним інтелектом, машинним навчанням та аналізом даних. Завдяки простому синтаксису та великій кількості готових інструментів Python є однією з найпоширеніших мов програмування у сфері штучного інтелекту та аналізу даних.

Однією з головних переваг мови програмування Python є наявність великої кількості бібліотек для машинного навчання та обробки зображень. Під час роботи використовувалися бібліотеки PyTorch, NumPy та OpenCV, які спрощують навчання нейронних мереж, обробки даних та оцінювання точності нейронної мережі після її навчання.

Під час розробки веб-застосунку мова Python використовувалася для створення моделі комп'ютерного зору, а саме для навчання згорткової нейронної мережі, яка виконує розпізнавання елементів одягу на зображеннях. Крім навчання моделі, Python використовувався для обробки набору даних, тестування точності моделі та експорту готової нейронної мережі у формат для подальшого використання формат.

У даному проєкті навчена згорткова нейронна мережа була експортована у формат ONNX, що дозволило легко використовувати модель у веб-застосунку на платформі .NET без необхідності запуску Python середовища на сервері або

створення моделі з нуля на мові C#. Для роботи моделі необхідно було лише використати фреймворк ONNX Runtime.

2.2.3 HTML та CSS

HTML (HyperText Markup Language) - мова розмітки, яка використовується для створення структури веб-сторінок. За допомогою HTML можна додавати текст, кнопки, зображення тощо на сторінку та створювати вміст сторінок веб-застосунків.

CSS (Cascading Style Sheets) - мова стилів, яка використовується для оформлення візуальної частини веб-сторінок. Ця мова дозволяє змінювати кольори, шрифти, розміри елементів, відступи та анімації, цим самим забезпечуючи створення зовнішнього вигляду веб-застосунку. Крім того, CSS використовується для адаптації інтерфейсу під різні розміри екранів, для коректної роботи як на великих так і на малих екранах.

HTML та CSS майже завжди використовуються разом, оскільки HTML відповідає за структуру сторінки, а CSS — за її зовнішній вигляд. При створенні даного проєкту ці мови були основним інструментами реалізації інтерфейсу користувача. З їх допомогою були додані усі видимі елементи та написані стилі, які створили візуал веб-додатку.

2.2.4 JavaScript

JavaScript - об'єктно орієнтована мова програмування, яка зазвичай використовується для створення інтерактивності на веб-сторінках. Найчастіше її використовують разом із HTML та CSS. JS дозволяє: динамічно змінювати вміст сторінки, додавати анімації та додаткову поведінку елементам, наприклад показ модального вікна при натисканні кнопки.

Основною перевагою JavaScript є покращення взаємодії користувача із веб-застосунком та перевірка введених даних на стороні клієнта, яка зменшує навантаження на сервер, тим самим пришвидшуючи роботу веб-застосунку.

При розробці веб-додатку для створення персоналізованих стилістичних образів JS використовувалася для створення невеликих анімацій при наведенні на елементи та показу додаткових виборів при наведенні на елемент. \

Хоча JavaScript займає лише невелику частину написаного коду, вона позитивно впливає на зручність користування додатком та користувацький досвід використання продукту.

2.3 Фреймворки

Фреймворк - це ПЗ або набір бібліотек, який має у собі вже готовий функціонал, який можна використовувати для прискорення та спрощення розробки. Використання фреймворків дозволяє розробникам не створювати базові механізми з нуля, а використовувати вже готові рішення.

При веб-розробці фреймворки є важливою частиною розробки, адже вони спрощують інтеграцію елементів та допомагають створити структуру проєкту. Під час роботи на веб-застосунку для формування персоналізованих стилістичних образів було використано декілька різних фреймворків для роботи з C# веб-застосунком, але серед них можна виділити чотири основних: Razor Pages, Identity, Entity Framework Core та Onnx Runtime.

2.3.1 Razor Pages

Razor Pages - один з двох основних фреймворків ASP.NET Core для створення архітектури веб-додатків [8]. Фреймворк використовує підхід сторінково-орієнтованої архітектури, при якому кожна веб-сторінка має власний файл інтерфейсу та окремий клас із логікою роботи.

Razor Pages має схожу до MVC структуру, яка розділяє інтерфейс та бізнес логіку. Основною різницею між ними є те що в Razor Pages контролер виконує функціонал лише для однієї сторінки, тоді як контролер MVC зазвичай відповідає за одну сутність і може використовуватись для роботи з декількома сторінками.

У даному проєкті було використано фреймворк Razor Pages через простоту та швидкість роботи. Веб-застосунок для формування персоналізованих стилістичних образів має відносно невеликий масштаб, тому використання повної MVC-архітектури було б менш доцільним через складнішу структуру та більшу кількість файлів. Razor Pages дозволяє швидше створювати сторінки, реалізовувати обробку форм та взаємодію із серверною логікою.

Ще однією перевагою Razor Pages є тісна інтеграція з іншими компонентами ASP.NET Core, зокрема Entity Framework Core та Identity. Це значно спрощує реалізацію авторизації, роботи з базою даних та обробки HTTP-запитів.

2.3.2 Identity

Для забезпечення авторизації та автентифікації користувачів використано ASP.NET Core Identity та ASP.NET Core Authentication JwtBearer. Ці фреймворки допомагають створити функціонал реєстрації, входу в систему, зміни пароля та інших пов'язаних з акаунтом дій.

ASP.NET Core Identity — це фреймворк для керування користувачами, який містить готові механізми реєстрації, авторизації, зміни паролю, підтвердження електронної пошти та відновлення доступу до акаунту. Крім того, цей фреймворк має готовий клас користувача, який використовує хешування паролів для забезпечення конфіденційності даних користувача.

Для додаткового забезпечення безпеки у веб-застосунку було використано JwtBearer - компонент для роботи з JWT-токенами. JWT використовується для безпечної передачі інформації між клієнтом та сервером. Після успішної авторизації користувач отримує токен, який використовується для підтвердження особи під час виконання запитів до сервера, що забезпечує безпеку роботи та додатково підтверджує особу користувача.

Крім того, під час реалізації системи авторизації використовувалися стандартні механізми захисту ASP.NET Core, зокрема обмеження доступу до окремих сторінок для неавторизованих користувачів.

2.3.3 Entity Framework Core

Entity Framework Core - це кросплатформна фреймворк, який використовується для об'єктно-реляційного мапінгу, тобто дозволяє працювати з базою даних за допомогою об'єктів мови програмування C# [9].

Під час розробки за допомогою Entity Framework Core були створені таблиці для збереження інформації про користувачів, елементи одягу, стилістичні образи та інші дані веб-застосунку. Також EF Core використовувався для виконання операцій створення, редагування, видалення та отримання даних із бази даних.

Для роботи з базою даних було створено клас AppDbContext, який відповідає за взаємодію між моделями застосунку та таблицями бази даних. Через AppDbContext виконувалося отримання, додавання, оновлення та видалення даних, а також налаштування зв'язків між таблицями.

Entity Framework Core також підтримує механізм міграцій, який використовувався для автоматичного створення та оновлення структури бази даних під час розробки проєкту. Завдяки міграціям зміни у моделях можуть автоматично переноситися до БД.

2.3.4 ONNX Runtime

ONNX Runtime - це фреймворк для запуску моделей машинного навчання у форматі ONNX [10]. Основним призначенням даної технології є швидка та ефективна робота з моделями штучного інтелекту на різних платформах та мовах програмування. ONNX Runtime підтримує оптимізацію обчислень та дозволяє значно пришвидшити процес роботи моделей розпізнавання зображень.

Використання ONNX моделей було найбільш підходящим вибором для розробки даного проекту, адже інші способи є надто занадто повільними при роботі або потребують більшого часу для розробки. Серед інших способів розпізнавання зображень було проаналізовано створення моделі на мові C# за допомогою фреймворку Microsoft.ML та створення мікросервісу на мові Python.

Використання фреймворку Microsoft.ML не є підходящим для створенні моделі розпізнавання зображень через обмеженість роботи з машинним зором мовою C#, а мікросервісний варіант хоч і є найбільш потужним, він також є й найскладнішим у реалізації.

Тому ONNX модель є оптимальним компромісом між швидкістю та потужністю. Формат ONNX забезпечив просту інтеграцію натренованої згорткової нейронної мережі у .NET-застосунок та дозволив виконувати розпізнавання елементів одягу без необхідності створення окремого мікросервісу роботи зі штучним інтелектом.

2.4 Бібліотеки

Бібліотеки загалом можна порівняти з фреймворками, оскільки вони також являють собою збірку написаного іншими розробниками функціоналу, але вони не є однаковими. По-перше, бібліотеки зазвичай менші та більш вузьконаправлені, наприклад бібліотека може лише функціонал роботи з текстовими змінними. По-друге, фреймворки мають у собі абстракції, адже вони скоріше основа на якій будується ПЗ, аніж зручне доповнення.

Використання бібліотек значно спрощує процес розробки програмного забезпечення, оскільки дозволяє не реалізовувати складний функціонал вручну. Особливо важливими бібліотеки є під час роботи з машинним навчанням, комп'ютерним зором та аналізом даних, де створення власних алгоритмів з нуля потребувало б значної кількості часу та ресурсів. У проекті було використано різні бібліотеки при роботі з Python кодом та обробкою зображень, основними з яких є: Pytorch, Pandas та OpenCV.

2.4.1 Pytorch

PyTorch є однією з найбільш популярних бібліотек для навчання нейронних мереж і роботи з системами машинного зору [11]. Бібліотека набула широкої популярності завдяки простоті використання, гнучкості та високій продуктивності. Основою PyTorch є використання тензорів - багатовимірних структур даних, які можуть виконувати обчислення з використанням графічних процесорів.

Робота з тензорами дозволяє пришвидшити навчання нейронних мереж, особливо при роботі з зображеннями. Для створення нейронних мереж у бібліотеці є модуль `torch.nn`, який містить готові шари, функції активації, механізми оптимізації та інші інструменти для побудови моделей штучного інтелекту.

Бібліотека PyTorch була використана під час розробки веб-застосунку для формування персоналізованих стилістичних образів для створення моделі розпізнавання одягу, а точніше навчання згорткової нейронної мережі та експорту мережі в ONNX формат для подальшого використання в веб-додатку.

Під час створення моделі розпізнавання одягу в PyTorch використовувалась готова згорткова нейронна мережа ResNet18. Використання попередньо навченої моделі дозволило скоротити час на навчання та покращити точність розпізнавання.

PyTorch підтримує використання графічних процесорів CUDA, які набагато краще підходять для роботи з великими наборами зображень, аніж звичайні процесори. Використання CUDA дуже скоротило час необхідний для створення та навчання моделі.

2.4.2 Pandas

Pandas - популярна бібліотека для мови Python, яка використовується для роботи з даними та їх аналізу [12]. Зазвичай цю мову використовують при

роботі з ШІ, адже вона може допомогти підготувати дані для тренування різних моделей.

Одним із головних компонентів бібліотеки є структура `DataFrame`, яка дозволяє працювати з даними у вигляді таблиць, що спрощує роботу з великими наборами даних. Крім того, `Pandas` має функціонал для роботи з `CSV` файлами.

У даному проєкті `Pandas` використовувалася для підготовки даних перед навчанням моделі розпізнавання одягу. Бібліотека допомагала працювати із файлами, які містили інформацію про зображення та їх категорії, а також виконувати структурування даних для подальшого використання під час навчання нейронної мережі.

2.4.3 OpenCV

`OpenCV` (`Open Source Computer Vision Library`) - це бібліотека для роботи з машинним зором. Вона є однією з найпоширеніших бібліотек для аналізу зображень та відео й активно використовується у системах машинного навчання, розпізнавання об'єктів та комп'ютерного зору.

Однією з основних переваг бібліотеки є висока швидкість обробки зображень та підтримка великої кількості алгоритмів комп'ютерного зору. Висока швидкість обробки досягається за рахунок використання багатопотокових обчислень та апаратного прискорення.

`OpenCV` містить велику кількість інструментів для роботи із зображеннями, такі як: зміна розміру, фільтрація, обробка кольорів, пошук об'єктів тощо. Бібліотека підтримує роботу з різними форматами зображень.

В проєкті бібліотека `OpenCV` в основному використовувалася для визначення основного кольору елементу одягу та характеристик його кольору, таких як: відтінок, тон та яскравість. Ці характеристики були використані для підбору гармонійних кольорових комбінацій при створенні стилістичних образів.

Під час аналізу кольору одягу в проєкті використовувалася модель HSV, яка дозволяє окремо працювати з відтінком, насиченістю та яскравістю кольору. Перетворення зображень із RGB у HSV за допомогою OpenCV спростило визначення основного кольору елементів одягу.

Окрім аналізу кольорів, OpenCV використовувалася для попередньої обробки зображень при навчанні згорткової нейронної мережі. Попередня обробка включала в себе зміну розміру зображень, нормалізацію та підготовку даних до формату, необхідного для роботи з нейронною мережею. Такий підхід дозволив підвищити стабільність та точність роботи моделі. Це дозволило покращити точність роботи згорткової нейронної мережі.

2.5 Система контролю версій

Під час розробки було використано систему контролю версій GitHub. Системи контролю версій є важливим інструментом сучасної розробки програмного забезпечення, оскільки дозволяють зберігати історію змін проєкту та керувати різними версіями коду [13].

Основною перевагою використання систем контролю версій є можливість збереження декількох версій одного й того самого програмного коду або файлу. Завдяки цьому розробник може у будь-який момент повернутися до попереднього стану у випадку появи помилок або некоректної роботи нових змін.

У межах даного проєкту GitHub використовувався для збереження всіх етапів розробки веб-застосунку, резервного копіювання коду та контролю внесених змін. GitHub використовувався як віддалений репозиторій для збереження програмного коду та синхронізації змін між різними версіями проєкту та виконував роль резервного сховища програмного коду.

Використання системи контролю версій також спростило процес внесення нових змін до проєкту. Кожна зміна коду може бути збережена

окремо, що дозволяє переглядати історію розробки, аналізувати зміни та швидко знаходити причини помилок у роботі застосунку.

2.6 Інструменти дизайну інтерфейсу

У межах розробки web-застосунку для підбору персоналізованих стилістичних образ було використано сервіс Figma для створення прототипу майбутнього інтерфейсу користувача та дизайну сторінок застосунку.

Figma - один з найпопулярніших застосунків для проєктування інтерфейсу користувача, має як веб-версію так і додаток на різні пристрої. Figma дозволяє створювати макети та інтерактивні прототипи сторінок, проєктувати елементи інтерфейсу та перевіряти взаємодію між різними екранами застосунку. Однією з переваг Figma є можливість швидкого внесення змін до дизайну та автоматичного оновлення всіх пов'язаних компонентів.

За допомогою даного сервісу було створено загальний стиль інтерфейсу користувача, підібрано кольорову гаму, створено шаблони сторінок та реалізовано логіку переходів між ними. Крім того, у Figma були створені прототипи сторінок для різних типів пристроїв, що дозволило заздалегідь перевірити адаптивність інтерфейсу та зручність використання застосунку на смартфонах, планшетах та персональних комп'ютерах.

Використання даного сервісу полегшило створення HTML та CSS для усіх сторінок додатку, через можливість копіювання розмірів з прототипу та легкого перегляду змін між різними розмірами екранів.

2.7 База даних та СУБД

Для збереження даних про користувачів, елементи одягу та образи було використано реляційну базу даних MSSQL та SQL Server Manager. Вибір MSSQL був обумовлений його високою продуктивністю, надійністю та простою інтеграцією з ASP.NET Core.

Перевагою реляційних баз даних є чітка структура таблиць, яка дозволяє уникати помилок збереження інформації та підтримувати правильні зв'язки між даними. Також SQL Server підтримує використання транзакцій, які забезпечують коректне виконання операцій з базою даних навіть у випадку виникнення помилок або збоїв під час роботи системи.

Структура бази даних веб-застосунку «PerVi Closet» була спроектована з урахуванням зв'язків між основними сутностями системи. Для кожного користувача може зберігатися декілька елементів одягу та створених образів. Зв'язок між образами та елементами одягу реалізовано за допомогою проміжної таблиці OutfitItem, що дозволяє використовувати один елемент одягу у різних образах.

Для взаємодії серверної частини застосунку з базою даних використовувався Entity Framework Core. За його допомогою класи моделей автоматично відображаються у таблиці бази даних, а зміни структури можуть бути перенесені до БД через міграції. Це спрощує процес розробки та підтримки структури бази даних під час розвитку проєкту.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Для реалізації веб-застосунку «PerVi Closet» була обрана клієнт-серверна архітектура, яка найчастіше використовується при створенні веб-сайтів та веб-додатків. Ця архітектура є найбільш підходящою для роботи з фреймворком Razor Pages, який є основою створюваного застосунку.

Клієнт-серверна архітектура заснована на ідеї комунікації двох сутностей клієнта та сервера [14]. Клієнт надсилає HTTP-запити до сервера, а сервер обробляє ці запити та повертає відповідь у вигляді HTML-сторінок, JSON-даних або інших типів інформації.

Клієнтом у даному застосунку вважається браузер, який використовує користувач, а сервером веб-додаток, який обробляє дані. Серверна частина відповідає за обробку даних користувача, виконання алгоритмів підбору стилістичних образів, взаємодію з базою даних та інтеграцію моделі розпізнавання одягу.

Для відображення архітектури та взаємодії основних модулів веб-застосунку була створена діаграма компонентів, яка демонструє взаємозв'язки між клієнтською частиною, серверними модулями, базою даних та моделлю розпізнавання зображень.

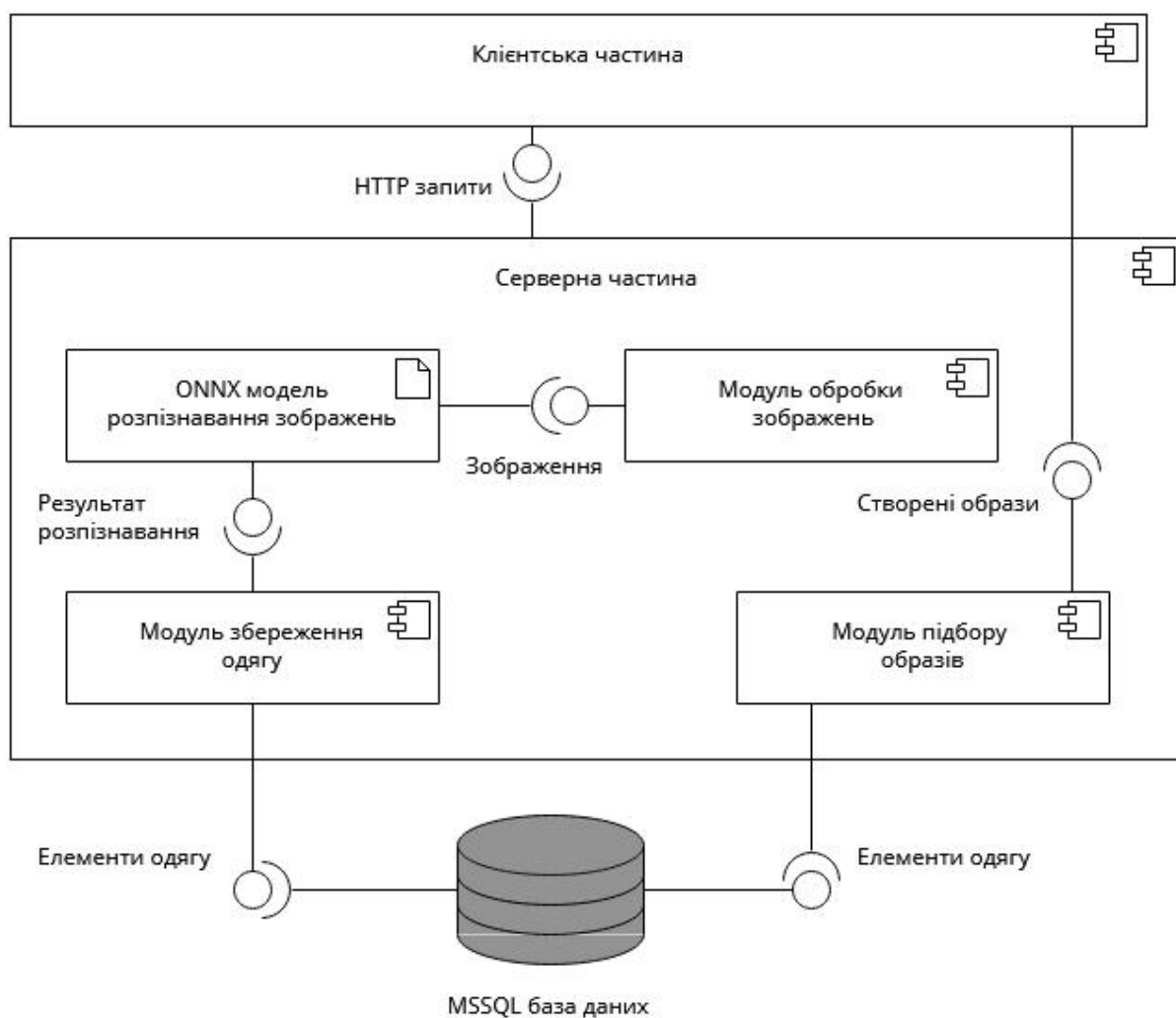


Рис. 3.1 Діаграма компонентів

Модуль обробки зображень виконує підготовку зображення для подальшого розпізнавання ONNX моделлю. Модуль збереження одягу відповідає за збереження характеристик розпізнаних елементів одягу до бази даних. Модуль підбору образів використовує елементи гардероба користувача для формування персоналізованих стилістичних рекомендацій.

Використання клієнт-серверної архітектури забезпечує кросплатформність додатку, адже залежить лише від HTML запитів, а не пристрою користувача. Також така архітектура може підтримувати роботу з кількома одночасними користувачами та є більш безпечною через відсутність прямого доступу клієнту до серверної частини веб-застосунку [15]. А серед

негативних сторін даної архітектури можна виділити: єдину точку відмови, залежність від мережі та більшу витрату ресурсів системи.

Через використання Razor Pages як основи веб-застосунку для кожної сторінки створюється відповідний PageModel клас, який відповідає за обробку HTTP-запитів, підготовку даних та взаємодію із сервісами. Такий підхід спрощує реалізацію CRUD-операцій та зменшує складність проєкту порівняно з використанням MVC архітектури.

Загалом клієнт-серверна архітектура є найбільш доцільним вибором для реалізації веб-застосунку «PerVi Closet», оскільки вона забезпечує кросплатформність та зручну інтеграцію з технологіями ASP.NET Core і Razor Pages.

3.2 Архітектура клієнтської частини

Клієнтська частина веб-застосунку для формування персоналізованих стилістичних образів складається з мов програмування HTML, CSS та JavaScript. Використання цих мов дозволяє створити функціональні та візуально привабливі сторінки та інтерфейс.

HTML використовувався для створення структури всіх сторінок веб-застосунку та розміщення всіх елементів інтерфейсу. Кожна сторінка має власну структуру та набір елементів залежно від її призначення, тому HTML для кожної сторінки значно відрізняється, але CSS та JavaScript писалися з приділенням уваги до можливостей масштабування та повторного використання коду.

Оскільки веб-застосунок реалізований із використанням фреймворку Razor Pages, HTML-сторінки містять вставки C# коду, які забезпечують взаємодію із серверною частиною застосунку. За допомогою Razor-синтаксису реалізується відображення даних користувача, зокрема списків завантаженого одягу, створених образів, інформації про окремі елементи гардеробу, а також даних облікового запису користувача.

Через використання Razor Pages усі HTML сторінки окрім звичної розмітки та тегів містять вставки C# коду для взаємодії деяких компонентів з моделлю сторінки, тобто з серверною частиною застосунку. Серед особливостей Razor синтаксису є такі директиви, як: `@model`, `@if`, `@foreach`. За цих директив було створено такий функціонал як: відображення списку завантаженого користувача одягу та списку створених образів, відображення даних про конкретний елемент або образ з бази даних, відображення імені користувача тощо.

Окрім звичайних сторінок, у застосунку використовуються часткові сторінки (Partial Views), які є складовою ASP.NET Core. Вони дозволяють повторно використовувати деякі елементи інтерфейсу на різних сторінках веб-застосунку. Наприклад, Partial Views використовувалися для реалізації навігаційної панелі, меню користувача та елементів авторизації.

Сторінки веб-застосунку розроблювали з урахуванням логічних переходів між сторінками, нижче зображена діаграма варіантів використання та карта сторінок веб-додатку.

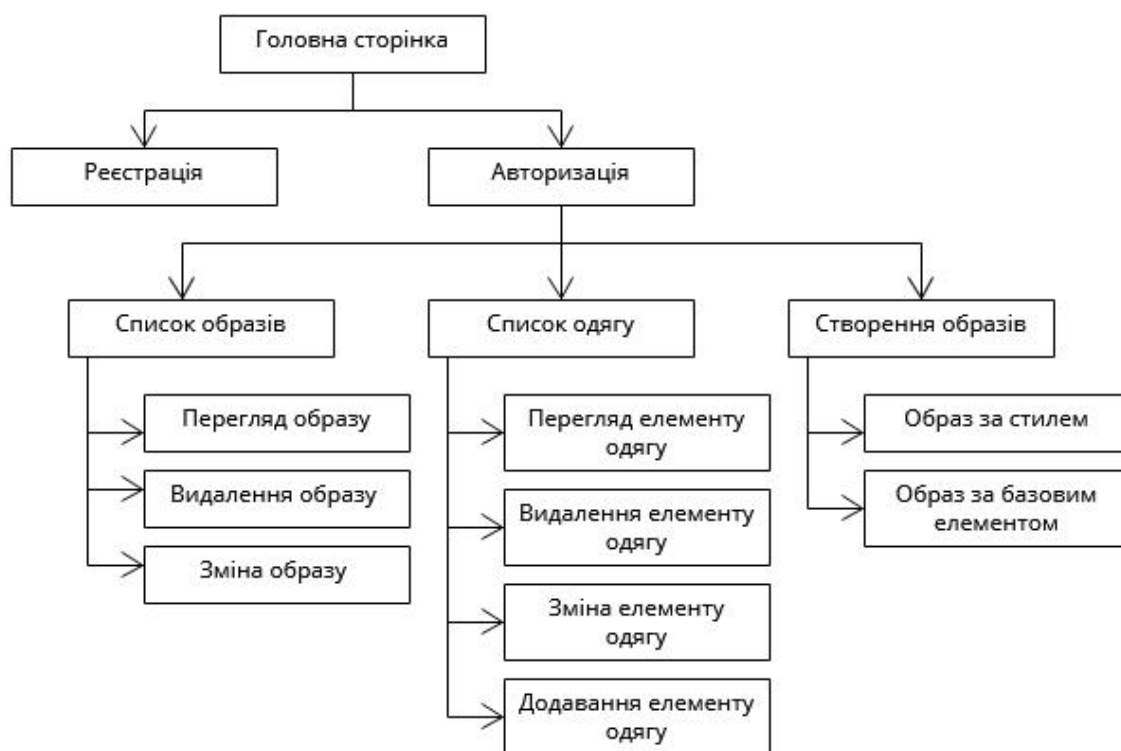


Рис. 3.2 Структура сторінок веб-застосунку

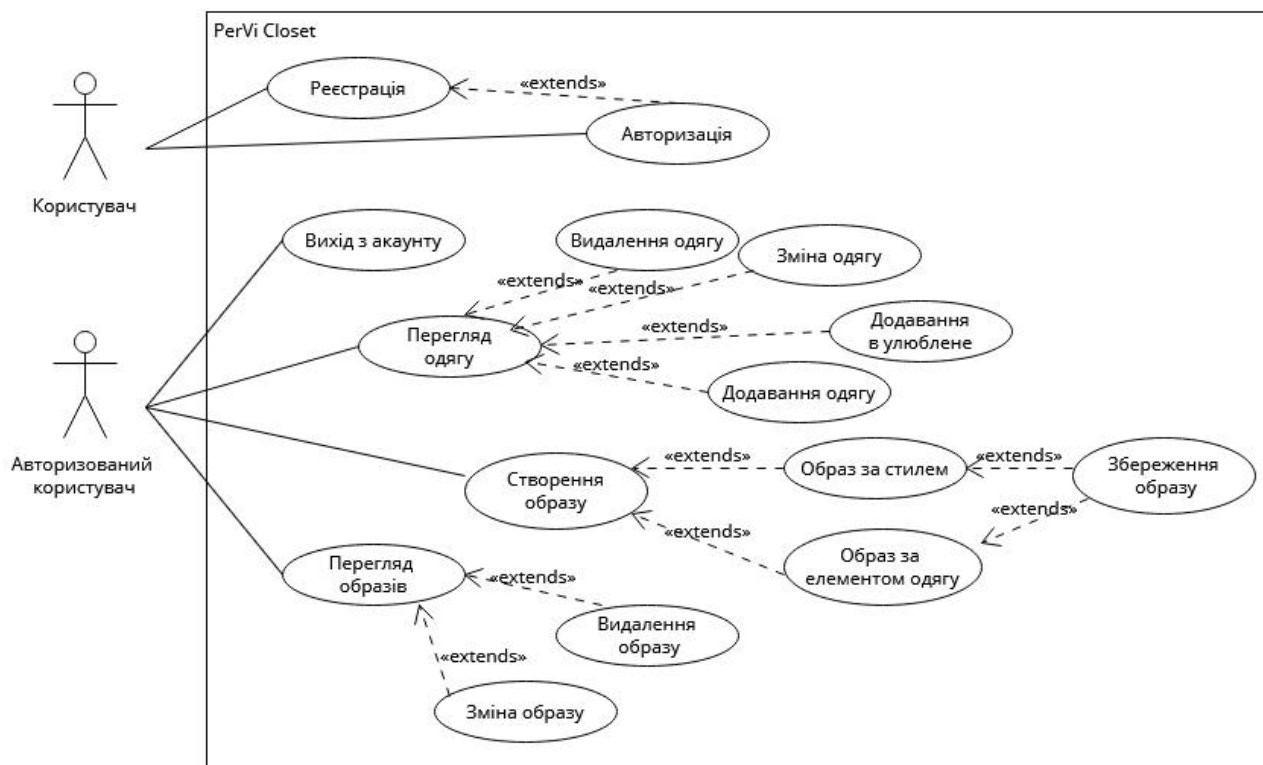


Рис. 3.3 Діаграма варіантів використання

CSS-стилі використовувалися для формування вигляду користувацького інтерфейсу та адаптивності веб-застосунку. Значна частина стилів була реалізована окремо для мобільних та десктопних пристроїв. Для адаптації інтерфейсу застосовувалися *media queries*, що дозволяють автоматично змінювати структуру сторінки залежно від розміру екрана пристрою.

При написанні CSS стилів за основу було обрано розширення екрану 393x852 пікселі яке є одним з п'яти найбільш поширених мобільних розширень екранів, а пізніше було написано селектори для розміру екрану 1920x1080, який є найбільш популярним десктопним розширенням екрану. До названих раніше загальних селекторів входять і деякі зміни, такі як; кольори, розміри шрифтів та назви шрифтів [16].

JavaScript використовувався для реалізації інтерактивних елементів інтерфейсу та створення невеликих анімацій. Наприклад, за допомогою JavaScript реалізовувались ефекти наведення курсору, відкриття додаткових

меню, а також динамічна зміна окремих елементів сторінки без її повного перезавантаження.

3.3 Архітектура серверної частини

Серверна частина веб-застосунку відповідає за обробку даних користувача, взаємодію з базою даних, реалізацію алгоритмів підбору одягу, авторизацію користувачів та інтеграцію моделі розпізнавання зображень. Під час проєктування серверної частини особлива увага приділялася структурованості програмного коду, розділенню його за логічні частини.

Веб-застосунок «PerVi Closet» створювався з використанням фреймворку ASP.NET Core Razor Pages. Структура проєкту складається з декількох директорій, кожна з яких зберігає файли з різним призначенням. Основними директоріями є Pages, Models та Services. Інші директорії або створювалися автоматично при роботі фреймворків, або не мають великої кількості коду.

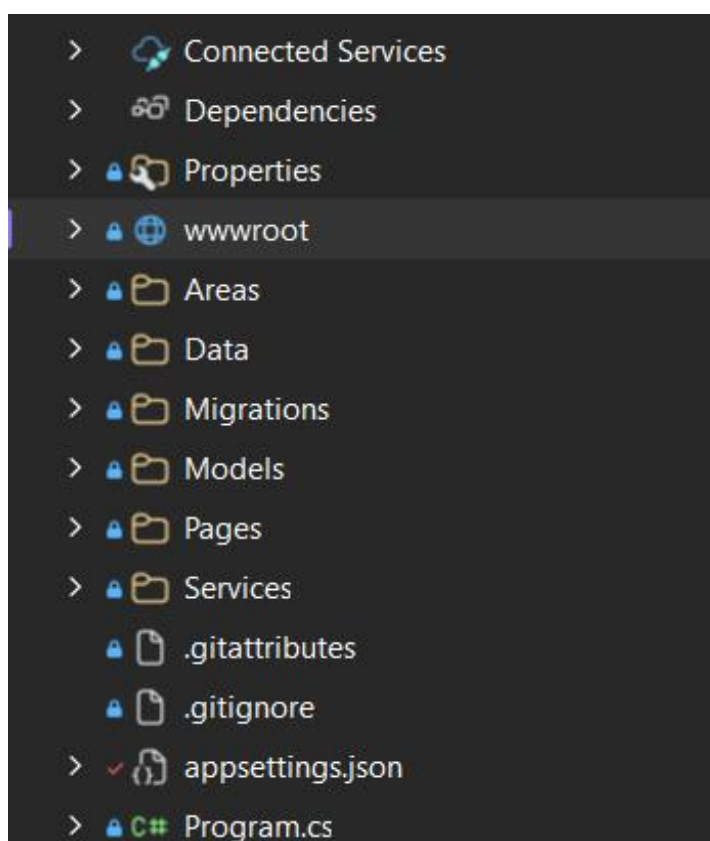


Рис. 3.4 Структура файлів проєкту

Директорія Pages є основою Razor Pages та містить сторінки та їх моделями. Таким чином разом зберігаються як HTML, так й C# файли, які відповідають за клієнтську та серверну частину веб-застосунку.

Директорія Pages також містить декілька піддиректорій, зокрема LoggedIn, Clothes та Outfits. Директорія LoggedIn включає всі сторінки доступ до яких можна отримати після авторизації [17]. Серед сторінок, які є доступними без авторизації є лендінг сторінка та сторінки для реєстрації й авторизації.

Директорія Clothes містить сторінки, пов'язані з переглядом, додаванням, редагуванням та видаленням елементів одягу. Саме в цих сторінках реалізовано основний функціонал роботи з гардеробом користувача. Директорія Outfits включає сторінки для створення, перегляду, редагування та видалення стилістичних образів.

Однією з ключових функцій веб-застосунку є автоматичне формування стилістичних образів. Для реалізації цього функціоналу було розроблено алгоритми підбору одягу за обраним стилем та за окремим елементом гардероба. Нижче наведено діаграму діяльності, яка відображає принцип роботи алгоритму створення образу.

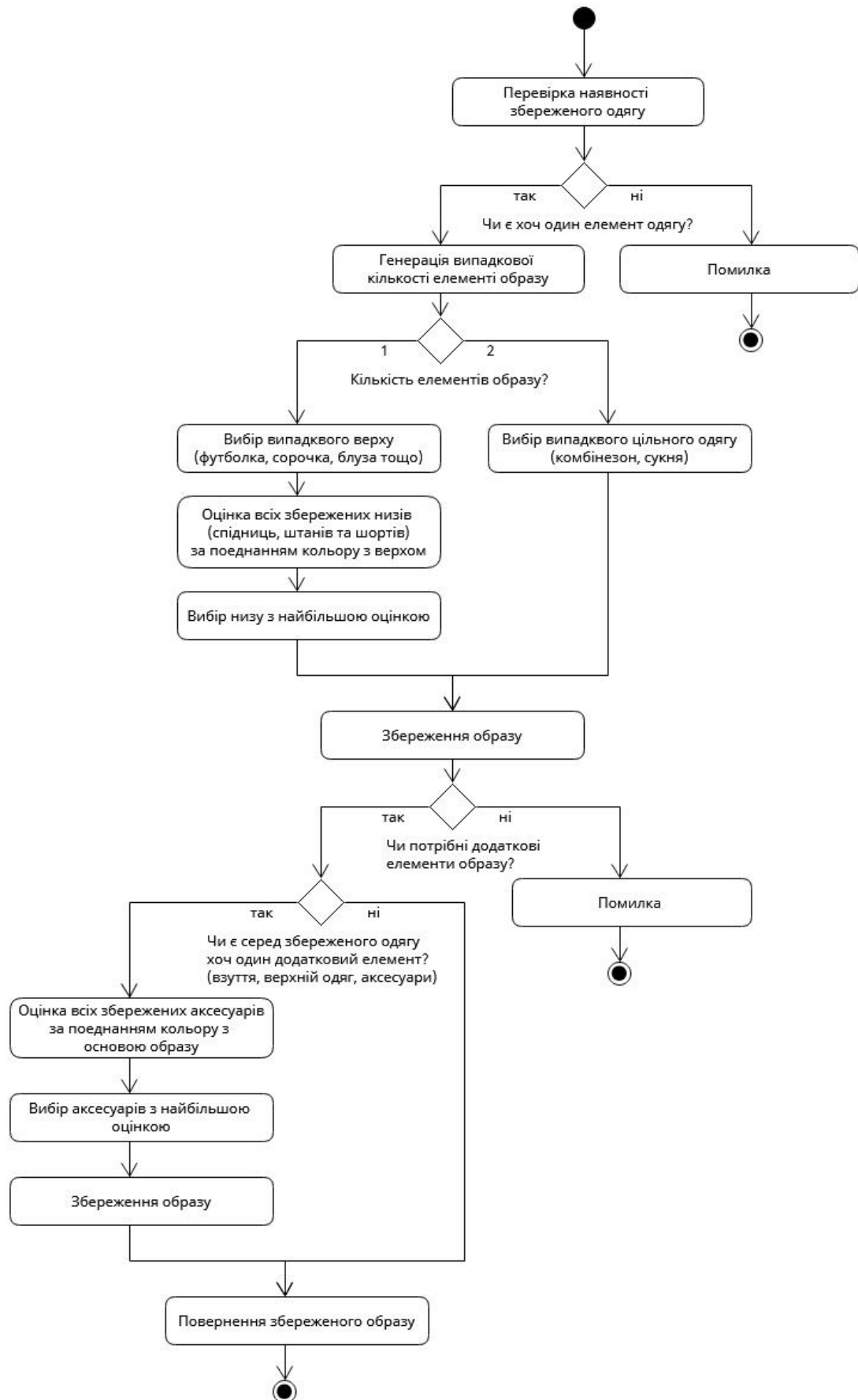


Рис 3.5 Діаграма діяльності створення образу

Директорія Models містить усі створені класи веб-застосунку. Частина цих класів зберігається БД. Ці класи використовуються для збереження даних користувача, збереження даних про елемент одягу, збереження зображень у вигляді бітів, збереження елементів образу ат збереження образу.

Інша частина класів не зберігається у базі даних і використовується лише під час виконання програми. До таких класів належать класи для передачі даних між сторінками, результати роботи алгоритмів, класи для роботи з моделлю розпізнавання зображень, а також переліки (enum), які використовуються для опису категорій одягу, кольорів, стилів та інших характеристик.

Директорія Services включає допоміжний функціонал, який не прив'язаний до окремої сторінки, але використовується різними частинами веб-застосунку. У даному проєкті сервіси використовувалися для реалізації алгоритмів підбору одягу, роботи з моделлю розпізнавання зображень, аналізу кольорів та інших допоміжних функцій. Використання сервісів дозволяє винести бізнес-логіку за межі сторінок та уникнути дублювання коду.

Для взаємодії з базою даних використовувався Entity Framework Core, який реалізує технологію ORM. Завдяки цьому робота з даними здійснювалася через C# об'єкти без створення SQL-запитів вручну.

Під час проєктування серверної частини застосовувався принцип розділення відповідальності між частинами системи. Моделі сторінок відповідали за обробку HTTP-запитів та взаємодію з користувачем, сервіси реалізовували бізнес-логіку й алгоритми, а моделі використовувалися для збереження та передачі даних. Такий підхід відповідає сучасним принципам розробки програмного забезпечення та значно спрощує підтримку й подальший розвиток веб-застосунку.

4 РЕАЛІЗАЦІЯ

4.1 Створення моделі розпізнавання зображень

Під час опису системи розпізнавання зображень будуть використанні два різних терміни: «модель розпізнавання зображень» та «згорткова нейронна мережа». Термін «модель розпізнавання зображень» буде використовуватися при загальному описі роботи чи взаємодії з C# кодом, адже модель збережена в ONNX форматі до якого зазвичай вживається термін модель. Термін «згорткова нейронна мережа» буде використовуватись під час опису взаємодії з Python кодом, адже основою роботи моделі є саме згорткова нейронна мережа. Кожен термін окремо не повністю описує модель, адже є описує лише одну складову: формат збереження або спосіб роботи.

Для створення моделі розпізнавання зображень було використано мову програмування Python та декілька бібліотеки для роботи з: файлами (cv2, json, os, google.colab), математичними функціями (numpy), системами кольорів (colorsys), випадковими значеннями (random), глибоким та машинним навчанням (torch, sklearn) [18]. Основою створюваної моделі є згорткова нейронна мережа ResNet18 та набори зображень одягу та взуття DeepFashion і UT Zappos50K.

Перед створенням моделі було виділено основні характеристики зображень одягу, які вона повинна розпізнавати, ними стали: категорія, підкатегорія, тип одягу та характеристика його основного кольору (відтінок, тон та яскравість) [19]. Під категорією мається на увазі те якою частиною одягу вважається елемент, наприклад це є взуття, верхній одяг чи нижній одяг. Підкатегорія є більш точною категорією, наприклад нижній одягу поділяється на: штани, спідниці та шорти, і вже в кожній з цих підкатегорії я свої типи одягу.

Використаний навчання нейронної мережі набір даних в основному складається з зображень одягу з датасетів DeepFashion та UT Zappos50K, але також було включено знайдені в інтернеті зображення одягу з безкоштовною ліцензією використання. Додаткові зображення були необхідними, адже співвідношення кількості та різноманіття зображень в деяких категоріях датасетів були недостатніми. Всі збережені зображення було поділено на папки які названі за категоріями, підкатегоріями та типами одягу. В кожній папці з типом є мінімум 100 різних зображень одягу. Далі було створено CSV файл з описом розташування та характеристик кожного зображення, який було використано для навчання згорткової нейронної мережі.

Для забезпечення стабільної роботи моделі всі зображення перед навчанням проходили етап попередньої обробки. Зображення масштабувалися до однакового розміру та перетворювалися у формат, придатний для подальшої обробки згортковою нейронною мережею.

Після завершення навчання мережі було проведено оцінювання її точності. Для оцінки використовувалися метрики accuracy та loss, які дозволяють визначити якість класифікації зображень. Accuracy показує наскільки багато правильних відповідей надала нейронна мережа, а loss показує кількість помилок. Отримані значення метрик та річна перевірка декількох зображень показали, що модель здатна коректно визначати основні характеристики кольору та типу одягу за умов різного освітлення, фону та ракурсу зображення.

Окрім класифікації одягу, під час підготовки системи також було реалізовано визначення кольору одягу. Для цього використовувалися алгоритми аналізу кольорових каналів зображення та перетворення кольорів у моделі HSV. Отримані характеристики надалі використовувалися у системі підбору для оцінювання гармонійності поєднання елементів одягу.

Створення моделі розпізнавання зображень включало декілька етапів: збір та підготовку даних, структурування набору зображень, створення CSV-файлу з характеристиками, побудову архітектури згорткової нейронної мережі,

навчання моделі та експорт результату у формат ONNX для подальшої інтеграції у веб-застосунок «PerVi Closet».

4.2 Реалізація серверної частини

Серверна частина веб-застосунку для формування персоналізованих стилістичних образів була створена за допомогою платформ .NET та Razor Pages, на мові програмування C#.

Веб-застосунок складається з одного проекту поділеного на різні директорії. Серверною частиною можна вважати директорії: Models, Services та Pages. В директорії Models знаходяться файли з класами та переліки (enum), які використовуються в проєкті. Services зберігає функції, які використовуються в декількох сторінках, або потребують ініціалізації в program.cs. Pages зберігає cshtml та cs файли для кожної сторінки додатку, але серверною частиною вважаються лише cs файли, які відповідають за функціонал сторінок.

Якщо говорити детальніше про директорію Models, то вона має в собі такі класи, як:

1. User - зберігає дані користувача.
2. ClothingItem - зберігає дані про елементи одягу.
3. ImageBlob - зберігає зображення у вигляді бітів.
4. OutfitItem - проміжний клас між ClothingItem та Outfit.
5. Outfit - зберігає дані про образ.
6. Style - зберігає дані про типи та кольори одягу, які відповідають та не відповідають стилю.
7. Color - зберігає дані про колір одягу.
8. ImageInput та ImageOutput - для роботи з моделлю розпізнавання зображень, а саме для передавання зображення та отримання даних про зображення.
9. OutfitMakerResult - зберігає в собі створений образ, текст помилки, якщо щось пішло не так під час створення образу.

З перелічених класів лише клас User, ClothingItem, ImageBlob, OutfitItem, Outfit зберігаються у базі даних, інші класи є необхідними, але тимчасовими, тому не зберігаються у БД. Для класів, що зберігаються в базі даних було створено конструктор AppDbContext, для створення міграцій, тобто перетворення класу в таблицю для бази даних.

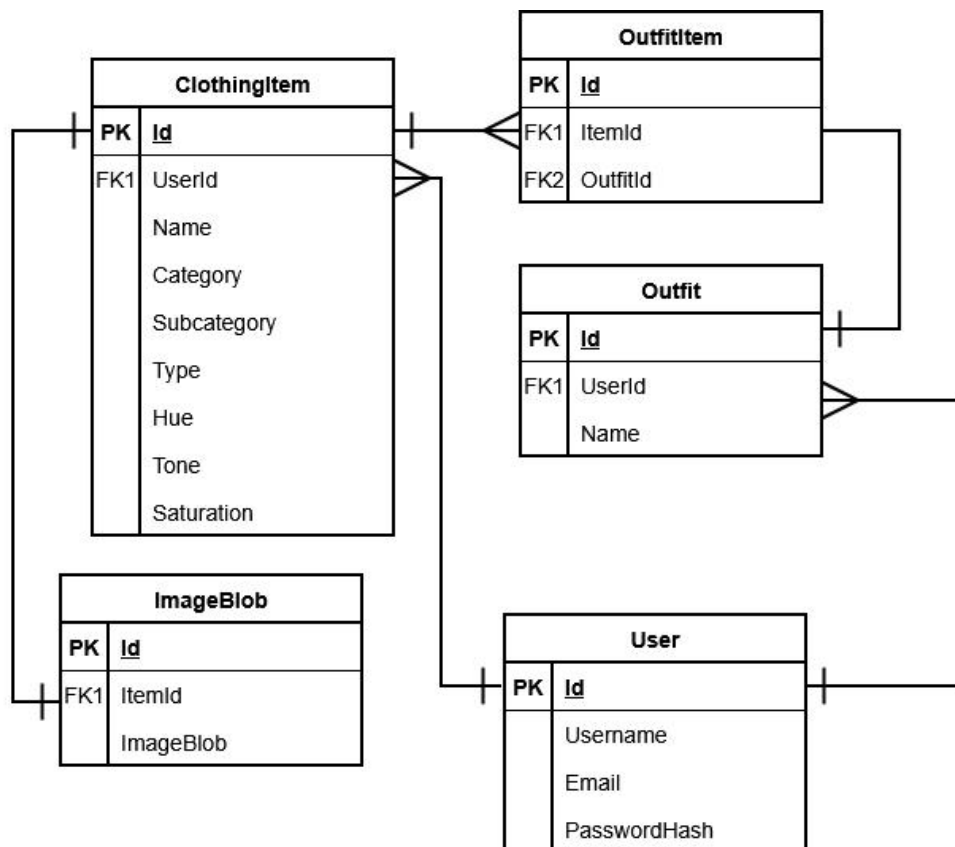


Рис. 4.1 ER Діаграма БД

В веб-застосунку для взаємодії з базою даних у використовується фреймворк Entity Framework Core, який дозволяє працювати з даними у вигляді C# об'єктів без необхідності створення SQL-запитів вручну. Цей фреймворк був використаний для створення SQL таблиць з класів User, ClothingItem, ImageBlob, OutfitItem, Outfit та для записування та отримання даних з БД.

Окрім моделей та бази даних важливою частиною веб-застосунку є сервіси, які реалізують велику частину функціоналу застосунку. Серед найбільш важливих сервісів можна виділити такі:

1. ClothingPredictor - передає зображення моделі та отримує результат.

2. ImageService - клас з функціями нормалізації та перевірки валідності зображень.
3. JsonLabelDecoder - декодер для переведення числових значень характеристик зображень в текст.
4. PredictionResultsSaver - зберігає результат розпізнавання зображень у БД.
5. OutfitMaker - має функції для створення образів, перевірки валідності списку одягу користувача.
6. JwtTokenService - має функції для створення та перевірки валідності JWT токенів.

Всі ці сервіси є необхідними для роботи веб-застосунку та відповідають за його основний функціонал та важливі функції. Загалом більшість сервісів відповідає за різні етапи в процесі розпізнавання зображень. Виключенням є сервіси ClothingPredictor, ImageService та PredictionResultsSaver працюють разом задля розпізнавання та збереження зображень одягу навантажених користувачем.

Спочатку файл зображення перевіряється на розмір та тип файлу, а після успішної перевірки нормалізується за допомогою функцій в ImageService. Далі в ClothingPredictor дані обробленого зображення передаються моделі розпізнавання зображень, яка повертає значення характеристик зображення (категорія, підкатегорія, тип тощо) у числовому форматі. Після цього в PredictionResultsSaver числові значення характеристик перетворюються на текстові і зберігаються у базі даних.

Нижче приведено структуру директорії Services та файлів, які знаходяться в ній.

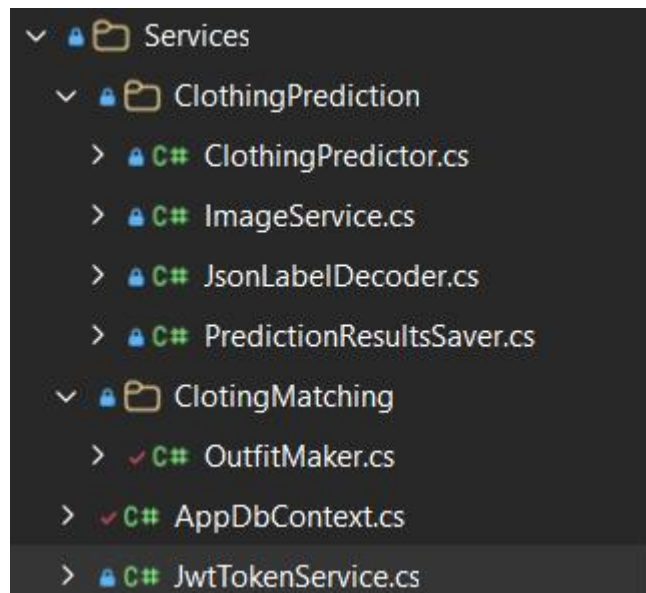


Рис. 4.2 Структура директорії Services

Окрім моделей та сервісів є також сторінки, які зберігаються у директорії Pages та відповідають за інтерфейс та функціоналі окремих сторінок. В директорії Pages є багато файлів зі сторінками, але серед них основними можна назвати:

1. UserClosetModel - відповідає за отримання даних про збережені користувачем елементи одягу.
2. ClothingImageUploadModel - використовується для додавання одягу.
3. ClothingMatchModel - відповідає за створення образів.
4. OutfitClosetModel - відповідає за отримання даних про збережені користувачем образи.

В директорії наявні й інші сторінки, але вони або мають доволі малу кількість коду, або були створені за допомогою фреймворків. Серед створених автоматично можна виділити сторінки Identity, які відповідають за логін та авторизацію і сторінки CRUD операцій створені для одягу та образів.

Також задля забезпечення безпеки веб-застосунку було реалізовано систему автентифікації та авторизації користувачів на основі Identity та JWT токенів. Використання JWT токенів забезпечує перевірку автентичності користувача. Авторизовані користувачі мають доступ до усіх функцій веб-

застосунку. Авторизація є обов'язковою, адже система розроблена з прив'язкою одягу та образів до акаунту користувача.

4.3 Реалізація клієнтської частини

До клієнтської частини можна віднести файли з директорії Pages та директорію wwwroot. Pages має в собі cshtml файли з HTML кодом для розмітки сторінки та вставки C# коду для відображення даних з моделі сторінки. Директорія wwwroot зберігає CSS стилі та JS код, зображення з інтерфейсу сторінок. Основні сторінки веб-застосунку:

1. Index - лендінг сторінка.
2. UserCloset - відображення збережених користувачем елементів одягу.
3. ClothingImageUpload - завантаження зображень одягу
4. ClothingMatch - створення образів.
5. OutfitCloset - перегляд збережених образів.

Нижче приведено екранні форми деяких з приведених вище сторінок веб-застосунку.

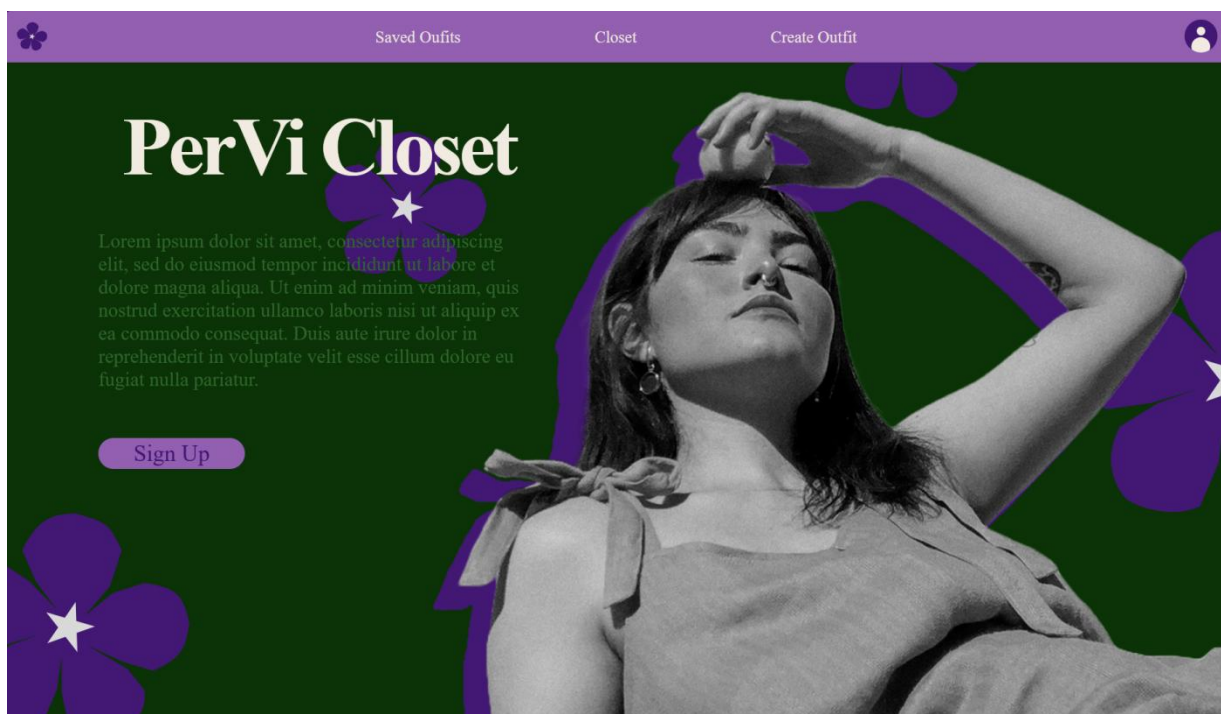


Рис. 4.3 Головна сторінка

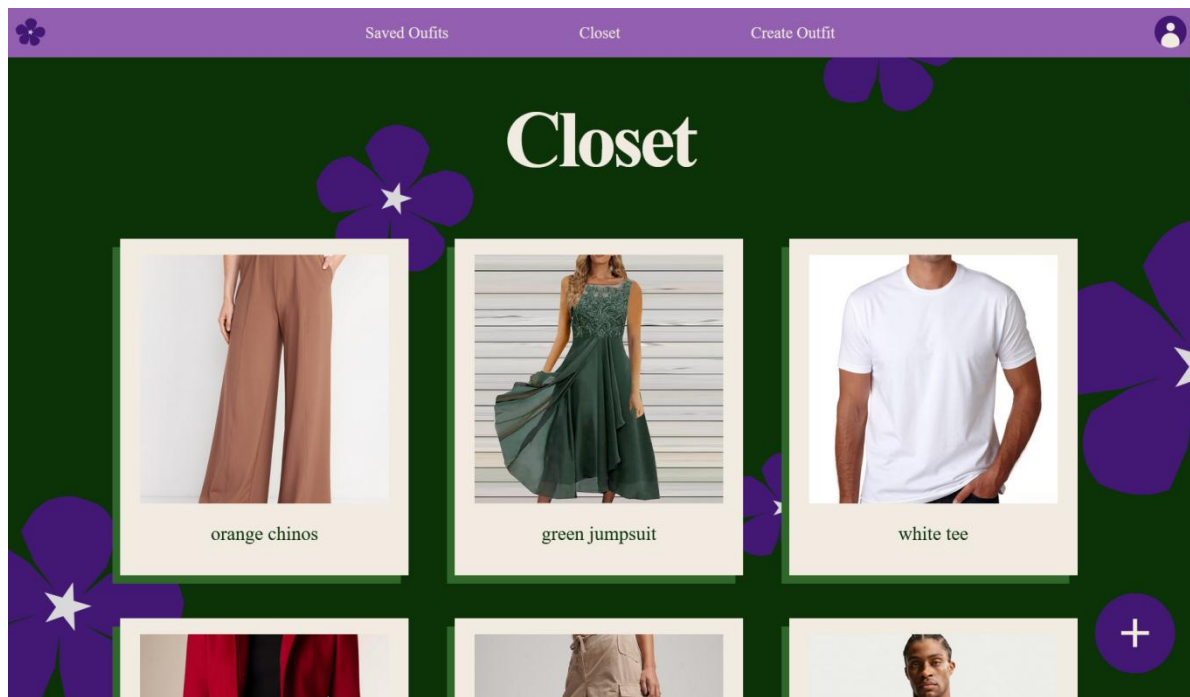


Рис. 4.4 Гардероб користувача

Для створення інтерфейсу кожної сторінки використовувалися: HTML, CSS та JavaScript. Стили сторінок збережені в файлах `app_style` та `page_style`, які відповідають за загальні елементи веб-застосунку та за більш детальні [20].

Під загальними елементами маються на увазі: кольори, шрифти, вигляд кнопок, стиль заголовків та посилань тощо. Також у файлі `app_style` збережено загальні змінні з шрифтами, розмірами шрифтів та кольорами веб-застосунку. Файл `page_style` включає селектори для окремих сторінок, які не повторюються. Також важливим є файл `site.js`, який включає в себе всі використані JavaScript функції за допомогою яких було додано анімації та модальні вікна.

Інтерфейс веб-застосунку був реалізований з урахуванням принципів адаптивного дизайну, що забезпечує коректне відображення сторінок на різних типах пристроїв, зокрема персональних комп'ютерах, планшетах та смартфонах. Для цього використовувалися CSS `flexbox` та `media queries`, які дозволяють автоматично змінювати розташування та розміри елементів інтерфейсу залежно від роздільної здатності екрана.

Для взаємодії клієнтської та серверної частин використовувалися Razor Pages, які поєднують HTML розмітку та C# код у межах однієї сторінки. Такий

підхід дозволяє динамічно відображати дані користувача, передавати інформацію між формами та сервером, а також обробляти дії користувача без необхідності створення окремого API. Кожна сторінка має відповідний PageModel клас, який виконує обробку HTTP-запитів та підготовку даних для відображення.

4.4 Інтеграція та взаємодія компонентів

Після створення моделі розпізнавання зображень, деяких сторінок та функціоналу було здійснено інтеграцію моделі з веб-застосунком та написання всіх пов'язаних з розпізнаванням зображень функцій і класів.

Загальний принцип роботи всієї системи можна описати так:

1. Користувач взаємодіє з веб-застосунком через інтерфейс та надсилає HTTP-запити.
2. Надісланий запит обробляється в моделі сторінки за допомогою різних функцій.
3. При необхідності моделі сторінки викликає функції з сервісів або зробить запит до БД.
4. Результат повертається моделі, а модель передає його в розмітку сторінки відображаючи результат користувачу.

Такий підхід дозволяє розділити логіку інтерфейсу, обробки даних та роботи з базою даних між різними частинами веб-застосунку. Моделі сторінок виступають проміжною ланкою між користувацьким інтерфейсом та серверною частиною застосунку і відповідають за обробку та перевірку даних користувача.

Найчастіше між собою взаємодіють сторінки веб-застосунку, моделі сторінок та база даних, оскільки основна частина функціоналу системи пов'язана зі створенням, редагуванням та переглядом елементів одягу й стилістичних образів.

Важливу роль у взаємодії компонентів також займає інтеграція моделі розпізнавання зображень. Принцип її роботи полягає у тому, що після

завантаження користувачем фотографії одягу це зображення передається до сервісу ClothingPredictor, який виконує попередню обробку зображення та запускає ONNX-модель через ONNX Runtime. Результатом роботи моделі є визначення типу одягу та його основного кольору, що потім використовується для автоматичного заповнення інформації про елемент одягу.

Також невід’ємною частиною інтеграції компонентів є взаємодія із БД. Для роботи з даними використовувався Entity Framework Core, який забезпечує мапінг між класами C# та таблицями SQL. Завдяки цьому сервер може працювати з даними у вигляді об’єктів, що спрощує створення функціоналу та покращує читабельність коду.

Більшість операцій взаємодії між компонентами реалізовано з використанням асинхронних методів `async` та `await`. Це дозволяє уникнути блокування основного потоку виконання під час роботи з базою даних, обробки зображень або взаємодії з моделлю розпізнавання. Використання асинхронного програмування покращує продуктивність веб-застосунку та забезпечує швидшу обробку запитів користувачів.

Під час взаємодії компонентів також реалізовано перевірку коректності даних та обробку можливих помилок. Зокрема, перевіряється формат і розмір завантажених зображень, валідність даних користувача та результати роботи моделі розпізнавання. У випадку виникнення помилки користувачу відображається відповідне повідомлення, що дозволяє уникнути некоректної роботи системи та покращує взаємодію з веб-застосунком.

ВИСНОВКИ

У процесі виконання кваліфікаційної бакалаврської роботи було створено веб-застосунок для формування персоналізованих стилістичних образів. Для створення застосунку було виконано декілька різних етапів роботи, результати яких були використанні в розробці.

Першочергово було проведено аналіз аналогів для поглиблення розуміння галузі. З отриманих результатів було створено вимоги до функціоналу веб-застосунку та нефункціональні вимоги. Крім того, було визначено основну проблему аналогів - нестабільність результатів створення образів через використання лише III методів підбору одягу.

Наступним етапом був аналіз існуючих методів розпізнавання зображень та їх використання у створенні веб-застосунків. Результатом аналізу став вибір використання готової згорткової нейронної мережі з додатковим навчанням на наборі зображень одягу та експорт нейронної мережі в формат ONNX моделі, яка легко інтегрується з веб-застосунками.

Також було проведено аналіз способів створення стилістичних образів для автоматизації процесу формування персоналізованих образів. Після аналізу були створенні правила кольорових поєднань та комбінацій одягу, які відповідають п'яти стилям: формальному, вуличному, повсякденному, елегантному та вечірньому.

Було проаналізовано та обрано засоби програмної реалізації. Основою веб-застосунку стала мова C# та фреймворк Razor Pages, а для тренування нейронної мережі та її експорту були використані Python та його бібліотеки. Крім того, було обрано клієнт-серверну архітектуру, як основу веб-застосунку.

В процесі роботи було натреновано згорткову нейронну мережу на розпізнавання одягу за характеристиками категорії, підкатегорії та типу одягу і за характеристиками основного кольору елементу одягу.

Після всіх описаних процесів було реалізовано й функціонал застосунку, його клієнтську та серверну частину та інтеграція їх між собою. Проект реалізував всі поставлені завдання: аналіз методів розпізнавання зображень та способів створення стилістичних образів, вибір засобів реалізації, створення моделі розпізнавання зображень та програмної реалізації веб-застосунку.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Казак В.Ю., Ярошевський О.В. Інструменти та методи авторизації у веб-розробці. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, С. 215-216.

2. Казак В.Ю., Ярошевський О.В. Використання ONNX моделей для розпізнавання зображень у веб-розробці. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. К.: ДУІКТ, 2025. С.526-528.

ПЕРЕЛІК ПОСИЛАНЬ

1. Shirkhani S., Mokayed H., Saini R., H.Y. Chai. Study of AI-Driven Fashion Recommender Systems. *SN Computer Science*. 2023. Vol.4. no.514. URL: <https://doi.org/10.1007/s42979-023-01932-9> (дата звернення 02.05.2026).
2. Chakraborty S., Md. Hoque S., Jeem N.R., Biswas M.C., Bardhan D., Lobaton E. Fashion recommendation systems, models and methods: a review. *Informatics*. 2021. Vol.8, no.3. P.49. URL: <https://doi.org/10.3390/informatics8030049> (дата звернення 02.05.2026).
3. Diaz-Guerra F., Jimenez-Molina A. Continuous Prediction of Web User Visual Attention on short span Windows based on Gaze Data Analytics. *Sensors*. 2023. Vol.23, no.4, P.2294 URL: <https://www.mdpi.com/1424-8220/23/4/2294> (дата звернення 03.05.2026).
4. Khan S. Advancing Mobile UI Testing by Learning Screen Usage Semantics. 2025. URL: <https://doi.org/10.48550/arXiv.2505.09894> (дата звернення 10.05.2026).
5. Screen Resolution Stats Worldwide. *Statcounter*. 2026. URL: <https://gs.statcounter.com/screen-resolution-stats/> (дата звернення 03.05.2026).
6. Visual Studio documentation. *Microsoft Learn*. URL: <https://learn.microsoft.com/en-us/visualstudio/windows/> (дата звернення 04.05.2026).
7. C# language documentation. *Microsoft Learn*. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата звернення 04.05.2026).
8. Razor Pages architecture and concepts in ASP.NET Core. *Microsoft Learn*. 2025. URL: <https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-10.0&tabs=visual-studio> (дата звернення 05.05.2026).
9. Entity Framework Core. *Microsoft Learn*. 2024. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення 05.05.2026).

10. ONNX Runtime and Models. *Microsoft Learn*. URL: <https://learn.microsoft.com/en-us/azure/machine-learning/concept-onnx?view=azureml-api-2> (дата звернення 06.05.2026).
11. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Neural Information Processing System*. 2019. URL: <https://www.semanticscholar.org/paper/PyTorch%3A-An-Imperative-Style%2C-High-Performance-Deep-Paszke-Gross/3c8a456509e6c0805354bd40a35e3f2dbf8069b1#cited-papers> (дата звернення 06.05.2026).
12. pandas documentation. pandas. 2026. URL: <https://pandas.pydata.org/docs/index.html> (дата звернення 07.05.2026).
13. GitHub Docs. *GitHub Docs*. URL: <https://docs.github.com/en> (дата звернення 07.05.2026).
14. Hamid S.A., Abdulrahman R.A., Khamees R.A. What is Client-Server System: Architecture, Issues and Challenge of Client -Server System (Review). *Recent Trends in Cloud Computing and Web Engineering*. 2020. Vol.2. no.1. P.1-6. URL: <https://doi.org/10.5281/zenodo.3673071> (дата звернення 09.05.2026).
15. Architectural Review of Client-Server Models. *International Journal of Advanced Science and Computer Applications*. 2024. Vol.3. no.2. P.139-143. URL: <https://doi.org/10.47679/ijasca.v3i1.48> (дата звернення 09.05.2026).
16. Qu Q.X., Zhang L., Guo F., Duffy V.G. Reducing UI Complexity Using Use Case Analysis in Adaptive Interfaces. *Computers, Materials & Continua*. 2025. Vol.85. no.3. P.4607-4627. URL: <https://doi.org/10.32604/cmc.2025.069245> (дата звернення 10.05.2026).
17. Яшина, О. Методи забезпечення безпеки для розробки вебзастосунків в ASP.NET CORE. *Herald of Khmelnytskyi National University*.

Technical Sciences, 2026. 361(1). 496-500. URL: <https://doi.org/10.31891/2307-5732-2026-361-69> (дата звернення 09.05.2026).

18. Bhalani, V. Advancements in Computer Vision: Techniques, Applications, and Future Trends. *International Journal of Computational Science and Engineering Research*. 2024. Vol. 14. URL: https://www.researchgate.net/publication/404472205_Advancements_in_Computer_Vision_Techniques_Applications_and_Future_Trends (дата звернення 08.05.2026).

19. Schindler A., Lidy T., Karner S., Hecker M. Fashion and Apparel Classification using Convolutional Neural Networks. *10th Forum Media Technology and 3rd All Around Audio Symposium*, November 29-30, 2017, St. Poelten, Austria. URL: <https://ui.adsabs.harvard.edu/abs/2018arXiv181104374S/abstract> (дата звернення 08.05.2026).

20. Jain R., Shrivastava V., Pandey A., Sharma A. Modern Web Development using CSS & HTML. *International Journal of Emerging Science and Engineering*. 2024. Vol.12. no.6. P.13-16. URL: <https://www.ijese.org/portfolio-item/G257412070624/>

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



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



Розробка web-застосунку для формування персоналізованих стилістичних образів

Виконала студентка 4 курсу
групи ПД-44
Казак Вероніка Юріївна
Керівник роботи
асист. Ярошевський Олександр Вікторович

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи спрощення процесу формування персоналізованих стилістичних образів за рахунок автоматизації підбору одягу.

Об'єкт дослідження рекомендаційні алгоритми для формування персоналізованих стилістичних образів та робота з моделями машинного зору.

Предмет дослідження технології розробки веб-додатків та алгоритми формування персоналізованих стилістичних образів.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Огляд та аналіз існуючих методів та технологій розпізнавання зображень в контексті задачі підбору одягу.
2. Огляд та аналіз способів та принципів створення образів для автоматизації процесу формування персоналізованих стилістичних образів.
3. Вибір засобів програмної реалізації веб-застосунку та моделі розпізнавання зображень одягу.
4. Створення моделі розпізнавання зображень одягу для використання в веб-застосунку.
5. Проєктування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».
6. Програмна реалізація та тестування веб-застосунку для створення персоналізованих стилістичних образів «PerVi Closet».

3

АНАЛІЗ АНАЛОГІВ

Критерій	Closet.Style	AI Closet	Closet OS	Aura	PerVi Closet
Заповнення даних про одяг	Повністю автоматично	Повністю автоматично	Частково пишеться користувачем	Частково пишеться користувачем	Повністю автоматично
Збереження створених образів	Є	Є	Є	Немає	Є
Додавання аксесуарів в образ	Немає	Є	Є	Є	Є
Спосіб створення образу	Вручну або ШІ запит	Вручну або ШІ запит	ШІ запит	ШІ запит	Система правил
Основа створення образу	Завантажений користувачем одяг	Завантажений користувачем одяг	Завантажений користувачем одяг	Користувацький та системний одяг	Завантажений користувачем одяг

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

1. Реєстрація та авторизація користувача.
2. Завантаження фотографій одягу користувача.
3. Розпізнавання фотографій одягу та заповнення інформації про нього.
4. CRUD операції з одягом.
5. Перегляд списку завантаженого одягу.
6. Підбір образу за стилем.
7. Підбір образу з урахуванням теорії кольору.
8. Збереження створених образів.
9. Перегляд збережених образів.

Нефункціональні вимоги

1. Час відгуку системи не більше 3 сек.
2. Підтримка браузерів: Chrome, Firefox та Safari.
3. Адаптація інтерфейсу для розширень екранів 1920×1080 та 1366×768, 360×800 та 393×852.
4. Захист персональних даних використанням PBKDF2 хешування та JWT токенів.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Створення моделі для розпізнавання зображень

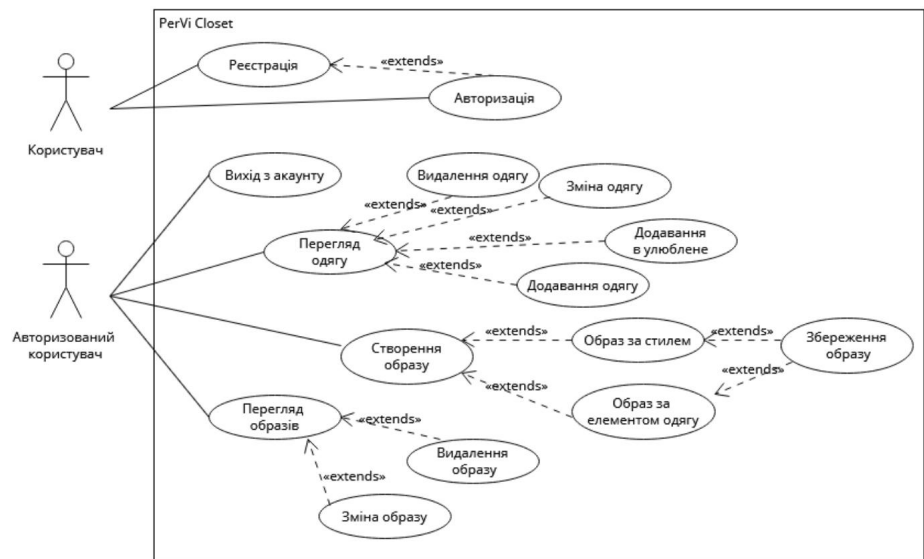


Створення веб-додатку



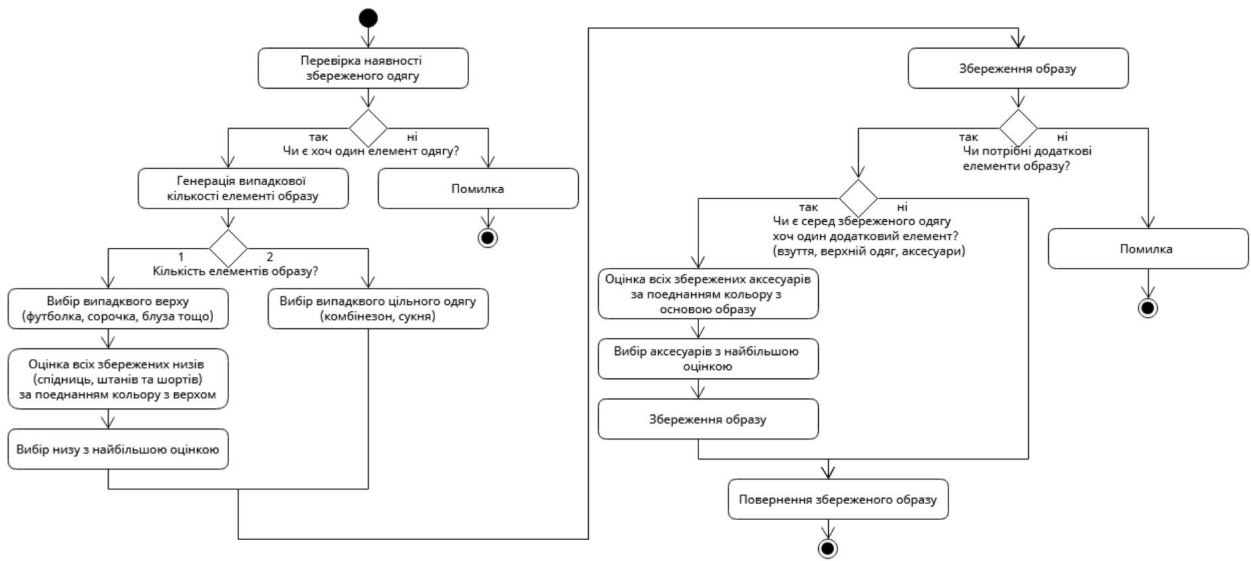
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



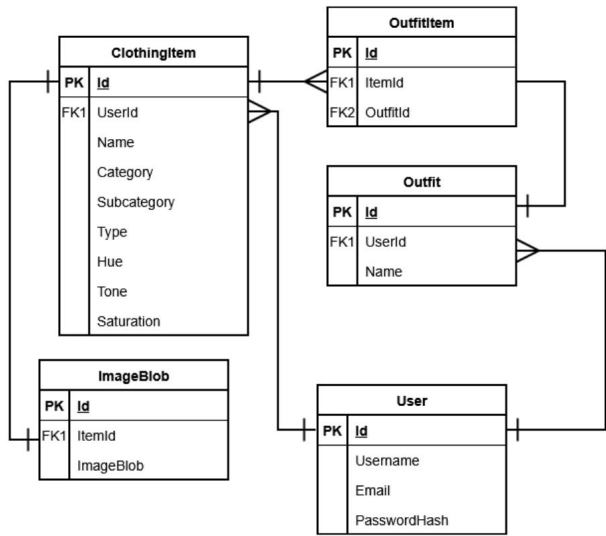
7

ДІАГРАМА ДІЯЛЬНОСТІ ПІДБОРУ ОБРАЗУ

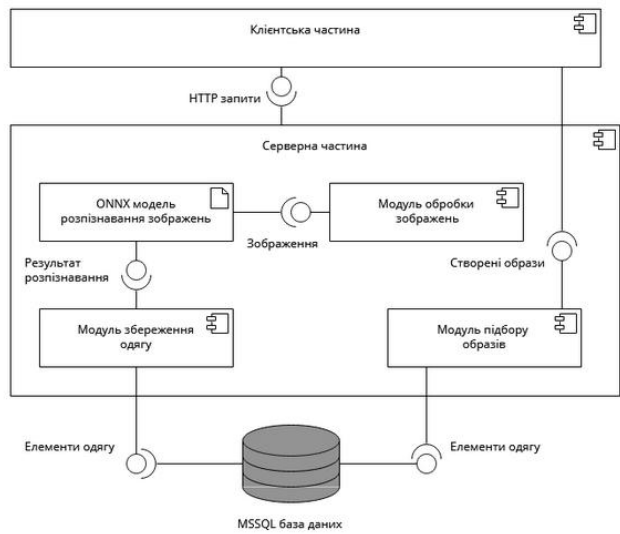


8

ER ДІАГРАМА БАЗИ ДАНИХ



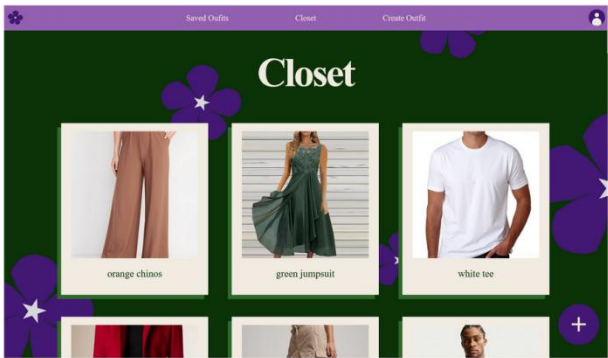
ДІАГРАМА КОМПОНЕНТІВ



ЕКРАННІ ФОРМИ

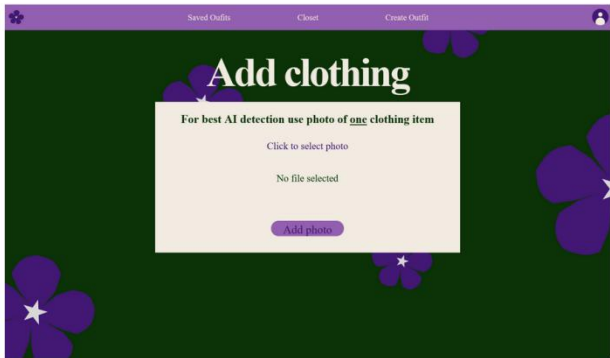


Лендінг сторінка

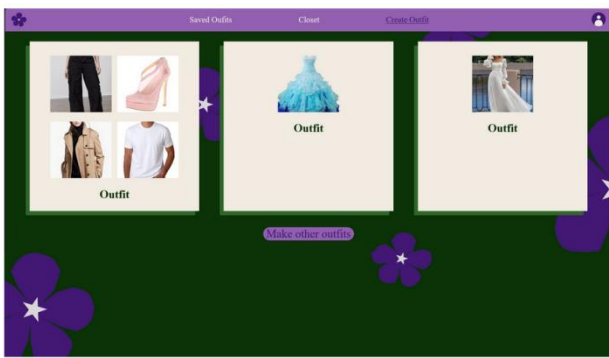


Список одягу користувача

ЕКРАННІ ФОРМИ



Сторінка додавання фото



Список створених образів

ВІДЕО РОБОТИ



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Казак В.Ю., Ярошевський О.В. Інструменти та методи авторизації у веб-розробці. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, С. 215-216.
2. Казак В.Ю., Ярошевський О.В. Використання ONNX моделей для розпізнавання зображень у веб-розробці. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. К.: ДУІКТ, С. 217-218.

14

ВИСНОВКИ

1. Було проведено аналіз аналогів. З отриманих результатів було створено вимоги до веб-застосунку. Крім того, було визначено основну проблему аналогів - нестабільність результатів створення образів через використання лише ШІ методів підбору.
2. Було проведено аналіз існуючих методів розпізнавання зображень та їх використання у створенні веб-застосунків. Результатом аналізу став вибір використання готової згорткової нейронної з додатковим навчанням експорт мережі в формат ONNX моделі для використання з веб-додатком.
3. Було проаналізовано та обрано засоби програмної реалізації. Основою веб-застосунку стала мова C# та фреймворк Razor Pages, а для тренування нейронної мережі та її експорту були використані Python та його бібліотеки.
4. Було проведено додаткове тренування згорткової нейронної мережі на розпізнавання одягу за характеристиками категорії, підкатегорії та типу одягу і за характеристиками основного кольору елементу одягу.
5. Було реалізовано функціонал застосунку, його клієнтську та серверну частину та проведена інтеграція частин між собою.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

ClothingUpload.cshtml

```

@page
@model
PerViCloset_pre_final_.Pages.LoggedIn.Clothing.ClothingImageUploadModel
@{
    ViewData["Title"] = "Add Item";
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"]</title>
    <script type="importmap"></script>
    <link rel="stylesheet" href="~/css/test_style.css" />
</head>
<body>
    <div class="content centered">
        <div class="column-content">
            <h1>Add clothing</h1>
            <div class="card-gen long" id="add-card">
                <h3>For best AI detection use photo of <span
style="text-decoration:underline">one</span> clothing
item</h3>
                <form method="post"
enctype="multipart/form-data" class="content centered
column-content">
                    <label for="file-upload" class="text
clickable">Click to select photo</label>
                    <input id="file-upload" type="file"
name="File" />
                    <p class="text dark-green" id="file-
name">No file selected</p>

```

```

        @Html.ValidationMessageFor(model =>
model.userImage.File, "", new { @class = "text-
danger" })
        <button class="btn clickable" type="submit"
asp-page="/Item">
            Add photo
        </button>
    </form>
</div>
</div>
</div>
</body>

<script>
    const fileInput = document.getElementById("file-
upload");
    const fileName = document.getElementById("file-
name");

    fileInput.addEventListener("change", function () {
        if (this.files.length > 0) {
            fileName.textContent = this.files[0].name;
        }
    });
</script>
</html>

```

ImageService.cs

```

using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

namespace
PerViCloset_pre_final_.Services.ClothingPrediction
{
    public static class ImageService
    {

```

```
public static float[] LoadImage(string path)
{
    using var image = Image.Load<Rgb24>(path);
    image.Mutate(x => x.Resize(new ResizeOptions
    {
        Size = new Size(224, 224),
        Mode = ResizeMode.Crop
    })); ;

    float[] data = new float[3 * 224 * 224];
    int channelSize = 224 * 224;

    for (int y = 0; y < 224; y++)
    {
        for (int x = 0; x < 224; x++)
        {
            var p = image[x, y];

            int idx = y * 224 + x;

            data[idx] = p.R / 255f;
            data[channelSize + idx] = p.G / 255f;
            data[2 * channelSize + idx] = p.B / 255f;
        }
    }
    return data;
}

public static bool
IsImageExtensionAllowed(IFormFile file)
{
    string[] allowedExtensions = [".png", ".jpg"];

    var extension =
Path.GetExtension(file.FileName);
    return allowedExtensions.Contains(extension);
}

public static bool IsFileSizeWithinLimit(IFormFile
file)
{
    return file.Length <= 5242880;
}
```

OutfitMaker.cs

```
using PerViCloset_pre_final_.Models;
using PerViCloset_pre_final_.Models.ClothingModels;
using PerViCloset_pre_final_.Models.Enums;
using
PerViCloset_pre_final_.Models.Enums.Clothing_Emums;
using PerViCloset_pre_final_.Models.OutfitModels;

namespace
PerViCloset_pre_final_.Services.ClotingMatching
{
    public class OutfitMaker
    {
        public OutfitMakerResult
GenerateOutfit(List<ClothingItem> userCloset, bool
withShoes, bool withOuterwear)
        {
            if (!ValidateExtraCloset(userCloset))
            {
                return new OutfitMakerResult
                {
                    Success = false,
                    ErrorMessage = "Not enough extra items to
add to an outfit."
                };
            }

            var outfit = GenerateBasicOutfit(userCloset);
            if (outfit.Success == false)
                return outfit;

            OutfitItem baseItem;

            if (outfit.Outfit.TopItem != null)
                baseItem = outfit.Outfit.TopItem;
            else
                baseItem = outfit.Outfit.OnePieceItem;
```

```

        if (withShoes)
        {
            outfit.Outfit.Shoes =
AddExtraItem(ClothingCategory.shoes, userCloset,
baseItem.GetFullColorInfo(), outfit.Outfit.Id);
        }

        if (withOuterwear)
        {
            outfit.Outfit.OuterwearItem =
AddExtraItem(ClothingCategory.outerwear, userCloset,
baseItem.GetFullColorInfo(), outfit.Outfit.Id);
        }

        return outfit;
    }

    public OutfitMakerResult
GenerateBasicOutfit(List<ClothingItem> userCloset)
    {
        if (!ValidateBasicCloset(userCloset))
        {
            return new OutfitMakerResult
            {
                Success = false,
                ErrorMessage = "Not enough clothing items
to create an outfit."
            };
        }

        var random = new Random();
        var outfitPieceCount = random.Next(1, 3);

        if (outfitPieceCount == 1)
            return GeneratOnePieceOutfit(userCloset);

        return GenerateTwoPieceOutfit(userCloset);
    }

    public OutfitMakerResult
GenerateTwoPieceOutfit(List<ClothingItem> userCloset)

```

```

    {
        var tops = userCloset.Where(i => i.Category ==
ClothingCategory.top).ToList();
        var bottoms = userCloset.Where(i => i.Category
== ClothingCategory.bottom).ToList();

        var baseItem = tops.Shuffle().FirstOrDefault();

        var best = FindBestMatch(bottoms,
baseItem.GetFullColorInfo());

        var newOutfit = new Outfit(new
List<ClothingItem> { baseItem, best });
        var result = new OutfitMakerResult { Success =
true, Outfit = newOutfit };
        return result;
    }

    public OutfitMakerResult
GeneratOnePieceOutfit(List<ClothingItem> userCloset)
    {
        var onepiece = userCloset.Where(i => i.Category
== ClothingCategory.onepiece).ToList();
        var baseItem =
onepiece.Shuffle().FirstOrDefault();

        var newOutfit = new Outfit(new
List<ClothingItem> { baseItem });
        var result = new OutfitMakerResult { Success =
true, Outfit = newOutfit };
        return result;
    }

    public OutfitItem AddExtraItem(ClothingCategory
category, List<ClothingItem> userCloset, Color
baseItemColor, Guid outfitId)
    {
        var items = userCloset.Where(i => i.Category ==
ClothingCategory.shoes).ToList();
        var best = FindBestMatch(items, baseItemColor);

        return new OutfitItem(best, outfitId);
    }

```

```

    }

    // Check if closet has items to make basic outfit of 1
or 2 items
    public static bool
ValidateBasicCloset(List<ClothingItem> userCloset)
    {
        if (userCloset == null || userCloset.Count == 0)
            return false;

        bool hasTop = userCloset.Any(i =>
            i.Category == ClothingCategory.top);

        bool hasBottom = userCloset.Any(i =>
            i.Category == ClothingCategory.bottom);

        bool hasOnePiece = userCloset.Any(i =>
            i.Category == ClothingCategory.onepiece);

        return (hasTop && hasBottom) || hasOnePiece;
    }

    // Check if closet has items to add to a basic outfit
    // (aka shoes, outerwear, accessories)
    public static bool
ValidateExtraCloset(List<ClothingItem> userCloset)
    {
        if (userCloset == null || userCloset.Count == 0)
            return false;

        bool hasTop = userCloset.Any(i =>
            i.Category == ClothingCategory.top);

        bool hasBottom = userCloset.Any(i =>
            i.Category == ClothingCategory.bottom);

        bool hasOnePiece = userCloset.Any(i =>
            i.Category == ClothingCategory.onepiece);

        return (hasTop && hasBottom) || hasOnePiece;
    }

```

```

    public static bool
EmptyCheck(List<ClothingItem> list)
    {
        if (list.Count() == 0 || list == null)
            return true; // is empty

        return false;
    }

    public static ClothingItem
FindBestMatch(List<ClothingItem> list, Color
baseItemColor)
    {
        var best = list
            .OrderByDescending(item =>
                baseItemColor.ScoreColor(item.GetFullColorInfo()))
            .FirstOrDefault();

        return best;
    }
}

Color.cs
using PerViCloset_pre_final_.Models.Enums;

namespace PerViCloset_pre_final_.Models
{
    public class Color
    {
        public Hue Hue { get; set; }
        public Tone Tone { get; set; }
        public Saturation Saturation { get; set; }

        public Color(Hue hue, Tone tone, Saturation
saturation)
        {
            Hue = hue;
            Tone = tone;
            Saturation = saturation;
        }
    }
}

```

```

public Color()
{

}

private bool IsNeutral(Hue hue)
{
    switch (Hue)
    {
        case (Hue.white):
            return true;

        case (Hue.gray):
            return true;

        case (Hue.black):
            return true;

        case (Hue.brown):
            return true;

        default:
            return false;
    }
}

private int ScoreToneCompatibility(Tone tone2)
{
    int diff = Math.Abs((int)Tone - (int)tone2);

    return diff switch
    {
        0 => 3, // 3 points form same tone
        1 => 2, // 2 points for light->mid mid->dark etc
        2 => 1, // 1 point for light->dark
        _ => 0 // 0 for other
    };
}

private int ScoreSaturationCompatibility(Saturation
saturation2)

{
    int diff = Math.Abs((int)Saturation -
(int)saturation2);

    return diff switch
    {
        2 => 3, // 3 points for saturated -> muted
        1 => 2, // 2 points for saturated->mid mid-
>muted etc
        0 => 1, // 1 point form same saturation
        _ => 0 // 0 for other
    };
}

private int ScoreHueCompatibility(Hue hue2)
{
    if (IsAnalogous(Hue, hue2))
        return 10;
    // analogous colors look best together

    if (Hue == hue2)
        return 8;
    // same colors look good together

    if (IsNeutral(Hue) || IsNeutral(hue2))
        return 7;
    //neutral + neural is also good

    if (IsComplementary(Hue, hue2))
        return 7;
    // complementary is still good

    if (IsTriad(Hue, hue2))
        return 6;
    //triad is ok

    return 1;
    //mismatch is bad
}

public int ScoreColor(Color color2)
{

```

```

int score = 0;

score += ScoreHueCompatibility(color2.Hue);
score += ScoreToneCompatibility(color2.Tone);
score +=
ScoreSaturationCompatibility(color2.Saturation);

return score;
}

private bool IsAnalogous(Hue hue1, Hue hue2)
{
    return AnalogousMap.ContainsKey(hue1) &&
AnalogousMap[hue1].Contains(hue2);
}

private bool IsTriad(Hue hue1, Hue hue2)
{
    return TriadMap.ContainsKey(hue1) &&
TriadMap[hue1].Contains(hue2);
}

private bool IsComplementary(Hue hue1, Hue hue2)
{
    return ComplementaryMap.ContainsKey(hue1)
&& ComplementaryMap[hue1] == hue2;
}

public static readonly Dictionary<Hue, List<Hue>>
AnalogousMap = new()
{
    { Hue.red, new List<Hue> { Hue.orange,
Hue.pink } },
    { Hue.orange, new List<Hue> { Hue.red,
Hue.yellow } },
    { Hue.yellow, new List<Hue> { Hue.orange,
Hue.green } },
    { Hue.green, new List<Hue> { Hue.yellow,
Hue.blue } },

```

```

    { Hue.blue, new List<Hue> { Hue.green,
Hue.purple } },
    { Hue.purple, new List<Hue> { Hue.blue,
Hue.pink } },
    { Hue.pink, new List<Hue> { Hue.purple,
Hue.red } }
};

public static readonly Dictionary<Hue, List<Hue>>
TriadMap = new()
{
    { Hue.red, new List<Hue> { Hue.yellow,
Hue.blue } },
    { Hue.orange, new List<Hue> { Hue.purple,
Hue.green } },
    { Hue.yellow, new List<Hue> { Hue.red,
Hue.blue } },
    { Hue.green, new List<Hue> { Hue.purple,
Hue.orange } },
    { Hue.blue, new List<Hue> { Hue.red,
Hue.yellow } },
    { Hue.purple, new List<Hue> { Hue.orange,
Hue.green } }
};

public static readonly Dictionary<Hue, Hue>
ComplementaryMap = new()
{
    { Hue.red, Hue.green },
    { Hue.orange, Hue.blue },
    { Hue.yellow, Hue.purple },
    { Hue.green, Hue.red },
    { Hue.blue, Hue.orange },
    { Hue.purple, Hue.yellow },
    { Hue.pink, Hue.green }
};
}
}

```