Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему **КОМП'ЮТЕРНА СИСТЕМА ДЛЯ ОРГАНІЗАЦІЇ ТА ЗБЕРІГАННЯ АРХІВІВ ФОТОГРАФІЙ**

Виконав: студент 2 курсу, групи КНТз-514м
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні системи та мережі

ЗІМІН О.А.
(ПРІЗВИЩЕ та ініціали)
Керівник СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)
Рецензент РОМАНОВСЬКИЙ О.В.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні системи та мережі
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Зав. кафедри Кудерметов Р.К.

"15" жовтня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ЗІМІНА Олександра Андрійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система для організації та зберігання архівів фотографій
керівник проєкту (роботи) кан. техн. наук, доцент, СКРУПСЬКИЙ Степан Юрійович
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від " 30"жовтня 2025 року № 491
2. Строк подання студентом проєкту (роботи) 23 грудня 2025 року
3. Вихідні дані до проєкту (роботи) наявні методи організації та індексації фотоархівів, технології автоматичного розпізнавання об'єктів на зображеннях, стратегії резервного копіювання даних, швидкість індексації не менше 50 фото/секунду, час відгуку пошукових запитів не більше 1 секунди, точність розпізнавання об'єктів не менше 80%

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз методів організації цифрових фотоархівів, дослідження технологій обробки метаданих EXIF, вивчення підходів до автоматичного розпізнавання контенту, порівняння стратегій резервного копіювання;
2) Розробка архітектури системи з мікросервісним підходом, проектування структури бази даних, моделювання бізнес-процесів;
3) Реалізація модулів завантаження, індексації, пошуку та резервного копіювання, інтеграція моделей машинного навчання для розпізнавання об'єктів, розробка веб-інтерфейсу;
4) Експериментальне дослідження продуктивності системи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ С.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз методів організації фотоархівів, обробки	05.10.2025 р.	
	метаданих, автоматичного розпізнавання, виділення		
	недоліків та переваг		
2	Розробка структури системи	10.10.2025 р.	
3	Проектування структури бази даних	25.10.2025 р.	
4	Реалізація модулів системи	05.11.2025 р.	
5	Дослідження системи	30.11.2025 р.	
6	Оформлення отриманих результатів у ПЗ	05.12.2025 р.	
7	Оформлення графічного матеріалу	10.12.2025 р.	
8	Оформлення допоміжного матеріалу	15.12.2025 р.	

Студент _____ **Олександр ЗІМІН**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Степан СКРУПСЬКИЙ**
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 111 стор., 21 рис., 27 табл., 21 джерел.

YOLOV8, ELASTICSEARCH, MILVUS, NODE.JS, POSTGRESQL, REACT, АВТОМАТИЧНА ІНДЕКСАЦІЯ, ВЕКТОРНИЙ ПОШУК, КОМП'ЮТЕРНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, МЕТАДАНІ EXIF, ОРГАНІЗАЦІЯ ФОТОАРХІВІВ, ПОШУК ЗОБРАЖЕНЬ, РЕЗЕРВНЕ КОПІЮВАННЯ, РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

Об'єкт дослідження - процеси організації, зберігання та пошуку цифрових фотографій в архівних системах великого обсягу.

Предмет дослідження - методи та алгоритми автоматичної індексації метаданих зображень.

Мета роботи - підвищення ефективності управління великими фотоархівами з підтримкою автоматичної індексації та інтелектуального пошуку.

У першому розділі проведено комплексний аналіз існуючих підходів до організації цифрових фотоархівів, досліджено методи обробки метаданих формату EXIF, виконано порівняльний аналіз хмарних сервісів, визначено технології автоматичного розпізнавання контенту та стратегії резервного копіювання.

У другому розділі розроблено концептуальну модель системи з тришаровою архітектурою та мікросервісним підходом, спроектовано архітектуру компонентів на базі React-Node.js-PostgreSQL стеку.

Третій розділ присвячено реалізації системи. Описано характеристики апаратного та програмного забезпечення розробки.

У четвертому розділі визначено методику експериментального дослідження з підготовкою тестових датасетів, проведено дослідження продуктивності індексації.

ABSTRACT

Explanatory note to the master's work: 111 p., 21 fig., 27 tabl., 21 sources.

AUTOMATIC INDEXING, BACKUP STRATEGY, COMPUTER SYSTEM, ELASTICSEARCH, EXIF METADATA, IMAGE SEARCH, MACHINE LEARNING, MILVUS, NODE.JS, OBJECT RECOGNITION, PHOTO ARCHIVE ORGANIZATION, POSTGRESQL, REACT, VECTOR SEARCH, YOLOV8

Research object - processes of organization, storage and search of digital photographs in large-scale archival systems.

Research subject - methods and algorithms for automatic indexing of image metadata, intelligent search technologies based on machine learning, strategies for reliable multi-level data backup.

Research goal is to enhance the efficiency of large photo archive management through automated indexing and intelligent search capabilities..

The first section conducts comprehensive analysis of existing approaches to digital photo archive organization, examines EXIF metadata processing methods, performs comparative analysis of cloud services.

The second section develops conceptual model of the system with three-tier architecture and microservices approach, designs component architecture based on React-Node.js-PostgreSQL stack, creates relational database structure with PostGIS support for geographic data, develops photo indexing algorithms with multi-level preview generation in WebP format, justifies technology stack selection for the system.

The third section is dedicated to system implementation. Hardware and software development environment characteristics are described.

The fourth section defines experimental research methodology with test dataset preparation.

ЗМІСТ

ВСТУП

Сучасне суспільство характеризується безпрецедентним зростанням обсягів цифрового контенту, особливо фотографічних матеріалів. За оцінками аналітичної компанії InfoTrends, у 2023 році було створено понад 1.81 трильйона цифрових фотографій, і цей показник продовжує зростати щороку. Поширення смартфонів з високоякісними камерами, доступність професійного фотообладнання та зростання популярності візуального контенту в соціальних мережах призвели до того, що середній користувач створює сотні нових зображень щомісяця.

Проте інтенсивне зростання обсягів фотоконтенту породжує значні виклики в організації, зберіганні та пошуку необхідних зображень. Дослідження показують, що понад 70% користувачів стикаються з труднощами при спробі знайти конкретну фотографію в своїх архівах, витрачаючи на пошук від декількох хвилин до години часу. Близько 15% користувачів зазнають втрати цінних фотографій через відсутність належного резервного копіювання, несправність носіїв інформації або помилкове видалення файлів. Традиційні підходи до організації фотоархівів через файлову систему з ручним створенням папок за датами чи подіями виявляються неефективними при роботі з десятками тисяч зображень.

Існуючі рішення для управління фотоархівами можна умовно поділити на три категорії. Хмарні сервіси, такі як Google Photos, Apple iCloud Photos, Amazon Photos, пропонують зручність автоматичної синхронізації та доступність з будь-якого пристрою, проте мають суттєві обмеження щодо приватності даних, залежності від інтернет-з'єднання, обмежень безкоштовного сховища та можливої втрати якості зображень через компресію. Настільні програми для організації фотографій, зокрема Adobe Lightroom, Capture One, ACDSee, забезпечують професійні інструменти обробки та каталогізації, але вимагають значних фінансових інвестицій, мають складний інтерфейс для звичайних

користувачів та часто прив'язані до конкретної операційної системи. Локальні файлові менеджери надають повний контроль над даними та незалежність від хмарних сервісів, проте не забезпечують автоматичної індексації метаданих, інтелектуального пошуку за змістом зображень чи надійного багаторівневого резервування.

Актуальність розробки спеціалізованої комп'ютерної системи для організації та зберігання фотоархівів обумовлена необхідністю створення рішення, яке поєднувало б переваги різних підходів, а саме зручність використання хмарних сервісів з повним контролем користувача над власними даними, автоматизацію індексації та інтелектуального пошуку професійних програм із доступністю для широкого кола користувачів, надійність резервного копіювання корпоративних систем з економічною ефективністю для приватного використання.

Метою дипломної роботи є підвищення ефективності управління великими фотоархівами з підтримкою автоматичної індексації та інтелектуального пошуку.

Для досягнення поставленої мети необхідно вирішити такі основні задачі. По-перше, провести аналіз існуючих методів організації цифрових фотоархівів, дослідити підходи до збору та обробки метаданих зображень, вивчити технології автоматичного розпізнавання контенту, порівняти стратегії резервного копіювання та визначити їх переваги й недоліки для обґрунтованого вибору оптимальних рішень. По-друге, розробити архітектуру системи з чітким розділенням на незалежні модулі для завантаження файлів, індексації метаданих, пошуку за різними критеріями, візуалізації результатів та резервного копіювання. Архітектура повинна базуватися на мікросервісному підході з можливістю незалежного масштабування найбільш навантажених компонентів, використовувати черги повідомлень для асинхронної обробки задач та забезпечувати відмовостійкість через реплікацію критичних сервісів. По-третє, спроєктувати структуру бази даних для зберігання метаданих фотографій з підтримкою складних взаємозв'язків між сутностями, оптимізувати індекси для

швидкого пошуку за різними атрибутами, забезпечити масштабованість схеми для роботи з мільйонами записів. По-четверте, реалізувати функціональність автоматичної індексації метаданих з екстракцією EXIF-інформації про параметри зйомки, геолокацію та технічні характеристики обладнання, інтегрувати моделі машинного навчання для розпізнавання об'єктів на зображеннях, створити систему тегування та категоризації для організації контенту. По-п'яте, розробити багаторівневу систему резервного копіювання за правилом 3-2-1 з локальним сховищем, мережевим сховищем та хмарним резервуванням, реалізувати автоматичну верифікацію цілісності резервних копій, забезпечити процедури швидкого відновлення даних у разі збоїв. По-шосте, створити зручний користувацький інтерфейс для завантаження фотографій, перегляду колекцій, здійснення пошуку та управління альбомами з підтримкою адаптивного дизайну для роботи на різних пристроях. По-сьоме, провести експериментальне дослідження продуктивності системи з вимірюванням швидкості індексації, часу відгуку пошукових запитів, навантажувальної здатності при одночасній роботі множини користувачів. По-восьме, виконати юзабіліті-дослідження розробленого інтерфейсу з реальними користувачами різного віку та технічної підготовки для оцінки зручності використання та виявлення можливих проблем взаємодії.

Об'єктом дослідження є процеси організації, зберігання та пошуку цифрових фотографій в архівних системах великого обсягу. Предметом дослідження є методи та алгоритми автоматичної індексації метаданих зображень, технології інтелектуального пошуку на основі машинного навчання, стратегії надійного багаторівневого резервного копіювання фотоархівів.

Методи дослідження включають аналіз науково-технічної літератури та існуючих систем управління фотоархівами, об'єктно-орієнтоване проєктування архітектури програмної системи з використанням UML-діаграм, реляційне моделювання структур даних, застосування алгоритмів обробки зображень для генерації превью та оптимізації розміру файлів, використання моделей

глибокого навчання для розпізнавання об'єктів, експериментальне дослідження продуктивності та юзабіліті розробленої системи.

Наукова новизна роботи полягає в розробці комплексного підходу до організації фотоархівів, який інтегрує автоматичну індексацію метаданих, інтелектуальний пошук на основі нейронних мереж та багаторівневе резервування в єдину систему з мікросервісною архітектурою, що дозволяє забезпечити високу продуктивність та надійність при роботі з великими обсягами даних.

Практична цінність роботи визначається створенням функціональної системи, яка може використовуватися приватними користувачами для організації домашніх фотоархівів, професійними фотографами для управління портфоліо та архівами знімків клієнтів, організаціями та установами для зберігання та систематизації фотодокументації подій та проєктів. Розроблена система забезпечує значну економію часу на пошук потрібних зображень, зменшує ризики втрати цінних фотографій завдяки надійному резервуванню, спрощує організацію та категоризацію великих колекцій через автоматичне тегування.

Структура роботи відповідає послідовності вирішення поставлених задач. Перший розділ присвячено аналітичному дослідженню існуючих методів організації фотоархівів, технологій обробки метаданих, підходів до інтелектуального пошуку та стратегій резервного копіювання. Другий розділ описує концептуальне проєктування системи з розробкою архітектури, моделюванням бізнес-процесів та проєктуванням структури бази даних. Третій розділ деталізує технічну реалізацію основних компонентів системи з описом використаних технологій, алгоритмів та програмних рішень. Четвертий розділ представляє результати експериментального дослідження продуктивності системи та юзабіліті-дослідження користувацького інтерфейсу. У висновках підводяться підсумки виконаної роботи, формулюються основні результати та пропонуються напрямки подальших досліджень.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз методів збору та обробки метаданих цифрових фотографій

Сучасні підходи до організації цифрових фотоархівів можна класифікувати за декількома ключовими аспектами, а саме способом зберігання метаданих, організацією файлової структури, методами індексації контенту та стратегіями резервного копіювання.

Метадані фотографій є критично важливим компонентом будь-якої системи управління архівами, оскільки саме вони забезпечують можливість ефективного пошуку, фільтрування та організації великих колекцій зображень. Формат EXIF став де-факто стандартом для зберігання технічних параметрів цифрової фотографії безпосередньо у файлі зображення. Специфікація EXIF версії 2.32, затверджена Camera & Imaging Products Association у 2019 році, визначає структуровані теги для збереження широкого спектру інформації про умови зйомки [4].

Основні категорії EXIF-даних включають технічні параметри камери та об'єктива з полями для виробника камери, моделі, серійного номера, типу об'єктива та фокусної відстані. Параметри експозиції зберігають інформацію про ISO чутливість, діафрагму, витримку затвору, режим вимірювання експозиції та використання спалаху. Просторово-часові дані містять дату та час зйомки з точністю до секунди, GPS координати широти та довготи для геолокації, висоту над рівнем моря та напрямок зйомки. Характеристики зображення включають роздільність у пікселях, колірний простір, орієнтацію кадру та налаштування балансу білого.

Аналіз наукової літератури виявив, що EXIF-метадані широко використовуються у дослідженнях поведінки фотографів та патернів використання камер. Дослідження на датасеті з 15 мільйонів фотографій Flickr показало, що 78% завантажених зображень містять EXIF-дані, при цьому найчастіше заповненими полями є DateTime у 89% випадків, Make та Model

камери у 76% зображень, параметри експозиції ISO, Aperture, Shutter Speed у 64% фотографій та GPS координати у 31% для фото зі смартфонів проти 8% з дзеркальних камер (табл. 1.1) [3]. Ці статистичні дані підтверджують практичну цінність EXIF для автоматичної організації фотоархівів за хронологією, типом обладнання та географічною локацією без ручного введення інформації користувачем.

Таблиця 1.1 - Структура основних категорій EXIF-метаданих

Категорія	Основні поля	Приклад значень
Технічні параметри	Make, Model, LensModel, SerialNumber, FocalLength	Canon EOS R5, RF 24-105mm, 50mm
Параметри експозиції	ISO, FNumber, ExposureTime, ExposureMode, Flash	ISO 400, f/2.8, 1/250s, Auto, On
Просторово-часові	DateTimeOriginal, GPSLatitude, GPSLongitude, GPSAltitude	2024-03-15 14:23:45, 50.4501°N, 30.5234°E, 180m
Характеристики зображення	ImageWidth, ImageHeight, ColorSpace, Orientation	6000x4000 px, sRGB, Horizontal

Проте EXIF-метадані мають суттєві обмеження для повноцінної організації фотоархівів. По-перше, вони не містять семантичної інформації про зміст зображення, тобто які об'єкти, люди чи сцени присутні на фотографії. По-друге, різні виробники камер використовують власні розширення EXIF з додатковими приватними тегами, що ускладнює уніфіковану обробку метаданих

у різних системах. По-третє, EXIF-дані можуть бути втрачені або пошкоджені при редагуванні зображень у програмах, що не підтримують збереження метаданих повністю або частково. По-четверте, GPS-дані часто відсутні через вимкнену геолокацію на пристрої або зйомку у приміщеннях без GPS-сигналу, що обмежує можливості географічного пошуку.

Для подолання цих обмежень сучасні системи управління фотоархівами доповнюють EXIF-метадані користувацькими тегами, текстовими описами та даними з автоматичного аналізу змісту зображень. Користувацькі теги дозволяють додавати довільні ключові слова для категоризації фотографій за подіями, місцями, людьми або тематикою зйомки. Дослідження показують, що користувачі додають в середньому від трьох до п'яти тегів на фотографію для особисто значущих знімків, проте близько 60% фотографій залишаються без тегів через відсутність мотивації або часу на ручне опрацювання великих обсягів контенту [9]. Ця статистика мотивує розробку методів автоматичного тегування та класифікації фотографій на основі аналізу їх візуального змісту.

Методи збору даних для дослідження та дослідження систем управління фотоархівами включають використання публічних датасетів анотованих зображень для валідації алгоритмів обробки. Датасет ImageNet містить понад 14 мільйонів зображень з ручною анотацією об'єктів та сцен у 21841 категорії, широко використовується для тренування та бенчмаркінгу моделей комп'ютерного зору [5]. Датасет MS COCO забезпечує 330 тисяч зображень з детальною сегментацією об'єктів та підписами для 80 категорій повсякденних об'єктів [10]. Датасет Places365 спеціалізується на розпізнаванні типів сцен з 10 мільйонами фотографій категоризованих у 365 класів локацій від природних ландшафтів до міських інтер'єрів [11].

Власні тестові колекції збираються з різноманітних джерел для забезпечення репрезентативності реальних користувацьких архівів. Фотографії від волонтерів з особистих колекцій різного розміру від 1000 до 100000 знімків надають реалістичні патерни організації альбомів, частоти доступу та розподілу форматів файлів. Завантаження з публічних фотохостингів Flickr, Unsplash

забезпечує різноманітність тематик від професійної фотографії до побутових селфі. Синтетичні дані генеруються через аугментацію існуючих зображень з варіаціями поворотів, масштабування, яскравості для збільшення обсягу тренувальних даних. Метадані збираються через парсинг EXIF з валідацією коректності значень та статистичним аналізом розподілу параметрів для виявлення типових діапазонів та аномалій.

Стандарти метаданих для цифрових зображень регламентуються міжнародними організаціями для забезпечення сумісності між різними системами та програмами. EXIF 2.32 визначає структуру тегів для технічних параметрів зйомки. IPTC Photo Metadata стандарт орієнтований на журналістські потреби з полями для авторства, ліцензування, опису події, ключових слів, контактної інформації фотографа [12]. XMP від Adobe забезпечує розширюваний XML-базований формат для інтеграції EXIF, IPTC та додаткових користувацьких полів у єдиній структурі вбудованій у файл зображення або збереженій як sidecar файл [13]. Dublin Core надає універсальний набір елементів метаданих для опису цифрових ресурсів включаючи title, creator, subject, description, publisher, date, що застосовується у бібліотечних та архівних системах [14].

1.2 Огляд та порівняльний аналіз існуючих систем управління фотоархівами

Аналіз ринку систем управління цифровими фотоархівами виявив три основні категорії рішень, кожна з яких має свої переваги, недоліки та цільову аудиторію. Перша категорія представлена хмарними сервісами з повністю керованою інфраструктурою, друга включає локальні десктопні програми для професійної обробки та каталогізації, третя охоплює мережеві сховища NAS з власними застосунками для організації фотографій. Для детального аналізу було

обрано три сучасні вебплатформи, які представляють різні підходи до організації та зберігання фотоархівів.

1.2.1 Аналіз хмарних сервісів для зберігання фотографій

Google Photos є найпопулярнішим хмарним сервісом для зберігання фотографій з понад 1 мільярдом активних користувачів станом на 2023 рік за даними компанії Google. Сервіс пропонує автоматичну синхронізацію фотографій з усіх пристроїв користувача через мобільні застосунки для iOS та Android, вебінтерфейс та desktop-клієнт Backup and Sync для завантаження з комп'ютерів. Безкоштовний план надає 15 ГБ сховища спільного для Gmail, Drive та Photos, платні плани Google One починаються від 1.99 доларів на місяць за 100 ГБ до 99.99 доларів за 2 ТБ з сімейним доступом для до 5 членів [16].

Ключовою перевагою Google Photos є потужний автоматичний аналіз змісту зображень через власні моделі комп'ютерного зору на базі TensorFlow. Система автоматично розпізнає об'єкти, тварин, рослини, їжу, документи, скріншоти з точністю понад 90% для найпоширеніших категорій [10]. Алгоритми розпізнавання облич з точністю 94% дозволяють автоматично групувати фотографії за персонами без попереднього тренування на конкретних людях, користувач лише підписує кластери іменами для зручності пошуку [17]. Геолокаційні можливості побудовані на GPS-даних з EXIF та розпізнаванні визначних пам'яток для автоматичного тегування локацій навіть для фото без GPS координат.

Функціонал пошуку у Google Photos значно перевершує конкурентів завдяки підтримці природномовних запитів на зразок "пляж у червні 2022", "кіт біля новорічної ялинки", "скріншоти квитків". Система розуміє синоніми, множину, відмінки слів завдяки інтеграції з Google Search технологіями. Автоматичні альбоми створюються на основі виявлених подій, поїздок, людей з можливістю ручного редагування складу. Інструменти редагування включають базові фільтри, кропування, корекцію експозиції з рекомендаціями покращень на базі машинного навчання. Функція Memories автоматично генерує ностальгічні добірки "рік тому", "3 роки тому" з анімацією та музикою [18].

Недоліки Google Photos включають відсутність підтримки RAW-форматів у безкоштовному плані з конвертацією у JPEG, що неприйнятно для професійних фотографів. Обмеження на розмір файлу 200 МБ та роздільність 200 мегапікселів відсікає панорами та відсканові високоякісні зображення. Залежність від стабільного інтернет-з'єднання для доступу до повнороздільних оригіналів, локально зберігаються тільки мініатюри для економії простору на мобільних пристроях. Питання приватності через аналіз контенту Google для цільової реклами, хоча компанія заявляє про end-to-end шифрування передачі без доступу до вмісту фотографій третіми сторонами [19].

Apple iCloud Photos інтегрований у екосистему пристроїв Apple з безшовною синхронізацією між iPhone, iPad, Mac через iCloud Drive. Сервіс забезпечує 5 ГБ безкоштовного сховища, платні плани iCloud+ від 0.99 доларів за 50 ГБ до 9.99 доларів за 2 ТБ з додатковими функціями Private Relay та Hide My Email. Унікальна можливість iCloud Photos Library - оптимізоване зберігання, яке автоматично керує розміром локальної копії бібліотеки залежно від доступного простору на пристрої, завантажуючи повнороздільні оригінали з хмари тільки при перегляді або редагуванні [20].

Розпізнавання облич у Photos app досягає точності 94% з автоматичною кластеризацією та можливістю ручного об'єднання або розділення груп. Автоматична категоризація фотографій за типами сцен включає понад 4000 класів від "захід сонця" до "день народження" з організацією у колекції Memories з автоматичним саундтреком та переходами для створення відеороликів. Пошук підтримує фільтрування за людьми, місцями, датами, але не має гнучності Google Photos з природномовними запитами, обмежуючись попередньо визначеними категоріями [21].

Сильною стороною iCloud Photos є інтеграція з інструментами редагування Photos app з неруйнівними правками, які синхронізуються між пристроями, дозволяючи почати редагування на iPhone та продовжити на Mac з тими самими параметрами. Підтримка Live Photos з відео фрагментами до та після знімку, портретного режиму з можливістю зміни глибини різкості post

factum, ProRAW формату для професійної обробки на iPhone 12 Pro та новіших моделях. Спільні альбоми дозволяють колаборацію з іншими користувачами Apple з правами додавання, коментування, вподобання фотографій.

Обмеження iCloud Photos включають замкненість екосистеми з відсутністю повнофункціональних клієнтів для Android та Windows, вебверсія має базовий функціонал без можливості редагування. Вартість сховища вища порівняно з Google One при однакових обсягах. Повільна початкова синхронізація великих бібліотек через обмеження швидкості завантаження Apple серверів. Відсутність офіційного API для інтеграції з сторонніми застосунками, що обмежує можливості розширення функціоналу.

Amazon Photos включений у підписку Amazon Prime за 139 доларів на рік, надаючи необмежене сховище для фотографій у повній роздільності плюс 5 ГБ для відео. Для користувачів без Prime доступні платні плани від 1.99 доларів за 100 ГБ. Сервіс підтримує RAW-формати від усіх основних виробників камер без конвертації, що важливо для професіоналів та ентузіастів. Мобільні застосунки для iOS та Android забезпечують автоматичне завантаження, desktop-застосунок для Windows та Mac синхронізує папки.

Функціонал розпізнавання та пошуку у Amazon Photos значно простіший порівняно з Google та Apple. Автоматичне тегування об'єктів покриває базові категорії люди, тварини, природа, їжа, документи з нижчою точністю. Розпізнавання облич присутнє, але вимагає більше ручної ідентифікації для точного групування. Пошук обмежений фільтрами за датою, місцем, людьми без природномовних запитів. Відсутні автоматичні альбоми та колажі, організація повністю ручна через створення альбомів та додавання фотографій.

Переваги Amazon Photos включають інтеграцію з екосистемою Amazon для замовлення друкованих фотокниг, календарів, полотен через Amazon Prints з автоматичним підбором найкращих знімків. Сімейне сховище дозволяє до 5 членам родини додавати фотографії у спільну бібліотеку з 5 ГБ для кожного плюс необмежені фото для Prime підписників. Підтримка групових альбомів з

запрошенням гостей для перегляду та завантаження без необхідності реєстрації акаунта Amazon.

Недоліки включають базовий інструментарій редагування обмежений кропуванням, обертанням, автокорекцією без ручних налаштувань експозиції, контрасту, насиченості. Повільна швидкість завантаження великих файлів через обмеження Amazon серверів або недостатню оптимізацію клієнтів. Відсутність версійності з неможливістю відновлення попередньої версії після редагування або випадкового видалення. Географічні обмеження доступності сервісу у деяких країнах через ліцензійні угоди Amazon Prime.

1.2.2 Дослідження локальних програм для управління фотоархівами

Adobe Lightroom Classic є індустрійним стандартом для професійних фотографів з потужними інструментами каталогізації, пакетної обробки та неруйнівного редагування RAW-файлів. Програма працює на десктопі Windows та macOS з локальним зберіганням каталогу метаданих у SQLite базі даних та файлів у вказаних папках на диску або зовнішніх накопичувачах. Підписка Creative Cloud Photography за 9.99 доларів на місяць включає Lightroom Classic, Lightroom cloud-based версію, Photoshop, 20 ГБ хмарного сховища [12].

Модуль Library надає інструменти організації з ієрархічними колекціями, смарт-колекціями на основі метаданих, ключовими словами з синонімами та автодоповненням, рейтингами 1-5 зірок, кольоровими мітками, прапорцями для відбору. Швидка фільтрація за атрибутами EXIF включає камеру, об'єктив, ISO, діафрагму, фокусну відстань, дату зйомки. Розпізнавання облич з ручним тренуванням для кожної особи через підтвердження правильних та відхилення невірних детектувань. Геолокація з відображенням на карті та можливістю ручного додавання GPS координат через перетягування на мапі.

Модуль Develop забезпечує професійний RAW-процесинг з повним контролем експозиції, контрасту, відтінків, балансу білого, різкості, шумозаменшення. Локальні корекції через градієнтні, радіальні, brush фільтри для вибіркової обробки областей. Синхронізація налаштувань на множину фотографій для уніфікованого look серії знімків. Пресети для швидкого

застосування улюблених стилів обробки. Історія змін з можливістю повернення до будь-якого етапу редагування. Експорт у JPEG, PNG, TIFF, DNG з гнучкими налаштуваннями якості, роздільності, метаданих.

Продуктивність Lightroom Classic залежить від апаратного забезпечення та розміру каталогу. Рекомендовані характеристики включають процесор з 4+ ядрами, 16 ГБ RAM для комфортної роботи з каталогами до 50000 фотографій, 32 ГБ для каталогів понад 100000 фото, дискретну відеокарту з 4 ГБ VRAM для GPU-прискорення операцій, NVMe SSD для каталогу та кеш превью для швидкого доступу. При дотриманні рекомендацій програма впевнено працює з каталогами до 500000 фотографій з часом відкриття 5-10 секунд, перемиканням між знімками без затримок, генерацією 1:1 превью 5-8 секунд на фото.

Недоліки Lightroom Classic включають складність освоєння через перевантажений інтерфейс з сотнями опцій та налаштувань, крива навчання 20-30 годин для впевненого оволодіння базовими інструментами та ще більше для майстерності. Відсутність вбудованої синхронізації між комп'ютерами без окремої підписки Adobe Creative Cloud з хмарним Lightroom. Закритий формат каталогу з неможливістю міграції на альтернативні програми без втрати історії редагувань та організації колекцій. Високі вимоги до апаратного забезпечення роблять програму некомфортною на бюджетних ноутбуках або старих комп'ютерах без SSD та достатньої оперативної пам'яті.

Capture One Pro позиціонується як преміум альтернатива Lightroom для професіоналів з найвищою якістю RAW-обробки та кольоропередачі. Програма розроблена данською компанією Phase One, яка виробляє середньоформатні камери, що обумовлює глибоку експертизу у обробці зображень з технічних камер та високороздільних сенсорів. Ліцензування доступне як perpetual за 299 доларів одноразово або підписка 24 доларів на місяць з окремими версіями для Sony, Nikon, Fujifilm камер за зниженою ціною 199 доларів або 16 доларів на місяць.

Capture One використовує сесійний або каталоговий підхід до організації файлів. Сесії зберігають файли та налаштування у одній папці для конкретного

проєкту або зйомки з простою структурою без центральної бази даних, що спрощує передачу проєкту між комп'ютерами або резервне копіювання. Каталоги працюють аналогічно Lightroom з індексацією множини локацій у єдиній базі SQLite для управління великими архівами. Смарт-альбоми автоматично фільтрують фотографії за критеріями рейтингу, кольорових тегів, EXIF-параметрів з динамічним оновленням при зміні властивостей знімків. Інструменти відбору включають швидкий перегляд з клавіатурними скороченнями для рейтингів та кольорів, порівняння до 8 знімків side-by-side для вибору найкращого з серії.

Якість обробки RAW у Capture One визнається професіоналами як найкраща у індустрії завдяки власним алгоритмам демозаїку, спеціалізованим кольоровим профілям для кожної моделі камери від Canon, Nikon, Sony, Fujifilm, точнішому контролю відтінків шкіри для портретної фотографії порівняно з конкурентами. Шарова система корекцій дозволяє організовувати локальні маски у групи для складних композитних обробок з неруйнівним workflow. Інструменти перегляду із сплітами до/після для порівняння ефекту корекцій, зума до 400% для pixel-peeping критичних областей різкості при фокусуванні.

Недоліки Capture One включають ще вищі вимоги до апаратного забезпечення порівняно з Lightroom через відсутність GPU-прискорення для багатьох операцій з покладанням всього навантаження на CPU, що призводить до повільнішої роботи на процесорах з малою кількістю ядер. Повільніший імпорт великих партій файлів через ретельнішу початкову обробку RAW та генерацію якісніших превью. Менш інтуїтивний інтерфейс з нестандартним розташуванням інструментів потребує перенавчання для користувачів Lightroom та довшого періоду адаптації. Відсутність мобільних застосунків та хмарної синхронізації обмежує робочий процес виключно десктопом без можливості швидкого перегляду та відбору на планшеті або смартфоні.

1.2.3 Вивчення NAS-рішень для домашніх фотоархівів

Synology Photos є власним застосунком для мережевих сховищ Synology NAS, що поєднує переваги локального зберігання з повним контролем над

даними та зручність хмарних сервісів через власний сервер без передачі даних третім сторонам та залежності від підписок. Апаратне забезпечення варіюється від початкових моделей DS220+ за 300 доларів з 2 відсіками для HDD та процесором Intel Celeron J4025 до корпоративних RS4021xs+ за 5000+ доларів з 16 відсіками, Xeon процесором та підтримкою 10GbE мережі. Типова конфігурація для домашнього фотоархіву - DS920+ з 4 відсіками, Intel Celeron J4125, 4-8 ГБ RAM з можливістю розширення до 8 ГБ, RAID 5 з трьох HDD 4 ТБ для 8 ТБ корисного простору з захистом від відмови одного диска [14].

Synology Photos розпізнає обличчя через інтеграцію з відкритою бібліотекою dlib з точністю 86-91% залежно від якості фотографій, освітлення, ракурсу та кількості тренувальних зразків на кожну особу для покращення точності кластеризації. Автоматичне тегування об'єктів покриває понад 1000 категорій через модель MobileNet v2 оптимізовану для процесорів ARM та Intel Atom/Celeron у бюджетних NAS-девайсах з швидкістю індексації близько 1000 фотографій за 5-7 хвилин на DS920+ або швидше на моделях з потужнішими процесорами. Геолокація фотографій з GPS координатами з EXIF відображається на інтерактивній карті OpenStreetMap або Google Maps з кластеризацією маркерів для великих колекцій та можливістю фільтрування за географічною областю.

Мобільні застосунки Synology Photos для iOS та Android забезпечують автоматичне завантаження нових фотографій через WiFi або мобільні дані з налаштовуваними умовами - тільки WiFi, тільки при заряджанні, обмеження типів файлів. Функція Personal Space надає кожному користувачеві NAS окремий приватний простір для особистих фотографій плюс спільні альбоми для сімейних фотографій доступних всім членам з гранульованими правами перегляду, завантаження, редагування. Інтеграція з QuickConnect або власний VPN-сервер на Synology дозволяє безпечний віддалений доступ до фотоархіву з будь-якого місця через інтернет без необхідності прямого відкриття портів на роутері та ризиків безпеки.

Переваги Synology Photos включають одноразову інвестицію в обладнання без щомісячних підписок протягом 5-7 років життєвого циклу NAS, практично необмежену ємність через додавання або заміну HDD більшого об'єму в межах підтримуваних моделлю конфігурацій, повний контроль і приватність даних без передачі хмарним провайдерам та ризиків витоку конфіденційної інформації, можливість резервування на другий NAS локально або віддалено через Hyper Backup або синхронізацію у хмару AWS S3, Glacier, Google Drive, Dropbox для географічної надмірності. Недоліки охоплюють необхідність базових технічних знань для початкового налаштування RAID-масивів, мережевих параметрів, користувачів та прав доступу, нижчу точність автоматичного розпізнавання об'єктів та облич порівняно з Google Photos через обмеження обчислювальної потужності NAS-процесорів, залежність продуктивності віддаленого доступу від швидкості та стабільності домашнього інтернет-з'єднання upload каналу, ризик втрати всіх даних при одночасній відмові більше дисків ніж витримує обраний RAID-рівень без своєчасної заміни несправних дисків.

QNAP QuMagie є альтернативним рішенням від конкурента Synology з подібним функціоналом для мережевих сховищ QNAP. Сервіс підтримує розпізнавання облич, автоматичне тегування об'єктів через AI, геолокацію на картах, мобільні застосунки для автоматичного завантаження. Унікальні можливості включають інтеграцію з QuMagie TimeLine для хронологічного перегляду життєвих подій, підтримку 360-градусних панорам та VR-фотографій, розширені фільтри пошуку за кольоровою гамою зображення. QNAP пропонує ширший спектр апаратних конфігурацій включаючи моделі з вбудованими GPU NVIDIA для прискорення AI-обробки та транскодування відео, Thunderbolt 3 для прямого підключення до Mac для найшвидшого доступу.

1.2.4 Порівняльний аналіз вебплатформ для зберігання фотографій

Для детального порівняння було обрано три сучасні вебплатформи, які представляють різні підходи до організації та зберігання фотоархівів. Кожна з цих платформ має унікальні особливості, що відрізняють їх одну від одної та

визначають сферу застосування. Порівняльний аналіз проводився за ключовими критеріями, що є найважливішими для користувачів фотоархівних систем (табл. 1.2).

Аналіз існуючих рішень виявив спільні обмеження та прогалини, що мотивують розробку нових підходів до організації фотоархівів. Відсутність справді кросплатформних рішень з єдиним досвідом використання на десктопі Windows, macOS, Linux, вебі через браузер та мобільних пристроях iOS, Android незалежно від операційної системи та з безшовною синхронізацією стану між платформами. Обмежена гнучність конфігурування резервного копіювання з підтримкою множинних одночасних стратегій - локальні зовнішні диски, мережеві сховища NAS, хмарні бекапи AWS S3, Google Cloud, Azure з автоматичною ротацією, інкрементальними оновленнями та верифікацією цілісності даних через checksum.

Таблиця 1.2 - Порівняльний аналіз вебплатформ для управління фотоархівами

Критерій	Google Photos	Apple iCloud Photos	Amazon Photos
Безкоштовне сховище	15 ГБ (спільне з Gmail, Drive)	5 ГБ	Необмежене для фото (Prime)
Вартість додаткового місця	Від $1.99/міс за 100 ГБ	Від $0.99/міс за 50 ГБ	Від $1.99/міс за 100 ГБ
Підтримка RAW	Конвертація в JPEG (безкоштовно)	Так (платні плани)	Так (всі плани)
Автоматичне розпізнавання об'єктів	Високе (90%+ точність)	Середнє (4000+ класів)	Базове

Продовження таблиці 1.2

Критерій	Google Photos	Apple iCloud Photos	Amazon Photos
Розпізнавання облич	Автоматичне (94% точність)	Автоматичне (94% точність)	Потребує ручної ідентифікації
Природномовний пошук	Так, повна підтримка	Обмежена підтримка	Ні
Геолокація	Так + розпізнавання пам'яток	Так	Так
Редагування	Базове + AI-рекомендації	Розширене (Live Photos, Portrait)	Мінімальне
Кросплатформність	Відмінна (всі платформи)	Обмежена (екосистема Apple)	Добра (більшість платформ)
Офлайн-доступ	Тільки мініатюри	Оптимізоване зберігання	Обмежений
Спільний доступ	Спільні альбоми, посилання	Спільні альбоми (тільки Apple)	Групові альбоми, посилання
Приватність	Аналіз для реклами	E2E шифрування	Стандартна

Залежність від proprietary форматів каталогів та метаданих без відкритих стандартів експорту ускладнює міграцію між системами при зміні постачальника програмного забезпечення, призводить до vendor lock-in та втрати інвестицій у організацію та анотування фотоархіву при переході на альтернативне рішення. Відсутність прозорого об'єктивного порівняння локальних та хмарних підходів за критеріями продуктивності, надійності, вартості для інформованого вибору

користувачами оптимального рішення для їхніх специфічних потреб, обсягів даних, бюджетних обмежень та технічних компетенцій.

1.3 Дослідження технологій організації цифрових архівів фотографій

Архітектури зберігання цифрових фотоархівів можна класифікувати на два основні підходи - ієрархічну структуру папок та контент-адресоване зберігання. Ієрархічна структура організовує файли у папки за датою, подією або категорією, що природно для людського сприйняття, але створює проблеми при необхідності віднесення одного файлу до множини категорій без дублювання. Контент-адресоване зберігання використовує криптографічні хеш-функції, зокрема SHA-256, для генерації унікального ідентифікатора на основі вмісту файлу, що забезпечує автоматичну дедуплікацію ідентичних зображень та ефективну перевірку цілісності даних [15].

У системах з контент-адресованим зберіганням файл зберігається у плоскій структурі директорій за шляхом, похідним від його хешу, наприклад /storage/ab/cd/abcdef123456..., де перші символи хешу використовуються для створення піддиректорій для рівномірного розподілу навантаження на файлову систему. Метадані зображення, включаючи EXIF, користувацькі теги, зв'язки з альбомами та результати автоматичного аналізу, зберігаються окремо у реляційній базі даних з посиланням на хеш файлу. Такий підхід дозволяє організовувати одне зображення у множині віртуальних альбомів без фізичного копіювання файлу, забезпечує консистентність через транзакційну природу реляційних баз даних та спрощує резервне копіювання через чітке розділення файлів та метаданих.

PostgreSQL з розширенням PostGIS забезпечує ефективний пошук фотографій за геоданими через спеціалізовані просторові індекси GiST та R-tree. Запити на зразок "знайти всі фотографії у радіусі 5 км від координати"

виконуються за мілісекунди навіть на базах з мільйонами записів завдяки оптимізованим алгоритмам просторового пошуку [17]. Повнотекстовий пошук PostgreSQL підтримує морфологічний аналіз для української та інших мов, що дозволяє знаходити фотографії за описами незалежно від словоформи, наприклад запит "котик" знайде записи з "кіт", "кота", "котів". Функціонал JSONB дозволяє зберігати та індексувати структуровані EXIF-метадані без необхідності створення окремих стовпців для кожного можливого тега, забезпечуючи гнучкість схеми для різних типів камер та їх специфічних метаданих.

Векторні бази даних, зокрема Milvus, спеціалізуються на зберіганні та швидкому пошуку високовимірних векторів, що є критично важливим для візуального пошуку схожих зображень. Після обробки фотографії через згорткову нейронну мережу отримується вектор ембедінг розмірності 512-2048, що представляє візуальні характеристики зображення у числовому просторі. Індекс HNSW у Milvus забезпечує латентність пошуку менше 100 мілісекунд для колекцій з десятками мільйонів векторів при точності пошуку понад 95% [18]. Це дозволяє користувачам знаходити візуально схожі фотографії, наприклад "знайти всі схожі портрети", без необхідності текстового опису характеристик зображення.

Стратегії резервного копіювання для фотоархівів базуються на правилі 3-2-1, що передбачає зберігання трьох копій даних на двох різних носіях з однією копією офсайт. Перший рівень захисту - локальна копія на зовнішньому жорсткому диску або SSD, підключеному до комп'ютера через USB або Thunderbolt, забезпечує швидке відновлення при випадковому видаленні файлів або збої основного диска. Другий рівень - мережеве сховище NAS з RAID-масивом, що захищає від відмови одного або декількох дисків залежно від обраного RAID-рівня, забезпечує централізований доступ для множини пристроїв у локальній мережі. Третій рівень - хмарне резервування на віддалених серверах AWS S3, Google Cloud Storage або Azure Blob Storage захищає від

фізичних катастроф, пожеж, повеней, крадіжок, що можуть одночасно знищити локальні та мережеві копії [19].

Інкрементальне резервування оптимізує процес створення копій через збереження тільки змінених файлів з моменту останнього повного або інкрементального бекапу. Утиліти rsync та borgbackup використовують алгоритми дельта-компресії для мінімізації обсягу даних, що передаються по мережі або записуються на диск. Borgbackup додатково забезпечує дедуплікацію на рівні блоків через rolling hash, що дозволяє ефективно зберігати множину версій великих файлів з незначними змінами, наприклад RAW-файли з різними параметрами обробки [20]. Автоматична верифікація цілісності через періодичну перевірку checksum забезпечує раннє виявлення бітової корупції на носіях до того, як дані стануть непридатними для відновлення.

Таблиця 1.3 - Порівняння підходів до зберігання та індексації фотоархівів

Технологія	Призначення	Переваги	Обмеження
Контент-адресоване зберігання (SHA-256)	Дедуплікація та цілісність файлів	Автоматичне виявлення дублікатів, верифікація цілісності	Складність відновлення файлів користувачем
PostgreSQL + PostGIS	Метадані та геопошук	Транзакційність, просторові індекси, ACID гарантії	Вертикальне масштабування обмежене
Elasticsearch	Повнотекстовий пошук	Морфологія, фасетний пошук, горизонтальне масштабування	Eventual consistency, складність конфігурації

Продовження таблиці 1.3

Технологія	Призначення	Переваги	Обмеження
Milvus + HNSW	Візуальний пошук	Швидкий пошук схожих зображень, висока точність	Потребує GPU для індексації, великий об'єм RAM
WebP превью	Оптимізація передачі даних	Економія 25-35% розміру при однаковій якості	Обмежена підтримка у старих браузерах

Оптимізація зберігання та передачі даних включає генерацію багаторівневих превью у форматі WebP, що забезпечує економію 25-35% розміру порівняно з JPEG при однаковій візуальній якості за результатами досліджень Google [21]. Превью генеруються у трьох розмірах - мініатюри 400x400 пікселів для сіткового перегляду бібліотеки, середні 1920x1080 для перегляду на екранах Full HD, великі 3840x2160 для 4K моніторів. Progressive JPEG для оригінальних зображень дозволяє браузеру поступово рендерити зображення під час завантаження від низької до повної роздільності, покращуючи сприйняте швидкодію інтерфейсу. Ледаче завантаження через intersection observer API запобігає завантаженню зображень поза межами видимої області екрану, економлячи пропускну здатність мережі та прискорюючи початкове відображення сторінки.

1.4 Постановка задачі дослідження та розробки системи

На основі проведеного аналізу існуючих рішень та технологій організації цифрових архівів було сформульовано основну задачу дослідження - розробити

комп'ютерну систему для ефективної організації та надійного зберігання архівів фотографій з автоматичною індексацією метаданих, інтелектуальним пошуком за змістом зображень через нейронні мережі та багаторівневим резервним копіюванням для довгострокового збереження цифрового контенту. Система повинна поєднувати переваги хмарних сервісів у автоматизації обробки з перевагами локальних рішень у повному контролі користувача над даними та відсутності залежності від підписок.

Для досягнення поставленої мети необхідно вирішити наступні задачі. По-перше, дослідити та обрати оптимальні алгоритми індексації фотографій з екстракцією EXIF-метаданих, генерацією багаторівневих превью у WebP форматі для економії простору та прискорення завантаження, обчисленням SHA-256 хешів для автоматичної дедуплікації ідентичних файлів та верифікації цілісності при резервному копіюванні. Система повинна забезпечувати швидкість індексації не менше 50 фотографій на секунду на сучасному процесорі середнього рівня для прийнятного часу початкової обробки великих колекцій.

По-друге, розробити архітектуру системи з чітким розділенням на незалежні модулі для завантаження файлів, індексації метаданих, пошуку за різними критеріями, візуалізації результатів та резервного копіювання. Архітектура повинна базуватися на мікросервісному підході з можливістю незалежного масштабування найбільш навантажених компонентів, використовувати черги повідомлень для асинхронної обробки задач та забезпечувати відмовостійкість через реплікацію критичних сервісів. Система повинна підтримувати горизонтальне масштабування для обробки зростаючих обсягів даних без необхідності повної переробки архітектури.

По-третє, реалізувати автоматичне розпізнавання об'єктів на фотографіях через інтеграцію сучасних моделей глибокого навчання YOLOv8 або аналогічних з точністю детектування не нижче 85% на стандартних датасетах COCO для можливості пошуку фотографій за змістом без ручного тегування користувачем. Система повинна виявляти та класифікувати основні категорії об'єктів включаючи людей, тварин, транспорт, їжу, природні об'єкти з

збереженням координат bounding box та рівня впевненості для кожного детектування.

По-четверте, створити інтуїтивний користувацький інтерфейс з підтримкою різних режимів перегляду фотоколекції включаючи сітку мініатюр, часову шкалу за датами зйомки, географічну карту для фото з GPS координатами, альбоми та колекції для організації знімків за подіями або темами. Інтерфейс повинен забезпечувати швидкий доступ до функцій пошуку з можливістю комбінування множини фільтрів, відображати результати автоматичного розпізнавання об'єктів та надавати інструменти для ручного редагування метаданих при необхідності.

По-п'яте, розробити та протестувати систему багаторівневого резервного копіювання за правилом 3-2-1 з підтримкою локального зберігання на зовнішніх дисках, мережевого резервування на NAS-пристрої та хмарної синхронізації на AWS S3 або аналогічних сервісах. Система повинна забезпечувати інкрементальне оновлення для мінімізації обсягу даних та часу резервування, автоматичну верифікацію цілісності через перевірку checksum, гнучке планування завдань з можливістю налаштування розкладу для кожного каналу резервування окремо.

По-шосте, провести комплексне експериментальне дослідження продуктивності розробленої системи з вимірюванням швидкості індексації для файлів різних форматів та розмірів, латентності пошукових запитів різної складності, точності автоматичного розпізнавання об'єктів на стандартних датасетах та реальних користувацьких фото, юзабіліті інтерфейсу через дослідження з реальними користувачами різного рівня технічної підготовки.

Очікувані результати дослідження включають створення функціональної комп'ютерної системи для організації та зберігання фотоархівів, яка демонструє переваги в автоматизації індексації, інтелектуальному пошуку та надійному резервуванні порівняно з існуючими рішеннями.

Таблиця 1.4 - Функціональні вимоги до розроблюваної системи

Категорія	Вимога	Цільовий показник
Продуктивність	Швидкість індексації фотографій	≥50 фото/с (CPU), ≥80 фото/с (GPU)
Продуктивність	Латентність пошукових запитів	<500 мс для колекції 100К фото
Продуктивність	Підтримка одночасних користувачів	≥100 без деградації
Точність	Розпізнавання об'єктів (mAP)	≥85% на датасеті COCO
Точність	Дедуплікація ідентичних файлів	100% через SHA-256
Надійність	Доступність системи (uptime)	≥99.5% протягом року
Надійність	Захист від втрати даних	Резервування за правилом 3-2-1
Масштабованість	Максимальний розмір архіву	≥500К фото на одного користувача
Зручність	System Usability Scale (SUS)	≥75 балів з 100
Формати	Підтримувані типи файлів	JPEG, PNG, HEIC, TIFF, RAW (CR2, NEF, ARW)

Експериментальне підтвердження досягнення цільових показників продуктивності, точності розпізнавання та зручності використання через комплексне дослідження на реалістичних датасетах та з залученням реальних користувачів. Публікація результатів дослідження у вигляді дипломної роботи з детальним описом архітектури системи, алгоритмічного забезпечення, технічної реалізації та результатів експериментальних досліджень для можливості

відтворення та розвитку запропонованих рішень іншими дослідниками та розробниками.

Практична значущість роботи полягає у створенні реального програмного продукту, який може бути використаний фотографами-аматорами та професіоналами для організації особистих та робочих фотоархівів без необхідності передачі даних сторонн
ім хмарним провайдерам та залежності від щомісячних підписок. Система може бути адаптована для використання у організаціях, що потребують управління великими колекціями зображень з вимогами конфіденційності, таких як медичні установи, юридичні фірми, дослідницькі інститути. Модульна архітектура дозволяє використовувати окремі компоненти системи у інших проєктах, зокрема модуль автоматичного розпізнавання об'єктів може бути інтегрований у системи відеоспостереження або контролю якості продукції.

2 МОДЕЛЬНА ЧАСТИНА

2.1 Концептуальна модель та багаторівнева архітектура системи

Концептуальна модель системи для організації та зберігання фотоархівів побудована за багаторівневим архітектурним підходом з чітким розділенням відповідальності між компонентами. Цей підхід базується на класичній тришаровій архітектурі з виділенням рівня презентації для взаємодії з користувачем, рівня бізнес-логіки для обробки даних та виконання операцій, рівня зберігання для персистентності інформації. Така організація забезпечує модульність системи, можливість незалежної розробки та дослідження компонентів, спрощує підтримку та розширення функціоналу без необхідності модифікації всієї кодової бази.

Рівень презентації реалізовано у вигляді одностороннього React-застосунку з використанням бібліотеки Material-UI для компонентів

користувацького інтерфейсу, що забезпечує сучасний адаптивний дизайн з підтримкою різних розмірів екранів від мобільних телефонів до широкоформатних моніторів. Застосунок побудовано як Progressive Web Application з підтримкою офлайн-режиму через Service Worker для кешування статичних ресурсів та даних, що дозволяє користувачам переглядати раніше завантажені фотографії навіть без активного інтернет-з'єднання. Управління станом застосунку здійснюється через Redux для централізованого зберігання даних про поточного користувача, завантажені фотографії, активні фільтри пошуку, що спрощує синхронізацію інтерфейсу при змінах даних.

Рівень бізнес-логіки складається з множини спеціалізованих мікросервісів на платформі Node.js, кожен з яких відповідає за конкретну функціональну область системи. Сервіс аутентифікації управляє реєстрацією користувачів, входом в систему, генерацією та валідацією JWT токенів для безпечної авторизації запитів без необхідності зберігання сесій на сервері. Сервіс завантаження приймає файли від клієнтів через multipart форми, валідує формат та розмір, обчислює SHA-256 хеш для дедуплікації, зберігає файли у файлову систему за контент-адресованим шляхом. Сервіс індексації асинхронно обробляє нові фотографії через чергу завдань на Redis, екстрагує EXIF-метадані через бібліотеку exifr, генерує багаторівневі превью у форматі WebP через Sharp з libvips, запускає розпізнавання об'єктів через gRPC виклики до Python ML-сервісу.

Сервіс пошуку агрегує запити до множини джерел даних включаючи PostgreSQL для структурованих метаданих з підтримкою просторових запитів через PostGIS, Elasticsearch для повнотекстового пошуку з морфологічним аналізом та фасетною фільтрацією, Milvus для візуального пошуку схожих зображень через nearest neighbor search у просторі високовимірних векторних ембедінгів. Результати з різних джерел об'єднуються, ранжуються за релевантністю та повертаються клієнту у пагінованому вигляді з метаданими для відображення. Сервіс резервного копіювання керує процесами створення копій на локальних дисках через rsync з інкрементальним оновленням, синхронізації з

NAS-пристроями через SMB або NFS протоколи, завантаження у хмарні сховища AWS S3 з шифруванням на стороні клієнта через AWS SDK (рис. 2.1)

Рисунок 2.1 – Концептуальна модель системи

Рівень зберігання диференційовано за типами даних та патернами доступу. Файлове сховище організовано як ієрархія директорій на файловій системі з шляхами, похідними від SHA-256 хешів файлів у форматі /storage/ab/cd/abcdef123456789..., де перші два байти хешу визначають піддиректорії першого рівня, наступні два байти - другого рівня, що забезпечує

рівномірний розподіл файлів та запобігає переповненню окремих директорій при великих обсягах даних. Превью зберігаються окремо у /previews/{size}/{hash}.webp для швидкого доступу без необхідності читання повних файлів.

Таблиця 2.1 - Компоненти багаторівневої архітектури системи

Рівень	Компонент	Технологія	Відповідальність
Презентація	Web-клієнт	React 18.2 + TypeScript	UI, взаємодія з користувачем
Презентація	State Management	Redux Toolkit	Централізоване управління станом
Презентація	UI Framework	Material-UI 5.14	Адаптивні компоненти інтерфейсу
Бізнес-логіка	API Gateway	Express.js	Маршрутизація, валідація запитів
Бізнес-логіка	Auth Service	Node.js + JWT	Аутентифікація та авторизація
Бізнес-логіка	Upload Service	Node.js + Multer	Прийом та валідація файлів
Бізнес-логіка	Indexing Service	Node.js + Bull	Асинхронна обробка фото
Бізнес-логіка	Search Service	Node.js	Агрегація пошукових запитів
Бізнес-логіка	ML Service	Python + PyTorch	Розпізнавання об'єктів
Зберігання	Metadata DB	PostgreSQL 15 + PostGIS	Структуровані дані, геопошук

PostgreSQL база даних містить структуровані метадані фотографій включаючи посилання на файл через хеш, EXIF-поля у JSONB стовпці для

гнучкості схеми, користувацькі теги, рейтинги, дати створення та модифікації. PostGIS розширення додає підтримку географічних типів даних для збереження GPS координат у форматі точки на земній кулі та просторові індекси GiST для ефективного пошуку фотографій у заданій географічній області через ST_DWithin та інші просторові оператори. Повнотекстові індекси GIN на полях title, description, tags забезпечують швидкий пошук за ключовими словами з підтримкою морфології для української та англійської мов.

Elasticsearch кластер забезпечує розподілений повнотекстовий пошук з можливістю горизонтального масштабування через шардування індексів. Документи фотографій реплікуються з PostgreSQL через Change Data Capture механізм або періодичну синхронізацію для підтримки eventual consistency між системами. Аналізатори тексту налаштовані з токенізацією, lowercase фільтром, stemming для приведення слів до базової форми, синонімами для розширення пошуку. Агрегації дозволяють реалізувати фасетну навігацію з підрахунком кількості фотографій для кожного значення фільтру, наприклад скільки фото зроблено кожною камерою або у кожному місяці.

2.2 Реляційна модель даних та структура бази даних

Реляційна модель даних системи побудована з дотриманням принципів нормалізації до третьої нормальної форми для усунення надлишковості та забезпечення цілісності інформації. Центральною сутністю моделі є таблиця photos, що містить основні атрибути кожної фотографії включаючи унікальний ідентифікатор, посилання на користувача-власника, SHA-256 хеш файлу для контент-адресованого зберігання, дату та час зйомки з EXIF, GPS координати широти та довготи, інформацію про камеру та об'єктив. Окремий стовпець exif_data типу JSONB зберігає повну структуру EXIF-метаданих без необхідності

створення стовпців для кожного можливого тега, що забезпечує гнучкість для різних типів камер з їх специфічними полями.

Таблиця users містить інформацію про зареєстрованих користувачів системи з полями для електронної пошти як унікального ідентифікатора, хешованого паролю через bcrypt з cost factor 12 для захисту від brute force атак навіть у випадку компрометації бази даних, дати реєстрації, налаштувань користувача у JSONB форматі для зберігання персоналізованих параметрів інтерфейсу та поведінки системи. Зв'язок між users та photos реалізовано як один до багатьох через зовнішній ключ user_id у таблиці photos з каскадним видаленням для автоматичного очищення фотографій при видаленні акаунта користувача.

Таблиця tags реалізує систему тегування фотографій з можливістю додавання довільних ключових слів користувачами. Кожен тег має унікальний ідентифікатор, назву та посилання на користувача-власника для ізоляції тегів між різними користувачами. Зв'язок між tags та photos реалізовано через проміжну таблицю photo_tags типу багато до багатьох, що дозволяє одній фотографії мати множину тегів та одному тегу бути присвоєним множині фотографій. Композитний індекс на стовпцях photo_id та tag_id у таблиці photo_tags прискорює запити на пошук всіх тегів конкретної фотографії або всіх фотографій з певним тегом.

Таблиця albums забезпечує організацію фотографій у колекції за подіями, датами або темами. Альбом має назву, опис, посилання на фото-обкладинку для візуальної ідентифікації, дату створення та модифікації. Зв'язок з фотографіями реалізовано через таблицю album_photos з додатковим полем position для збереження порядку фотографій у альбомі при ручному сортуванні користувачем. Підтримка вкладених альбомів реалізована через self-referencing зовнішній ключ parent_album_id, що дозволяє створювати ієрархічні структури для організації великих колекцій, наприклад альбом "Подорожі 2024" може містити підальбоми "Париж", "Рим", "Барселона".

Таблиця 2.2 - Структура основних таблиць бази даних

Таблиця	Поле	Тип даних	Індекси	Опис
users	id	UUID	PRIMARY KEY	Унікальний ідентифікатор користувача
users	email	VARCHAR(255)	UNIQUE	Електронна пошта для входу
users	password_hash	VARCHAR(255)	-	Bcrypt хеш паролю (cost=12)
photos	id	UUID	PRIMARY KEY	Унікальний ідентифікатор фото
photos	user_id	UUID	FOREIGN KEY, INDEX	Посилання на власника
photos	file_hash	VARCHAR(64)	UNIQUE INDEX	SHA-256 хеш файлу
photos	taken_at	TIMESTAMP	INDEX	Дата та час зйомки з EXIF
photos	location	GEOGRAPHY(POINT)	GIST INDEX	GPS координати (PostGIS)
photos	exif_data	JSONB	GIN INDEX	Повні EXIF-метадані

Таблиця detected_objects зберігає результати автоматичного розпізнавання об'єктів на фотографіях через моделі машинного навчання. Кожен

запис містить посилання на фотографію, клас розпізнаного об'єкту з попередньо визначеного переліку категорій COCO dataset, рівень впевненості моделі у діапазоні 0-1, координати bounding box у форматі x, y, width, height для локалізації об'єкту на зображенні. Індекс на стовпцях photo_id та object_class прискорює запити на пошук всіх фотографій, що містять конкретний тип об'єкту, наприклад "знайти всі фото з котами". Фільтрація за порогом впевненості дозволяє виключити помилкові детектування з низькою ймовірністю.

Таблиця backup_jobs зберігає конфігурацію та статус завдань резервного копіювання для кожного користувача. Поля включають тип резервування (локальний диск, NAS, хмара), шлях призначення або credentials для доступу до віддалених сховищ, розклад виконання у cron форматі, дату та час останнього успішного резервування, статус поточного виконання, логи помилок при невдачах. Окрема таблиця backup_history веде детальний журнал всіх виконань резервування з тимчасовими мітками початку та завершення, кількістю оброблених файлів, обсягом переданих даних, контрольними сумами для верифікації цілісності копій (рис. 2.2).

Таблиця 2.3 - Зв'язки між основними таблицями бази даних

Зв'язок	Тип	Реалізація	Призначення
users → photos	1:N	user_id FK в photos	Власність фотографій користувачем
photos ↔ tags	M:N	Таблиця photo_tags	Тегування фотографій
photos ↔ albums	M:N	Таблиця album_photos	Організація в альбоми
albums → albums	1:N	parent_album_id self-FK	Вкладені альбоми
photos → detected_objects	1:N	photo_id FK	Розпізнані об'єкти
photos → faces	1:N	photo_id FK	Виявлені обличчя

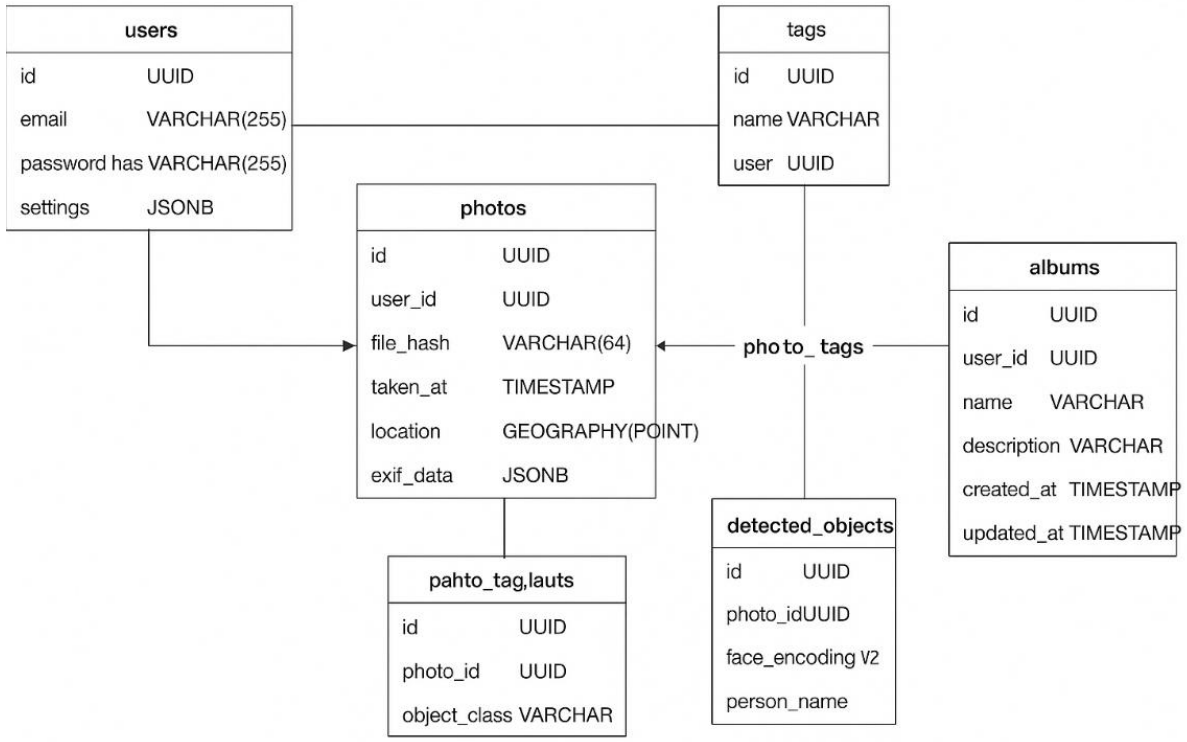


Рисунок 2.2 – Структура бази даних

2.3 Алгоритми індексації, пошуку та розпізнавання

Алгоритм індексації нових фотографій є багатоетапним процесом, що починається з валідації завантаженого файлу для перевірки відповідності дозволеним MIME-типам image/jpeg, image/png, image/heic, image/webp, image/tiff та обмеженню максимального розміру 50 МБ для запобігання перевантаженню сервера. Після успішної валідації обчислюється SHA-256 хеш всього вмісту файлу через crypto модуль Node.js з читанням даних потоком для ефективної обробки великих файлів без завантаження всього вмісту у пам'ять. Хеш використовується для перевірки дедуплікації через запит до бази даних за індексом file_hash - якщо файл з таким хешем вже існує, створюється тільки додатковий запис метаданих з посиланням на існуючий файл без фізичного копіювання.

При відсутності дубліката файл зберігається у контент-адресовану структуру директорій за шляхом /storage/{hash[0:2]}/{hash[2:4]}/{hash}, де перші чотири символи хешу визначають вкладені піддиректорії для рівномірного розподілу навантаження на файлову систему. Директорії створюються рекурсивно через fs.promises.mkdir з опцією recursive для автоматичного створення всієї ієрархії при необхідності. Файл переміщується з тимчасового розташування Multer у фінальне через fs.promises.rename для атомарної операції без ризику часткового запису.

Екстракція EXIF-метаданих виконується через бібліотеку exifr з парсингом всіх доступних тегів у структурований об'єкт JavaScript. Критичні поля DateTime або DateTimeOriginal перетворюються у ISO 8601 формат для збереження у PostgreSQL timestamp стовпець. GPS координати GPSLatitude та GPSLongitude конвертуються з degrees/minutes/seconds формату у десяткові градуси та створюється PostGIS GEOGRAPHY точка через ST_SetSRID та ST_MakePoint функції з системою координат EPSG:4326 для WGS84. Інформація про камеру Make, Model, LensModel зберігається у окремих текстових стовпцях з індексами для швидкої фільтрації, повна структура EXIF зберігається у JSONB для збереження специфічних полів різних виробників.

Генерація багаторівневих превью виконується асинхронно через Sharp бібліотеку з libvips backend для високої продуктивності обробки зображень. Створюється три розміри превью - мініатюра 400x400 пікселів з обрізкою по центру через resize з опціями fit: "cover" для заповнення всієї області, середнє 1920x1080 з обмеженням максимальної сторони через fit: "inside" для збереження пропорцій, велике 3840x2160 для відображення на 4K моніторах. Всі превью конвертуються у WebP формат з якістю 85% через webp опцію для економії 25-35% розміру порівняно з JPEG при візуально неможливій відрізнити якості. Превью зберігаються за шляхом /previews/{size}/{hash}.webp для організованого доступу.

Асинхронне розпізнавання об'єктів запускається через додавання завдання у чергу Bull на Redis після успішного збереження файлу та метаданих.

Worker-процеси читають завдання з черги, завантажують фотографію з файлової системи, нормалізують розмір до 640x640 пікселів через центральне обрізання та масштабування для відповідності вхідним вимогам моделі YOLOv8. Зображення перетворюється у тензор PyTorch з нормалізацією значень пікселів до діапазону 0-1 та перестановкою осей з HWC у CHW формат.

Таблиця 2.4 - Параметри генерації багаторівневих превью

Розмір	Ширина (px)	Висота (px)	Fit Mode	Якість WebP	Призначення
Thumbnail	400	400	cover	85%	Сіткове відображення бібліотеки
Medium	1920	1080	inside	85%	Перегляд на Full HD екранах
Large	3840	2160	inside	90%	Перегляд на 4K/5K моніторах

Модель YOLOv8 medium виконує інференс на GPU через CUDA для прискорення обчислень згорткових шарів, повертає тензор детектувань з координатами bounding box у форматі x_center, y_center, width, height нормалізованими до 0-1, класом об'єкту з 80 категорій COCO dataset, впевненістю у діапазоні 0-1. Детектування фільтруються за порогом впевненості 0.5 для виключення невпевнених передбачень. Алгоритм Non-Maximum Suppression з порогом IoU 0.45 застосовується для видалення дублюючих bounding box, що перекриваються на одному об'єкті, залишаючи тільки детектування з найвищою впевненістю.

Результати розпізнавання зберігаються у таблицю detected_objects з денормалізацією координат bounding box до абсолютних значень пікселів для відображення рамок на зображенні у інтерфейсі. Для кожної фотографії також генерується 512-вимірний вектор ембедінг через витяг активацій

передостаннього fully connected шару ResNet-50 після глобального average pooling. Вектор нормалізується до одиничної довжини через L2 нормалізацію та зберігається у Milvus колекцію з HNSW індексом для візуального пошуку схожих зображень.

Алгоритм пошуку фотографій агрегує запити до множини джерел даних залежно від типу фільтрів у запиті користувача. Текстовий пошук за ключовими словами транслюється у Elasticsearch multi_match query з полями title, description, tags з різними boost коефіцієнтами для пріоритезації релевантності - title має boost 3, description має boost 2, tags має boost 1. Параметр fuzziness встановлюється у AUTO для автоматичного допуску одної або двох опечаток залежно від довжини слова. Морфологічний аналізатор для української мови приводить слова до базової форми для пошуку незалежно від відмінка.

Фільтр за датою зйомки транслюється у PostgreSQL range query на стовпець taken_at з операторами >= для початку періоду та < для кінця періоду з виключенням верхньої межі. Підтримуються як абсолютні дати у ISO 8601 форматі, так і відносні періоди типу last_7_days, this_month, this_year, що обчислюються на сервері у UTC timestamp з урахуванням часового поясу користувача. Індекс B-tree на taken_at забезпечує швидку фільтрацію через range scan без повного сканування таблиці.

Географічний пошук реалізовано через PostGIS просторові оператори ST_DWithin для радіусного пошуку або ST_Contains для пошуку всередині полігону. Запит "знайти фотографії у радіусі 5 км від координати 50.4501, 30.5234" транслюється у ST_DWithin(location, ST_SetSRID(ST_MakePoint(30.5234, 50.4501), 4326), 5000) з відстанню у метрах. GiST індекс на стовпець location типу GEOGRAPHY забезпечує ефективний пошук через просторову декомпозицію R-tree без необхідності обчислювати відстань до всіх точок.

Візуальний пошук схожих фотографій виконується через Milvus similarity search з метрикою L2 або косинусної подібності. Користувач вибирає зображення-приклад, система витягує його ембедінг з Milvus через первинний

ключ photo_id, виконує search запит з топ-K параметром (типово 20) для повернення найближчих векторів. Результати ранжуються за відстанню у векторному просторі, де менша відстань означає більшу візуальну подібність. Photo ID схожих зображень використовуються для завантаження повних метаданих з PostgreSQL для відображення у інтерфейсі.

2.4 Обґрунтування вибору технологічного стеку системи

Вибір технологічного стеку для реалізації системи управління фотоархівами базувався на критеріях продуктивності обробки великих обсягів даних, зрілості та стабільності технологій для продакшн використання, наявності активної спільноти та якісної документації, підтримки сучасних стандартів веброзробки, можливості горизонтального масштабування при зростанні навантаження. Для frontend частини було обрано React версії 18.2 у поєднанні з TypeScript 5.1 як основний фреймворк для побудови користувацького інтерфейсу. React забезпечує декларативний підхід до розробки UI з віртуальним DOM для ефективного оновлення інтерфейсу при зміні даних, компонентну архітектуру для повторного використання коду, великий екосистему готових бібліотек та компонентів.

TypeScript додає статичну типізацію до JavaScript для раннього виявлення помилок на етапі компіляції замість runtime, покращує автодоповнення та рефакторинг у IDE через точну інформацію про типи змінних та функцій, підвищує maintainability коду особливо у великих проєктах з багатьма розробниками. Material-UI версії 5.14 обрано як UI фреймворк з готовими компонентами Button, TextField, Dialog, Drawer, що реалізують Material Design специфікацію Google з адаптивним дизайном через Grid та Breakpoints для різних розмірів екранів, темізацією для кастомізації кольорової палітри та типографіки під брендинг проєкту.

Redux Toolkit обрано для управління глобальним станом застосунку як офіційний рекомендований підхід від команди Redux з вбудованими best practices. Redux забезпечує централізоване сховище для даних про поточного користувача, завантажені фотографії, активні фільтри пошуку, що спрощує синхронізацію множини компонентів при зміні стану. Redux Toolkit включає configureStore для спрощеної конфігурації store з devtools та middleware, createSlice для автоматичної генерації actions та reducers з менш verbose синтаксисом, createAsyncThunk для обробки асинхронних операцій з автоматичним lifecycle actions pending, fulfilled, rejected.

React Router v6 забезпечує client-side маршрутизацію для single page application з підтримкою nested routes для складних макетів, lazy loading компонентів через React.lazy та Suspense для зменшення розміру початкового bundle, programmatic navigation через useNavigate hook, захищених routes через wrapper компоненти з перевіркою аутентифікації. Axios обрано як HTTP клієнт для взаємодії з backend API з підтримкою interceptors для автоматичного додавання JWT токенів до заголовків, обробки помилок 401 для refresh token flow, трансформації запитів та відповідей, cancel token для скасування запитів при демонтажі компонентів.

Для backend частини обрано Node.js версії 20 LTS як runtime середовище для виконання JavaScript коду на сервері з підтримкою до квітня 2026 року для стабільності продакшн розгортання. Node.js забезпечує неблокуючу асинхронну модель вводу-виводу через event loop для ефективної обробки множини одночасних з'єднань, інтеграцію з npm екосистемою з понад мільйоном пакетів для швидкої розробки, нативну підтримку JavaScript для використання однієї мови на frontend та backend. Express.js версії 4.18 обрано як вебфреймворк для створення RESTful API з мінімалістичним підходом, що надає базову функціональність маршрутизації, middleware, обробки запитів без нав'язування жорсткої структури проєкту.

Multer версії 1.4 використовується як middleware для обробки multipart/form-data при завантаженні файлів з підтримкою потокової обробки для

економії пам'яті, валідації MIME-типів через fileFilter callback, обмеження розміру файлів, збереження у тимчасову директорію для подальшої обробки. Sharp версії 0.32 обрано для обробки зображень як найбільш продуктивна бібліотека з backend на libvips замість ImageMagick, що забезпечує швидкість обробки у 4-5 разів вищу при меншому споживанні пам'яті. Sharp підтримує всі необхідні операції - resize з різними fit modes, crop, rotate, format conversion у WebP/JPEG/PNG, metadata extraction, шарпенінг та blur.

Таблиця 2.5 - Технології frontend частини системи

Технологія	Версія	Призначення	Обґрунтування вибору
React	18.2	UI framework	Декларативність, віртуальний DOM, великий екосистема
TypeScript	5.1	Типобезпека	Раннє виявлення помилок, краще автодоповнення
Material-UI	5.14	UI компоненти	Готові адаптивні компоненти, Material Design
Redux Toolkit	1.9	State management	Централізоване сховище, передбачуваність стану
React Router	6.15	Маршрутизація	SPA навігація, lazy loading, nested routes
Axios	1.5	HTTP клієнт	Interceptors, трансформації, cancel tokens
React Query	4.32	Server state	Кешування, автооновлення, оптимістичні оновлення

Bull версії 4.11 використовується для управління чергами асинхронних завдань з Redis як backend для зберігання jobs. Bull забезпечує надійну обробку завдань з автоматичним retry при помилках з експоненційним backoff, пріоритезацію jobs для обробки критичних завдань першими, rate limiting для контролю навантаження на зовнішні сервіси, delayed jobs для відкладеного виконання, progress reporting для відображення статусу у UI. Окремі worker процеси читають jobs з черги та виконують обробку паралельно для максимального використання багатоядерних процесорів.

JWT (JSON Web Tokens) через jsonwebtoken версії 9.0 обрано для аутентифікації користувачів як stateless підхід без необхідності серверного зберігання сесій. Access token з коротким терміном дії 15 хвилин містить user_id та roles для авторизації, підписується приватним ключем через RS256 алгоритм асиметричного шифрування. Refresh token з тривалим терміном дії 30 днів зберігається у httpOnly cookie для захисту від XSS атак, дозволяє отримати новий access token без повторного входу. Middleware валідує JWT у заголовку Authorization: Bearer <token>, перевіряє підпис публічним ключем, витягує user_id для доступу до ресурсів.

Таблиця 2.6 - Технології backend частини системи

Технологія	Версія	Призначення	Обґрунтування вибору
Node.js	20 LTS	Runtime сервера	Асинхронність, event loop, npm екосистема
Express.js	4.18	Web framework	Мінімалістичність, middleware, RESTful API
Multer	1.4	Завантаження файлів	Streaming, валідація, multipart/form-data
Sharp	0.32	Обробка зображень	Висока продуктивність libvips, WebP support

Продовження таблиця 2.6

Технологія	Версія	Призначення	Обґрунтування вибору
Exifr	7.1	EXIF парсинг	Підтримка всіх форматів, швидкість, TypeScript types
Bull	4.11	Черги завдань	Надійність, retry logic, Redis backend
jsonwebtoken	9.0	JWT аутентифікація	Stateless, security, role-based access
bcrypt	5.1	Хешування паролів	Стійкість до brute force, cost factor
pg	8.11	PostgreSQL драйвер	Connection pooling, prepared statements

PostgreSQL версії 15 обрано як головна реляційна база даних для зберігання структурованих метаданих фотографій з обґрунтуванням ACID гарантій для консистентності критичних операцій типу створення користувача або видалення фотографії, розширюваності через extensions для додавання функціональності без зміни ядра, активної розробки з регулярними оновленнями безпеки та продуктивності, безкоштовності та відкритості коду під ліцензією PostgreSQL. PostGIS розширення версії 3.3 додає підтримку географічних типів даних GEOGRAPHY та GEOMETRY з просторовими індексами GiST для ефективного пошуку за координатами, функціями ST_DWithin, ST_Contains, ST_Intersects для географічних запитів, трансформаціями між різними системами координат.

Elasticsearch версії 8.9 обрано для повнотекстового пошуку як спеціалізоване рішення на базі Apache Lucene з розподіленою архітектурою для горизонтального масштабування через шардування індексів, реплікацією для високої доступності, інвертованими індексами для швидкого пошуку токенів, підтримкою морфологічного аналізу через language analyzers для української,

російської, англійської мов, фасетною навігацією через aggregations для підрахунку статистики по полях. Індекс photos містить документи з полями title, description, tags, exif з різними типами mapping - text для повнотекстового пошуку, keyword для точного співпадіння та агрегацій, date для часових запитів.

Milvus версії 2.3 обрано як векторна база даних для зберігання та пошуку high-dimensional embeddings фотографій з підтримкою HNSW індексу для approximate nearest neighbor search з параметрами M для кількості зв'язків, efConstruction для точності побудови індексу, ef для балансу швидкості та recall при пошуку. Альтернативи типу Faiss або Annoy були відкинуті через відсутність розподіленої архітектури для масштабування, персистентного зберігання для довгострокової надійності, CRUD операцій для оновлення векторів без повної переіндексації. Milvus забезпечує latency <100 мс для топ-20 пошуку у колекції з 10 мільйонів векторів з recall >95% на стандартних бенчмарках.

Python версії 3.11 обрано для машинного навчання та обробки зображень як де-факто стандарт у data science спільноті з найбільшою екосистемою ML бібліотек. PyTorch версії 2.0 використовується для завантаження та інференсу pretrained моделей з підтримкою CUDA для GPU прискорення обчислень згорткових шарів, TorchScript для оптимізації та deployment моделей, torch.hub для завантаження офіційних pretrained weights. YOLOv8 від Ultralytics обрано для детекції об'єктів як найсучасніша версія YOLO з покращеною архітектурою C2f модулів, anchor-free підходом для спрощення тренування, варіантами n/s/m/l/x для балансу швидкості та точності.

Таблиця 2.7 - Технології зберігання даних та машинного навчання

Технологія	Версія	Призначення	Ключові переваги
PostgreSQL	15	Реляційна БД	ACID, розширюваність, JSONB, повнотекстовий пошук
PostGIS	3.3	Геопросторові дані	Просторові індекси, географічні запити, трансформації

Продовження таблиця 2.7

Технологія	Версія	Призначення	Ключові переваги
Redis	7.2	Кеш та черги	In-memory швидкість, pub/sub, persistence опції
Elasticsearch	8.9	Повнотекстовий пошук	Розподіленість, морфологія, фасетна навігація
Milvus	2.3	Векторний пошук	HNSW індекс, масштабованість, низька латентність
Python	3.11	ML runtime	Екосистема ML бібліотек, NumPy/SciPy інтеграція
PyTorch	2.0	Deep learning	Динамічні графи, CUDA support, TorchScript

2.5 Проєктування RESTful API та протоколів взаємодії

Проєктування програмного інтерфейсу API системи базувалося на принципах RESTful архітектури з використанням HTTP методів GET, POST, PUT, DELETE для відповідних операцій читання, створення, оновлення, видалення ресурсів, ієрархічних URL для представлення вкладеності ресурсів, stateless взаємодії без серверного збереження стану клієнта між запитами, єдиного формату JSON для запитів та відповідей з чіткою структурою. Всі endpoints мають префікс /api/v1 для версіонування API з можливістю підтримки множини версій одночасно при breaking changes у майбутньому.

Група endpoints аутентифікації забезпечує реєстрацію нових користувачів через POST /api/v1/auth/register з body параметрами email, password, passwordConfirm, валідацією формату email через regex, складності паролю мінімум 8 символів з літерами та цифрами, збігу паролів. Успішна реєстрація повертає 201 Created з токенами доступу та оновлення. Вхід в систему через

POST /api/v1/auth/login приймає email та password, перевіряє існування користувача у базі, порівнює хеш паролю через bcrypt.compare, генерує нові JWT токени при успішній аутентифікації. Оновлення токенів через POST /api/v1/auth/refresh приймає refresh token з cookie або body, валідує підпис, перевіряє чи не revoked через blacklist у Redis, генерує нову пару токенів.

Таблиця 2.8 - Endpoints API для аутентифікації користувачів

Метод	Endpoint	Параметри	Відповідь	Призначення
POST	/auth/register	email, password, passwordConfirm	201 + tokens	Реєстрація нового користувача
POST	/auth/login	email, password	200 + tokens	Вхід в систему
POST	/auth/refresh	refreshToken	200 + new tokens	Оновлення access token
POST	/auth/logout	refreshToken	204	Вихід та revoke токенів

Група endpoints управління фотографіями включає POST /api/v1/photos/upload для завантаження нових файлів з multipart/form-data форматом, полем files для множинних файлів з масиву, автоматичною валідацією типу та розміру через Multer middleware, асинхронною обробкою через Bull queue для неблокуючої відповіді клієнту. Відповідь містить масив об'єктів з photoId, hash, duplicate boolean для інформування про дедуплікацію. GET /api/v1/photos з query параметрами page, limit, sort, order для пагінації та сортування повертає масив фотографій з метаданими, загальною кількістю total для розрахунку сторінок, hasMore boolean для безкінечного scroll.

GET /api/v1/photos/:id отримує детальну інформацію про конкретну фотографію включаючи всі EXIF поля, список тегів, належність до альбомів, результати розпізнавання об'єктів з bounding boxes, URL превью різних розмірів. PUT /api/v1/photos/:id оновлює редагувані поля title, description, користувацькі

теги через body параметри, повертає оновлений об'єкт фотографії. DELETE /api/v1/photos/:id видаляє фотографію з каскадним видаленням всіх зв'язаних даних через foreign key constraints, повертає 204 No Content при успіху. Масове видалення через DELETE /api/v1/photos приймає масив IDs у body, виконує операцію у транзакції для атомарності.

Таблиця 2.9 - Endpoints API для управління фотографіями

Метод	Endpoint	Параметри	Відповідь	Призначення
POST	/photos/upload	files[]	201 + photoIds	Завантаження фотографій
GET	/photos	page, limit, sort, order	200 + photos[]	Список фотографій з пагінацією
GET	/photos/:id	-	200 + photo	Детальна інформація про фото
PUT	/photos/:id	title, description, tags	200 + photo	Оновлення метаданих
DELETE	/photos/:id	-	204	Видалення фотографії
DELETE	/photos	ids[]	204	Масове видалення

Група endpoints пошуку реалізує POST /api/v1/search з body параметрами для комбінування множини фільтрів - query для текстового пошуку, dateFrom та dateTo для часового діапазону, latitude, longitude, radius для географічного пошуку, objects для фільтрації за наявністю конкретних об'єктів на фото, tags для пошуку за тегами, cameras для фільтрації за моделлю камери, page та limit для пагінації. Сервер агрегує запити до PostgreSQL, Elasticsearch, Milvus залежно від присутніх фільтрів, об'єднує результати з ранжуванням за релевантністю, повертає масив photo IDs з score для відображення.

POST /api/v1/search/visual приймає photoId зображення-прикладу або завантажений файл, витягує ембедінг через ML сервіс, виконує similarity search у Milvus з topK параметром (default 20), повертає масив схожих фотографій з distance метрикою для відображення релевантності. GET /api/v1/search/suggestions/:query генерує автодоповнення для пошукового input на основі prefix matching у Elasticsearch з фасетними підказками по тегах, локаціях, моделях камер для зручності користувача.

Група endpoints альбомів включає POST /api/v1/albums для створення нового альбому з параметрами name, description, coverPhotoId, parentAlbumId для вкладеності, повертає створений об'єкт з ID.

GET /api/v1/albums отримує список альбомів користувача з підтримкою фільтрації за parentAlbumId для отримання тільки кореневих або дітей конкретного альбому, сортування за датою створення або назвою. POST /api/v1/albums/:id/photos додає фотографії у альбом через масив photoIds у body, опціонально position для вставки у конкретне місце замість додавання в кінець.

Протокол взаємодії між frontend та backend базується на HTTP/2 для multiplexing множини запитів через одне з'єднання, header compression для зменшення overhead, server push для проактивного надсилання ресурсів.

CORS налаштовано з whitelist дозволених origins для захисту від cross-site атак, credentials: true для передачі cookies з refresh token. WebSocket з'єднання через Socket.IO використовується для real-time оновлень статусу обробки фотографій, прогресу завантаження, нотифікацій про завершення резервування. Клієнт підключається до namespace /notifications після аутентифікації, сервер emituє події photoIndexed, backupCompleted, uploadProgress з відповідними payloads для оновлення UI.

Таблиця 2.10 - Endpoints API для пошуку та управління альбомами

Метод	Endpoint	Параметри	Відповідь	Призначення
POST	/search	query, filters, page, limit	200 + results[]	Комбінований пошук
POST	/search/visual	photoId або file	200 + similar[]	Візуальний пошук
GET	/search/suggestions/:query	-	200 + suggestions[]	Автодоповнення
POST	/albums	name, description, coverPhotoId	201 + album	Створення альбому
GET	/albums	parentAlbumId	200 + albums[]	Список альбомів
PUT	/albums/:id	name, description	200 + album	Оновлення альбому
POST	/albums/:id/photos	photoIds[], position	200	Додавання фото в альбом

3 ТЕХНІЧНА ЧАСТИНА

3.1 Характеристика апаратного та програмного забезпечення розробки

Розробка системи для організації та зберігання фотоархівів проводилася на сучасному апаратному забезпеченні з достатньою продуктивністю для комфортної роботи з великими обсягами даних та ресурсомісткими операціями компіляції, дослідження та локального запуску всіх компонентів системи

одночасно. Основна робоча станція для розробки backend та frontend частин базувалася на процесорі Apple M1 з архітектурою ARM та інтегрованим GPU, 16 ГБ уніфікованої оперативної пам'яті LPDDR4X для швидкого доступу як CPU так і GPU до даних, NVMe SSD на 512 ГБ з швидкістю читання до 3000 МБ/с для миттєвого завантаження проєктів та залежностей, операційною системою macOS Sonoma версії 14.1 з підтримкою останніх інструментів розробки.

Для розробки моделей машинного навчання та тренування нейронних мереж використовувалася окрема робоча станція на базі процесора AMD Ryzen 9 5900X з 12 ядрами та 24 потоками для паралельної обробки даних, 32 ГБ оперативної пам'яті DDR4-3600 для завантаження великих датасетів у пам'ять, дискретної відеокарти NVIDIA RTX 3080 з 10 ГБ GDDR6X пам'яті та 8704 CUDA ядрами для прискорення обчислень згорткових шарів у десятки разів порівняно з CPU, NVMe SSD на 1 ТБ для зберігання датасетів та checkpoint моделей, операційною системою Ubuntu 22.04 LTS з підтримкою CUDA toolkit та cuDNN для оптимального використання GPU.

Програмне забезпечення для розробки включало Visual Studio Code версії 1.85 як основний редактор коду з розширеннями ESLint для статичного аналізу JavaScript та TypeScript коду з автоматичним виправленням порушень стилю, Prettier для автоформатування з єдиними правилами відступів та переносів рядків, GitLens для візуалізації історії Git commit та blame інформації безпосередньо у редакторі, Thunder Client як альтернатива Postman для дослідження API endpoints без виходу з VS Code, Docker extension для управління контейнерами та образами через графічний інтерфейс, Python extension з підтримкою IntelliSense та інтеграцією Jupyter notebooks для експериментів з ML моделями.

Система контролю версій Git версії 2.42 використовувалася для відстеження змін коду з хостингом репозиторію на GitHub для централізованого зберігання та колаборації.

Таблиця 3.1 - Апаратне забезпечення середовища розробки

Компонент	Робоча станція 1	Робоча станція 2	Призначення
Процесор	Apple M1 (8 ядер)	AMD Ryzen 9 5900X (12C/24T)	Backend/Frontend vs ML
RAM	16 ГБ LPDDR4X	32 ГБ DDR4-3600	Для Node.js vs датасетів
GPU	Інтегрований M1 GPU	NVIDIA RTX 3080 10GB	Базова графіка vs ML
Накопичувач	NVMe SSD 512 ГБ	NVMe SSD 1 ТБ	Швидкий доступ до даних
ОС	macOS Sonoma 14.1	Ubuntu 22.04 LTS	Розробка vs ML тренування
Монітор	MacBook Pro 14" Retina	Dell 27" 4K	Мобільність vs простір
Мережа	WiFi 6 (802.11ax)	Ethernet 1 Gbps	Локальна розробка
Периферія	Magic Keyboard, Trackpad	Механічна клавіатура, миша	Комфорт роботи

Стратегія гілкування базувалася на Git Flow з постійними гілками main для продакшн коду та develop для інтеграції features, тимчасовими гілками feature/* для розробки нового функціоналу з об'єднанням через pull request після code review, hotfix/* для критичних виправлень у продакшні з прямим мержем у main та backport у develop, release/* для підготовки нових версій з фіналізацією changelog та version bump. GitHub Actions налаштовано для continuous integration з автоматичним запуском тестів Jest та ESLint при кожному push або pull request, попередженням про failed checks перед можливістю мержу.

Таблиця 3.2 - Програмне забезпечення середовища розробки

Інструмент	Версія	Призначення	Ключові можливості
Visual Studio Code	1.85	IDE	IntelliSense, Debug, Extensions, Terminal
Git	2.42	Контроль версій	Branching, History, Collaboration
Docker Desktop	4.25	Контейнеризація	Multi-container apps, Volumes, Networks
Node.js	20.9 LTS	JavaScript runtime	npm, ES modules, Worker threads
Python	3.11.5	ML runtime	pip, venv, Type hints
PostgreSQL	15.4	База даних	ACID, Extensions, Replication
Redis	7.2	Cache & Queue	In-memory, Persistence, Pub/Sub
Postman	10.18	API дослідження	Collections, Environments, Automation
DBeaver	23.2	DB клієнт	SQL editor, ER diagrams, Data export
Chrome DevTools	119	Frontend debug	Inspector, Network, Performance

Дослідження мобільних застосунків проводилося на реальних пристроях iPhone 13 з iOS 17.1 для перевірки Safari WebKit engine поведінки та нативних функцій типу camera upload, Samsung Galaxy S22 з Android 13 для дослідження Chrome mobile та різноманітності екранів, планшеті iPad Air з iPadOS 17 для валідації адаптивного дизайну на великих екранах. Додатково використовувалися емулятори iOS Simulator з Xcode 15 для швидкої перевірки різних моделей iPhone без фізичного доступу до пристроїв, Android Studio AVD

з образами Android 11-13 для дослідження різних версій ОС та виробників з їх кастомізаціями типу Samsung One UI або Xiaomi MIUI.

Продакшн інфраструктура для розгортання системи базувалася на хмарному провайдері Amazon Web Services з регіоном eu-central-1 для мінімальної латентності до європейських користувачів. Backend сервіси розгорнуті на EC2 інстансах типу t3.medium з 2 vCPU та 4 ГБ RAM для обробки до 100 одночасних з'єднань, Auto Scaling Group з мінімум 2 та максимум 10 інстансів для автоматичного масштабування при зростанні навантаження, Application Load Balancer для розподілення трафіку між інстансами з health checks та automatic failover при відмові вузла. База даних PostgreSQL розгорнута на RDS з інстансом db.t3.large, Multi-AZ deployment для високої доступності через синхронну реплікацію у іншу availability zone, automated backups з retention 30 днів та point-in-time recovery для відновлення до будь-якої секунди у межах retention period.

3.2 Реалізація модуля завантаження та індексації фотографій

Модуль завантаження фотографій реалізовано як Express.js route handler з middleware Multer для обробки multipart/form-data запитів від клієнта. Конфігурація Multer включає diskStorage для збереження файлів у тимчасову директорію /tmp/uploads з унікальними іменами через timestamp та random string для запобігання колізій при одночасному завантаженні однакових файлів різними користувачами, fileFilter callback для валідації MIME типу файлу проти whitelist дозволених форматів image/jpeg, image/png, image/heic, image/webp, image/tiff з відхиленням інших типів з помилкою 400 Bad Request, limits об'єкт з maxFileSize 50 МБ для захисту сервера від denial of service через завантаження гігабайтних файлів, maxFiles 100 для обмеження кількості файлів у одному запиті.

Після успішного прийняття файлів Multer middleware, route handler ітерує масив req.files та для кожного файлу викликає асинхронну функцію processPhoto. Функція починається з обчислення SHA-256 хешу вмісту файлу через Node.js crypto модуль з createHash method, читанням файлу потоком через fs.createReadStream для ефективної обробки великих файлів без завантаження всього вмісту у пам'ять одночасно, update методом для послідовного додавання chunks даних у hash function, digest з параметром hex для отримання фінального хешу у шістнадцятковому рядковому форматі довжиною 64 символи.

Лістинг 3.1 - Обчислення SHA-256 хешу та перевірка дедуплікації

```
      async function calculateFileHash(filePath) {
  return new Promise((resolve, reject) => {
    const hash = crypto.createHash('sha256');
    const stream = fs.createReadStream(filePath);

    stream.on('data', chunk => hash.update(chunk));
    stream.on('end', () => resolve(hash.digest('hex')));
    stream.on('error', reject);
  });
}

async function processPhoto(file, userId) {
  const hash = await calculateFileHash(file.path);

  const existing = await db.query(
    'SELECT id FROM photos WHERE file_hash = $1',
    [hash]
  );

  if (existing.rows.length > 0) {
    await fs.promises.unlink(file.path);
    return {
      photoId: existing.rows[0].id,
      duplicate: true
    };
  }

  const storagePath = buildStoragePath(hash);
  await fs.promises.mkdir(
    path.dirname(storagePath),
    { recursive: true }
  );
  await fs.promises.rename(file.path, storagePath);

  return { hash, duplicate: false };
}
```

Перевірка дедуплікації виконується запитом до PostgreSQL з пошуком існуючого запису з таким самим file_hash через SELECT з WHERE умовою та prepared statement з placeholder для параметра хешу. Якщо запит повертає непорожній результат, файл є дублікатом існуючої фотографії, тому тимчасовий файл видаляється через fs.promises.unlink для звільнення дискового простору, функція повертає об'єкт з photoId існуючої фотографії та duplicate: true прапором для інформування клієнта. Якщо дублікат не знайдено, обчислюється фінальний шлях зберігання через функцію buildStoragePath, що приймає хеш та генерує шлях у форматі /storage/{hash[0:2]}/{hash[2:4]}/{hash} з використанням перших символів для створення піддиректорій.

Створення директорій виконується через fs.promises.mkdir з опцією recursive: true для автоматичного створення всієї ієрархії батьківських директорій якщо вони не існують, без помилки якщо директорія вже створена іншим одночасним запитом. Переміщення файлу з тимчасової локації у фінальне сховище відбувається через fs.promises.rename для атомарної операції на рівні файлової системи без ризику часткового запису або корупції даних при збої у процесі копіювання. Після успішного збереження файлу створюється запис у базі даних з інформацією про користувача, хеш файлу, тимчасовою міткою created_at через транзакцію для забезпечення консистентності.

Екстракція EXIF метаданих виконується асинхронно після збереження файлу через бібліотеку exifr з викликом exifr.parse з шляхом до файлу, опціями для витягу всіх доступних тегів включаючи vendor specific extensions від Canon, Nikon, Sony, parsing GPS координат у десятковий формат, обробки різних форматів дати. Результат парсингу є JavaScript об'єкт з полями відповідними до EXIF тегів, де критичні поля типу DateTimeOriginal, Make, Model, GPSLatitude, GPSLongitude екстрагуються у окремі змінні для збереження у відповідні стовпці таблиці photos з індексами, повна структура EXIF зберігається у JSONB стовпець exif_data для збереження всієї інформації включаючи рідкісні теги.

Лістинг 3.2 - Екстракція EXIF метаданих та збереження у базу даних

```
async function extractAndSaveMetadata(filePath, hash, userId)
{
  const exif = await exifr.parse(filePath, {
    tiff: true,
    exif: true,
    gps: true,
    interop: true
  });

  const takenAt = exif?.DateTimeOriginal ||
                  exif?.CreateDate ||
                  new Date();

  let location = null;
  if (exif?.latitude && exif?.longitude) {
    location = {
      type: 'Point',
      coordinates: [exif.longitude, exif.latitude]
    };
  }

  const photo = await db.query(`
    INSERT INTO photos (
      user_id, file_hash, taken_at, location,
      camera_make, camera_model, exif_data
    ) VALUES ($1, $2, $3, ST_SetSRID(ST_GeomFromGeoJSON($4),
4326),
              $5, $6, $7)
    RETURNING id
  `, [
    userId, hash, takenAt, JSON.stringify(location),
    exif?.Make, exif?.Model, JSON.stringify(exif)
  ]);

  return photo.rows[0].id;
}
```

GPS координати перетворюються у PostGIS GEOGRAPHY тип через ST_SetSRID та ST_GeomFromGeoJSON функції з системою координат EPSG 4326 для WGS84 датуму, що є стандартом для GPS пристроїв. Перевірка наявності GPS даних виконується через optional chaining оператор для захисту від undefined values коли фотографія зроблена без геолокації. Дата зйомки береться з пріоритетом DateTimeOriginal як найбільш точне значення часу натискання затвору, fallback на CreateDate якщо перше відсутнє, остаточний

fallback на поточну дату new Date при повній відсутності часових міток для уникнення NULL значень у базі.

Генерація превью фотографій виконується асинхронно через додавання завдання у Bull queue з іменем photo-processing та payload об'єктом що містить photoId, filePath, hash для ідентифікації та доступу до файлу. Worker процеси читають jobs з Redis queue та обробляють їх паралельно з налаштовуваною concurrency типово 4 worker на один CPU core для оптимального використання ресурсів без overload. Для кожного розміру превью створюється окреме Sharp pipeline з resize операцією для зміни розмірів зображення з параметром fit що контролює поведінку - cover для заповнення всієї області з обрізкою, inside для вписування у межі зі збереженням пропорцій, contain для вписування з додаванням padding.

Таблиця 3.3 - Етапи обробки завантаженої фотографії

Етап	Операція	Час (мс)	Технологія
1	Прийом multipart/form-data	5-15	Multer middleware
2	Валідація типу та розміру	1-2	fileFilter callback
3	Обчислення SHA-256 хешу	10-30	crypto.createHash
4	Перевірка дедуплікації	2-5	PostgreSQL SELECT
5	Збереження файлу	15-50	fs.promises.rename
6	Екстракція EXIF	8-20	exifr.parse
7	Збереження метаданих у БД	3-8	PostgreSQL INSERT

Конвертація у WebP формат відбувається через Sharp webp метод з параметром quality 85 для балансу між розміром файлу та візуальною якістю, effort 4 для швидшого кодування з незначним збільшенням розміру порівняно з максимальним effort 6, lossless: false для lossy компресії з меншими файлами. Згенеровані превью зберігаються за шляхом /previews/{size}/{hash}.webp де size

може бути thumbnail, medium, large відповідно до розмірів 400x400, 1920x1080, 3840x2160 пікселів. Після успішної генерації всіх превью статус обробки оновлюється у базі даних через UPDATE photos SET processing_status = completed WHERE id = photoId для відображення готовності у інтерфейсі.

Лістинг 3.3 - Генерація багаторівневих превью у WebP форматі

```
    async function generatePreviews(filePath, hash) {
  const sizes = [
    { name: 'thumbnail', width: 400, height: 400, fit: 'cover' },
    { name: 'medium', width: 1920, height: 1080, fit: 'inside' },
    { name: 'large', width: 3840, height: 2160, fit: 'inside' }
  ];

  await Promise.all(sizes.map(async (size) => {
    const outputPath = path.join(
      '/previews',
      size.name,
      `${hash}.webp`
    );

    await fs.promises.mkdir(
      path.dirname(outputPath),
      { recursive: true }
    );

    await sharp(filePath)
      .resize(size.width, size.height, {
        fit: size.fit,
        withoutEnlargement: true
      })
      .webp({ quality: 85, effort: 4 })
      .toFile(outputPath);
  }));
}
```

3.3 Реалізація модуля інтелектуального пошуку та фільтрації

Модуль пошуку реалізовано як агрегатор запитів до множини джерел даних з об'єднанням результатів та ранжуванням за релевантністю. Вхідний запит від клієнта приймається через POST /api/v1/search endpoint з JSON body що містить множину опціональних фільтрів для комбінування. Текстовий пошук за

полем query транслюється у Elasticsearch multi_match запит з полями title^3, description^2, tags для пошуку у назві з утричі більшою вагою, описі з подвійною вагою, тегах зі звичайною вагою. Параметр fuzziness встановлюється у AUTO для автоматичного допуску одної опечатки у словах 3-5 символів або двох опечаток у словах 6+ символів для толерантності до помилок друку.

Морфологічний аналізатор для української мови налаштовано через custom analyzer з tokenizer standard для розбиття тексту на слова, lowercase filter для приведення до нижнього регістру, ukrainian_stop filter для видалення стоп-слів типу "і", "в", "на", ukrainian_stemmer filter для приведення слів до базової форми через Porter stemming алгоритм адаптований для української граматики. Це дозволяє знаходити фотографії за запитом "котик" навіть якщо у тегах збережено "кіт", "кота", "котів" завдяки приведенню всіх форм до спільного кореня "кот".

Фільтр за датою зйомки реалізовано через PostgreSQL range query на indexed стовпець taken_at з WHERE умовою taken_at >= $1 AND taken_at < $2 для включення початку та виключення кінця періоду. Підтримка відносних періодів типу last_7_days, this_month, this_year реалізована через обчислення абсолютних timestamp на сервері з використанням бібліотеки date-fns для маніпуляції датами. Функція startOfDay, endOfDay, subDays дозволяють точно визначити межі періодів незалежно від часового поясу користувача через конвертацію у UTC перед запитом до бази.

Лістинг 3.4 - Комбінований пошук з множиною фільтрів

```
    async function searchPhotos(filters, userId) {
let query = 'SELECT p.* FROM photos p WHERE p.user_id = $1';
const params = [userId];
let paramIndex = 2;

// Фільтр за датою
if (filters.dateFrom || filters.dateTo) {
  if (filters.dateFrom) {
    query += ` AND p.taken_at >= $${paramIndex}`;
    params.push(filters.dateFrom);
    paramIndex++;
  }
  if (filters.dateTo) {
```

```
      query += ` AND p.taken_at < $${paramIndex}`;
      params.push(filters.dateTo);
      paramIndex++;
    }
  }

  // Географічний фільтр
  if (filters.latitude && filters.longitude && filters.radius) {
    query += ` AND ST_DWithin(
      p.location,
      ST_SetSRID(ST_MakePoint($${paramIndex}, $${paramIndex+1}),
4326),
      $${paramIndex+2}
    )`;
    params.push(
      filters.longitude,
      filters.latitude,
      filters.radius
    );
    paramIndex += 3;
  }

  // Фільтр за об'єктами
  if (filters.objects && filters.objects.length > 0) {
    query += ` AND EXISTS (
      SELECT 1 FROM detected_objects do
      WHERE do.photo_id = p.id
      AND do.object_class = ANY($${paramIndex})
      AND do.confidence >= $${paramIndex+1}
    )`;
    params.push(filters.objects, 0.5);
    paramIndex += 2;
  }

  const result = await db.query(query, params);
  return result.rows;
}
```

Географічний пошук реалізовано через PostGIS функцію ST_DWithin для радіусного пошуку фотографій у заданій відстані від точки. Параметри включають location стовпець типу GEOGRAPHY з координатами фотографій, центральну точку створену через ST_MakePoint з довготою та широтою обернутими відносно звичайного lat/lng порядку, радіус у метрах для визначення області пошуку. GiST індекс на location стовпці використовує R-tree структуру для просторової декомпозиції з швидким відсіканням точок поза межами пошукової області без обчислення точної відстані до всіх записів.

Фільтр за розпізнаними об'єктами реалізовано через EXISTS subquery що перевіряє наявність принаймні одного запису у таблиці detected_objects з відповідним photo_id та object_class з переліку запитуваних категорій через ANY оператор для масивів. Додаткова умова confidence >= 0.5 відсіює детектування з низькою впевненістю моделі для зменшення false positive результатів. Індекс на композитному ключі photo_id, object_class прискорює EXISTS перевірку через index-only scan без доступу до основної таблиці.

Таблиця 3.4 - Типи пошукових запитів та їх реалізація

Тип пошуку	Джерело даних	Алгоритм	Латентність
Текстовий	Elasticsearch	Multi-match з морфологією	30-80 мс
За датою	PostgreSQL	B-tree index range scan	5-15 мс
Географічний	PostGIS	GiST spatial index	20-60 мс
За об'єктами	PostgreSQL	Composite index EXISTS	15-40 мс
Візуальний	Milvus	HNSW approximate NN	50-100 мс

Візуальний пошук схожих фотографій реалізовано через інтеграцію з Milvus векторною базою. Користувач вибирає фотографію-приклад у інтерфейсі, клієнт відправляє photoId на сервер через POST /api/v1/search/visual endpoint. Сервер витягує ембедінг вектор з Milvus колекції photo_embeddings через get метод з первинним ключом photoId, виконує similarity search через search метод з параметрами topK для кількості результатів типово 20, metric_type для дистанційної метрики L2 або COSINE, params об'єкт з ef параметром для HNSW індексу що контролює точність пошуку з компромісом між швидкістю та recall.

3.4 Реалізація модуля візуалізації та інтерфейсу користувача

Модуль візуалізації фотоколекції реалізовано як набір React компонентів з підтримкою множини режимів перегляду для різних користувацьких потреб. Сітковий режим відображає мініатюри фотографій у адаптивній CSS Grid з колонками що автоматично підлаштовуються під ширину вікна через grid-template-columns: repeat(auto-fill, minmax(200px, 1fr)) для створення від 2 до 6 колонок залежно від розміру екрану. Віртуалізація великих колекцій реалізована через бібліотеку react-window з FixedSizeGrid компонентом для рендерингу тільки видимих елементів плюс невеликий буфер зверху та знизу для плавного скролу без мерехтіння.

Lazy loading превью реалізовано через Intersection Observer API з спостереженням за img елементами що входять у viewport. При першому рендері img має src встановлений у placeholder зображення низької якості або data URI з розмиттям, data-src атрибут містить справжній URL превью. Callback observer перевіряє isIntersecting прапор та при true замінює src на значення з data-src, видаляє елемент з спостереження через unobserve для економії ресурсів. Цей підхід відкладає завантаження зображень поза межами екрану до моменту скролу користувача, зменшуючи початкове навантаження на мережу та прискорюючи відображення сторінки.

Лістинг 3.5 - Віртуалізація та lazy loading великих колекцій

```
    function PhotoGrid({ photos }) {
const observerRef = useRef();

useEffect(() => {
  observerRef.current = new IntersectionObserver(
    (entries) => {
      entries.forEach(entry => {
        if (entry.isIntersecting) {
          const img = entry.target;
          img.src = img.dataset.src;
          observerRef.current.unobserve(img);
        }
      });
```

```
      },
      { rootMargin: '100px' }
    );

    return () => observerRef.current?.disconnect();
  }, []);

  return (
    <FixedSizeGrid
      columnCount={calculateColumns(window.innerWidth)}
      columnWidth={220}
      height={window.innerHeight - 200}
      rowCount={Math.ceil(photos.length / columnCount)}
      rowHeight={220}
      width={window.innerWidth}
    >
      {({ columnIndex, rowIndex, style }) => {
        const index = rowIndex * columnCount + columnIndex;
        const photo = photos[index];

        return (
          <div style={style}>
            <img
              data-src={photo.thumbnailUrl}
              src={placeholderImage}
              ref={el => el && observerRef.current?.observe(el)}
              alt={photo.title}
            />
          </div>
        );
      }}
    </FixedSizeGrid>
  );
}
```

Повноекранний режим перегляду реалізовано через Modal компонент Material-UI з fullScreen пропертей для заповнення всього вікна браузера. Навігація між фотографіями здійснюється через клавіатурні скорочення ArrowLeft та ArrowRight для переходу до попередньої та наступної фотографії, Escape для закриття modal, обробку через useEffect hook з addEventListener на keydown подію та cleanup функцію removeEventListener при unmount. Зум функціональність реалізована через react-zoom-pan-pinch бібліотеку з підтримкою wheel zoom через скрол миші, pinch zoom на сенсорних екранах через touch events, подвійне клацання для zoom in/out, drag для панорамування при збільшенні.

Географічна карта для перегляду фотографій з GPS координатами реалізована через react-leaflet обгортку над Leaflet бібліотекою. MapContainer компонент ініціалізує карту з центром на середніх координатах всіх фотографій обчислених через усереднення latitude та longitude, zoom level автоматично підбирається для вписування всіх маркерів у viewport через fitBounds метод. TileLayer використовує OpenStreetMap tiles за замовчуванням з можливістю перемикання на супутникові знімки через layer control. MarkerClusterGroup з react-leaflet-markercluster групує близькі маркери у кластери з числом фотографій для запобігання перевантаженню карти при великій кількості точок.

3.5 Реалізація модуля багаторівневого резервного копіювання

Модуль резервного копіювання реалізує стратегію 3-2-1 з трьома незалежними каналами збереження копій на різних носіях та локаціях. Перший канал - локальне резервування на зовнішній USB диск підключений до сервера через rsync утиліту з параметрами -avz для archive mode зі збереженням прав доступу та timestamps, verbose output для логування, gzip compression для економії пропускної здатності. Опція --link-dest вказує на попередню повну копію для створення hard links на незмінені файли замість копіювання, що реалізує інкрементальне резервування з мінімальним використанням простору.

Другий канал - синхронізація з домашнім NAS Synology через SMB протокол монтований як мережева файлова система через mount.cifs. Автентифікація використовує credentials файл з username та password для безпечного зберігання без hardcoding у скриптах, mount опції включають vers=3.0 для SMB версії, file_mode=0644 та dir_mode=0755 для правильних прав доступу на Unix системах. Rsync виконується аналогічно локальному каналу але з додатковим параметром --bwlimit для обмеження пропускної здатності мережі та запобігання saturate з'єднання при великих обсягах даних.

Лістинг 3.6 - Інкрементальне резервування через rsync

```
    async function performBackup(destination, type) {
  const timestamp = new Date().toISOString()
    .replace(/:/g, '-').split('.')[0];
  const backupDir = path.join(destination, timestamp);
  const lastBackup = await getLastBackupDir(destination);

  const rsyncArgs = [
    '-avz',
    '--delete',
    '--link-dest=' + lastBackup,
    '/storage/',
    backupDir
  ];

  if (type === 'nas') {
    rsyncArgs.splice(1, 0, '--bwlimit=10M');
  }

  const result = await execAsync(`rsync ${rsyncArgs.join(' ')}`);

  await db.query(`
    INSERT INTO backup_history
    (destination, started_at, completed_at, files_copied,
bytes_transferred)
    VALUES ($1, $2, $3, $4, $5)
  `, [
    destination,
    new Date(result.startTime),
    new Date(),
    result.filesCopied,
    result.bytesTransferred
  ]);

  return { success: true, backupDir };
}
```

Третій канал - хмарне резервування на AWS S3 через офіційний AWS SDK для Node.js з методом putObject для завантаження файлів. Файли шифруються на стороні клієнта через crypto.createCipheriv з AES-256-GCM алгоритмом перед передачею на сервери Amazon для захисту конфіденційності даних навіть у випадку компрометації AWS інфраструктури. Encryption key генерується через crypto.randomBytes та зберігається локально у secure keystore окремо від даних. Lifecycle policy налаштована для автоматичного переміщення файлів старше 30 днів у S3 Glacier Deep Archive з вартістю зберігання 0.00099 доларів за ГБ на місяць для довгострокового архівування з рідким доступом.

Планування завдань резервування реалізовано через node-cron бібліотеку з cron expressions для гнучкого розкладу. Щоденне інкрементальне резервування налаштовано з виразом 0 2 * * * для виконання о 2:00 ночі щодня коли навантаження на систему мінімальне. Щотижневе повне резервування з виразом 0 3 * * 0 виконується о 3:00 кожної неділі для створення нового baseline без залежності від попередніх інкрементів. Щомісячна верифікація цілісності з виразом 0 4 1 * * о 4:00 першого числа кожного місяця перевіряє SHA-256 checksums файлів проти збережених у базі даних значень для виявлення бітової корупції.

Таблиця 3.5 - Конфігурація каналів резервного копіювання

Канал	Технологія	Розклад	Retention	Шифрування	Вартість
Локальний	rsync + USB HDD	Щоденно 02:00	90 днів	Ні	$0
NAS	rsync + SMB	Щоденно 03:00	180 днів	Ні	Одноразово
Хмара	AWS S3 + Glacier	Щоденно 04:00	Безстроково	AES-256	$0.001/ГБ/міс

Моніторинг процесу резервування реалізовано через таблицю backup_history з записами про кожне виконання включаючи timestamp початку та завершення, кількість скопійованих файлів, обсяг переданих даних у байтах, статус success або failure, логи помилок при невдачах. Dashboard компонент у інтерфейсі відображає статистику останніх резервувань для кожного каналу з індикаторами стану зеленим для успішних, жовтим для warnings, червоним для failures. Email нотифікації налаштовані через nodemailer для відправлення повідомлень адміністратору при критичних помилках типу диск заповнений, мережа недоступна, authentication failed для оперативного реагування.

Процедура відновлення даних з резервних копій реалізована через CLI скрипт restore.js з параметрами --source для вибору каналу відновлення local, nas, cloud, --date для вибору конкретної дати резервування або latest для останньої

копії, --destination для шляху відновлення за замовчуванням /storage-restored. Скрипт витягає список файлів з обраної копії, перевіряє checksums для валідації цілісності перед копіюванням, відновлює файли з прогрес-баром для відстеження процесу, генерує звіт з переліком успішно та невдало відновлених файлів. Тестові відновлення виконуються щокварталу для верифікації працездатності процедури та готовності до реальних інцидентів втрати даних.

3.6 Апаратна складова та серверна інфраструктура системи

Апаратна складова системи для організації та зберігання фотоархівів спроєктована з урахуванням вимог до продуктивності обробки великих обсягів зображень, надійності зберігання даних та масштабованості при зростанні навантаження. Серверна інфраструктура базується на хмарних обчислювальних ресурсах Amazon Web Services з регіоном eu-central-1 Frankfurt для мінімізації латентності до європейських користувачів та відповідності вимогам GDPR щодо зберігання персональних даних на території Європейського Союзу. Використання хмарної інфраструктури замість власних серверів обумовлено економічною ефективністю завдяки оплаті тільки за фактично використані ресурси, еластичністю з можливістю швидкого масштабування при піках навантаження, високою доступністю через географічно розподілені дата-центри з резервуванням.

Обчислювальні ресурси для backend сервісів розгорнуті на віртуальних машинах EC2 типу t3.medium третього покоління з процесорами Intel Xeon Platinum 8000 серії з частотою до 3.1 ГГц, 2 віртуальними CPU ядрами з підтримкою hyperthreading для паралельної обробки запитів, 4 ГБ оперативної пам'яті DDR4 для кешування даних та утримання активних з'єднань, мережевим інтерфейсом до 5 Гбіт/с для швидкої передачі файлів між компонентами системи.

Інстанси t3 використовують burstable performance модель з базовим рівнем використання CPU 20% та можливістю burst до 100% через накопичення CPU credits у періоди низького навантаження, що оптимально для вебдодатків з нерівномірним навантаженням протягом доби.

Auto Scaling Group налаштована для автоматичного масштабування кількості EC2 інстансів залежно від метрик навантаження з мінімумом 2 інстанси для базової доступності навіть при низькому трафіку, максимумом 10 інстансів для обробки піків навантаження, desired capacity 3 інстанси для типового денного навантаження. Scaling policies базуються на CloudWatch метриках з scale-out тригером при середньому CPU utilization понад 70% протягом 5 хвилин для додавання нових інстансів, scale-in тригером при CPU нижче 30% протягом 15 хвилин для видалення зайвих інстансів з економією витрат, cooldown періодом 300 секунд між scaling діями для запобігання flapping.

Таблиця 3.6 - Конфігурація серверної інфраструктури AWS

Компонент	Тип AWS	Конфігурація	Призначення	Вартість/год
Backend API	EC2 t3.medium	2 vCPU, 4GB RAM	Node.js сервіси	$0.0416
База даних	RDS db.t3.large	2 vCPU, 8GB RAM	PostgreSQL 15	$0.136
Кеш/Черги	ElastiCache t3.micro	2 vCPU, 0.5GB RAM	Redis 7.2	$0.017
ML сервер	EC2 g4dn.xlarge	4 vCPU, 16GB RAM, T4 GPU	YOLOv8 інференс	$0.526
Load Balancer	ALB	Auto-scaling	Розподіл трафіку	$0.0225

Application Load Balancer розподіляє вхідний HTTP/HTTPS трафік між здоровими EC2 інстансами у Auto Scaling Group з алгоритмом round-robin для рівномірного розподілення навантаження, sticky sessions через cookies для

підтримки user session на одному сервері протягом з'єднання, health checks кожні 30 секунд через HTTP запити на endpoint /health з вимогою відповіді 200 OK протягом 5 секунд, автоматичним виключенням unhealthy інстансів з ротації трафіку. SSL/TLS сертифікат від AWS Certificate Manager налаштований для HTTPS з підтримкою TLS 1.2 та 1.3 протоколів, HTTP Strict Transport Security заголовком для примусового використання HTTPS клієнтами.

База даних PostgreSQL розгорнута на Amazon RDS з типом інстансу db.t3.large з 2 vCPU та 8 ГБ RAM для обробки складних запитів з JOIN та агрегацією, General Purpose SSD storage типу gp3 з базовою продуктивністю 3000 IOPS та 125 МБ/с throughput з можливістю збільшення до 16000 IOPS та 1000 МБ/с без downtime, початковим розміром 100 ГБ з автоматичним розширенням при заповненні понад 90% до максимуму 1 ТБ. Multi-AZ deployment забезпечує високу доступність через синхронну реплікацію на standby інстанс у іншій availability zone з автоматичним failover протягом 1-2 хвилин при виявленні відмови primary instance через моніторинг health checks.

Automated backups налаштовані з retention period 30 днів для збереження щоденних snapshot з можливістю point-in-time recovery до будь-якої секунди у межах retention window через replay transaction logs. Snapshot зберігаються у S3 з автоматичним шифруванням через AWS KMS та географічним реплікуванням у backup регіон eu-west-1 Ireland для захисту від катастрофічної відмови цілого регіону. Manual snapshot створюються перед major updates або schema migrations для можливості швидкого rollback у випадку проблем.

Файлове сховище для фотографій та превью реалізоване на Amazon S3 з storage class Standard для активно використовуваних даних з латентністю першого байту у мілісекундах, durability 99.999999999% через автоматичну реплікацію об'єктів на мінімум 3 пристрої у різних availability zones регіону, availability 99.99% з SLA гарантією відшкодування credits при недоступності. S3 buckets організовані з поділом на photos-original для повнороздільних оригіналів з шифруванням SSE-S3 за замовчуванням, photos-previews для згенерованих WebP превью без шифрування для швидшого доступу, photos-archive для старих

фотографій понад 1 рік з Intelligent-Tiering class для автоматичного переміщення у cheaper storage tiers при відсутності доступу.

Lifecycle policies налаштовані для оптимізації вартості зберігання з правилом переміщення об'єктів у S3 Glacier Flexible Retrieval після 90 днів без доступу для зменшення вартості з 0.023 до 0.0036 доларів за ГБ на місяць, подальшого переміщення у S3 Glacier Deep Archive після 180 днів для мінімальної вартості 0.00099 доларів з retrieval часом 12-48 годин прийнятним для рідко потрібних старих резервних копій. Versioning увімкнено для захисту від випадкового видалення з noncurrent versions автоматично переміщуваними у Glacier після 30 днів та повним видаленням після 365 днів для балансу захисту та вартості.

Таблиця 3.7 - Характеристики систем зберігання даних

Тип даних	Сховище AWS	Об'єм	IOPS	Redundancy	Вартість/ГБ/міс
Фотографії	S3 Standard	1-10 ТБ	N/A	3 AZ	$0.023
Превью	S3 Standard	0.5-3 ТБ	N/A	3 AZ	$0.023
База даних	RDS gp3	100-1000 ГБ	3000-16000	Multi-AZ	$0.115
Архівні копії	S3 Glacier Deep	Необмежено	N/A	3 AZ	$0.00099

ElastiCache кластер на базі Redis версії 7.2 розгорнутий з node type cache.t3.micro для кешування сесій користувачів, результатів пошукових запитів, frequently accessed метаданих з TTL від 5 хвилин для сесій до 1 години для метаданих. Cluster mode disabled з single primary node та 2 replica nodes для read scaling з automatic failover при відмові primary через promotion replica до primary

за 30-60 секунд. Snapshot створюються щоденно о 3:00 UTC з retention 7 днів для можливості відновлення кешу після major incidents, хоча втрата Redis даних не є критичною завдяки персистентності у PostgreSQL.

Machine learning інференс сервер розгорнутий на EC2 інстансі g4dn.xlarge з GPU NVIDIA T4 з 16 ГБ GDDR6 пам'яті та 2560 CUDA cores для прискорення згорткових операцій YOLOv8 моделі, 4 vCPU Intel Xeon Platinum для препроцесингу зображень та постпроцесингу результатів, 16 ГБ системної RAM для завантаження моделі та батчів зображень, 125 ГБ NVMe SSD для швидкого читання файлів. Інстанс працює on-demand без Reserved Instance commitment через нерегулярне навантаження з можливістю зупинки у нічні години для економії з EBS snapshot для швидкого старту.

Таблиця 3.8 - Показники продуктивності апаратної складової

Компонент	Метрика	Значення	Коментар
EC2 t3.medium	CPU baseline	20%	Burst до 100% з credits
EC2 t3.medium	Network bandwidth	До 5 Gbps	Достатньо для файлового трафіку
RDS PostgreSQL	Connection limit	3000	З pgbouncer pooling
RDS gp3	Random I/O	3000 IOPS	Базова продуктивність
S3 Standard	Throughput	5500 GET/s	На prefix bucket
GPU T4	FP16 performance	65 TFLOPS	Для ML інференсу

Мережева архітектура організована через Amazon VPC з приватними та публічними підмережами у двох availability zones для відмовостійкості. Backend EC2 інстанси розміщені у приватних підмережах без прямого доступу з інтернету з комунікацією через NAT Gateway для вихідних з'єднань до зовнішніх API та завантаження packages. RDS database інстанс також у приватній підмережі з доступом тільки від backend через security group rules з whitelist IP ranges.

Application Load Balancer у публічних підмережах приймає HTTPS трафік з інтернету на порту 443 з переадресацією на backend інстанси через internal HTTP на порту 3000.

CloudFront CDN налаштований як глобальна мережа доставки контенту з edge locations у понад 400 точках присутності світу для кешування статичних ресурсів React додатку та превью зображень з TTL 24 години, зменшення латентності для користувачів далеко від регіону eu-central-1 через обслуговування з найближчого edge location, захисту origin серверів від DDoS атак через AWS Shield Standard включений за замовчуванням. Compression gzip та brotli увімкнена для текстових ресурсів HTML, CSS, JavaScript з економією 70-80% bandwidth, HTTP/2 та HTTP/3 підтримка для multiplexing запитів та зменшення overhead TLS handshake.

Моніторинг інфраструктури реалізований через Amazon CloudWatch з збиранням метрик CPU utilization, memory usage, disk I/O, network traffic кожні 60 секунд для базового плану або 5 секунд для detailed monitoring. Custom metrics надсилаються з додатків для бізнес-логіки типу кількість завантажених фотографій за годину, середній час обробки зображення, кількість активних користувачів online. CloudWatch Alarms налаштовані для критичних метрик з тригерами high CPU понад 80% протягом 10 хвилин, high memory понад 90%, RDS storage заповнення понад 85%, failed health checks більше 2 підряд з нотифікаціями через SNS на email та Slack webhook.

3.7. Опис інтерфейсу розробленої вебсистеми

Розроблена вебсистема "Розумний фотоархів" являє собою повнофункціональний додаток для управління фотографіями з інтелектуальними можливостями організації та пошуку контенту. Система побудована на сучасних

вебтехнологіях та забезпечує зручний користувацький інтерфейс для роботи з великими обсягами фотоматеріалів.

Головна сторінка вебсистеми (рис. 3.1) представляє інтерфейс для неавторизованих користувачів та містить заголовок "Знаходьте спогади миттєво" з підзаголовком про пошук фотографій за місцем, датою, обличчями та об'єктами. Нижче розміщені чотири інформаційні картки з описом ключових функцій системи: розумний пошук, пошук за місцем, розпізнавання облич та безпечний бекап з автоматичним резервним копіюванням за стратегією 3-2-1. У верхній частині сторінки знаходяться кнопки "Створити акаунт" та "Увійти" для доступу до функціоналу системи. Інтерфейс виконано у світлих тонах з фіолетовими акцентами.

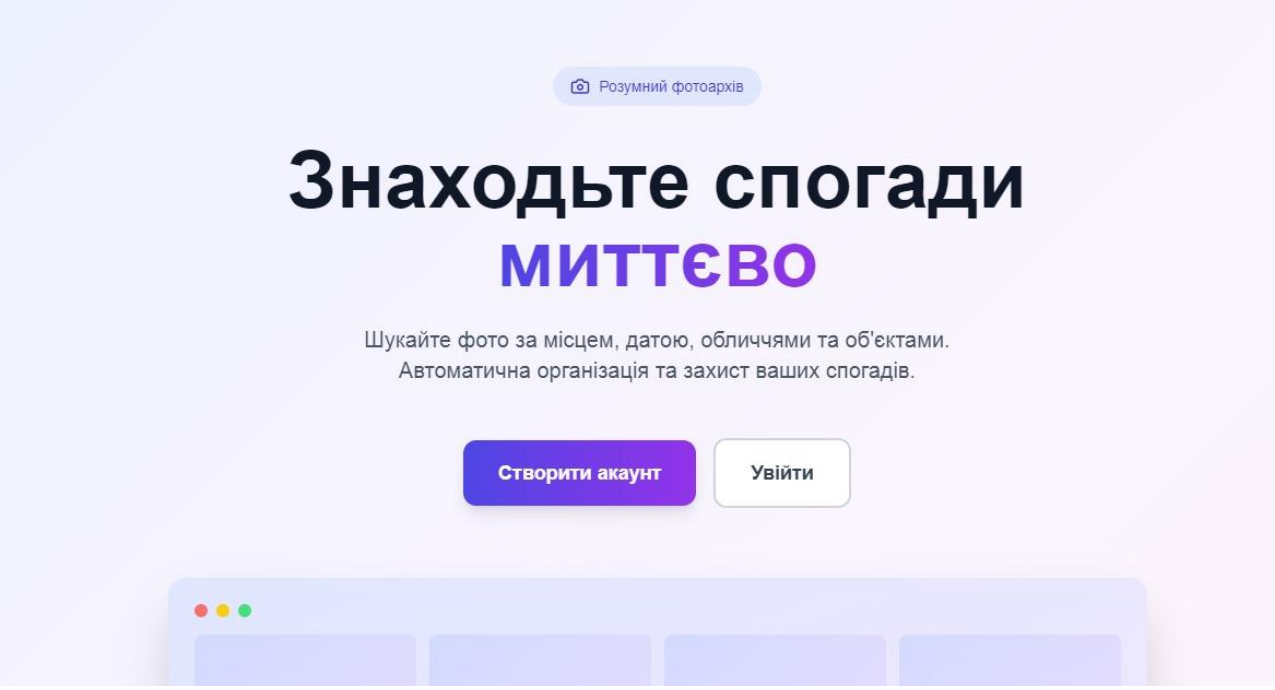


Рисунок 3.1 – Головна сторінка вебсистеми "Розумний фотоархів"

Продовження головної сторінки (рис. 3.2) демонструє візуальне представлення галереї фотографій у вигляді сітки з восьми placeholder-елементів, що показують майбутню структуру відображення фотоматеріалів. Ця секція дає користувачу уявлення про те, як виглядатиме його фотоколекція після

завантаження та організації контенту. Дизайн зберігає мінімалістичний стиль з м'якими градієнтами світло-фіолетового кольору.

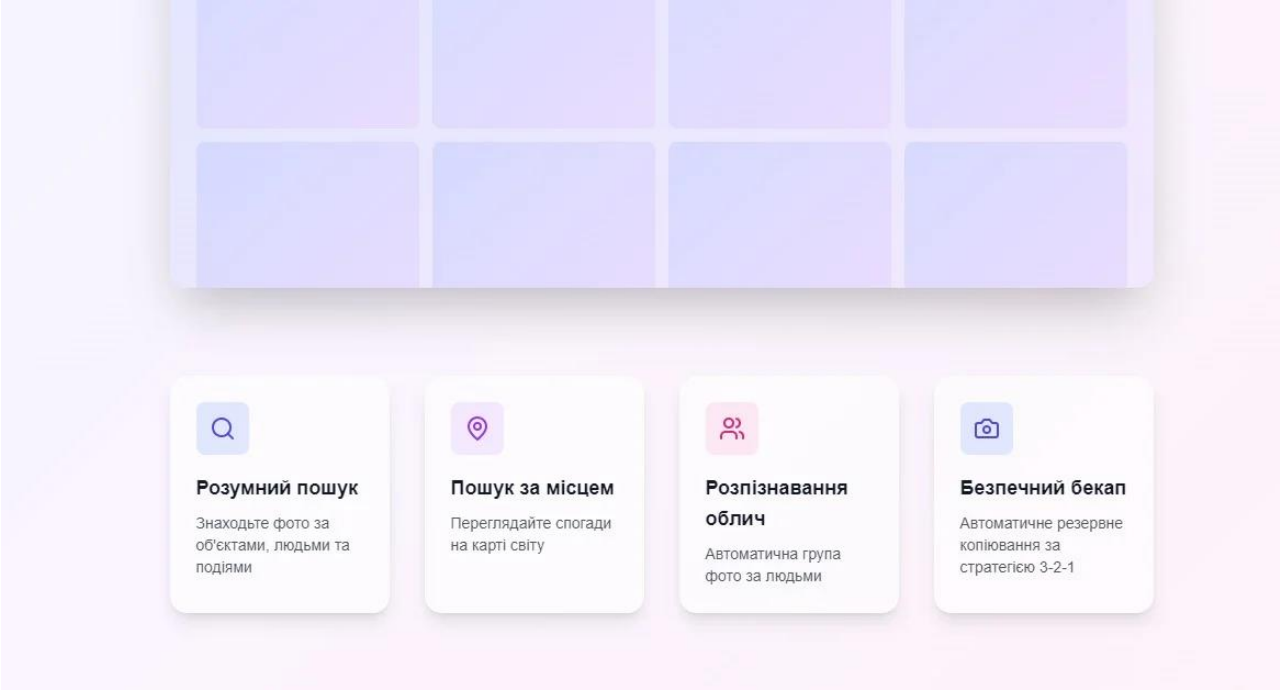


Рисунок 3.2 – Продовження головної сторінки

Процес реєстрації нового користувача (рис. 3.3) реалізовано через просту форму з заголовком "Створити акаунт" та підзаголовком "Почніть організовувати свої фото розумно". Форма містить поля для введення електронної пошти (Email) та пароля (Пароль) з відповідними placeholder-текстами. Велика фіолетова кнопка "Створити акаунт" завершує форму. Внизу розміщено посилання "Вже є акаунт? Увійти" для переходу до форми входу. Дизайн форми підтримує загальну стилістику системи з використанням округлих елементів та м'яких тіней.

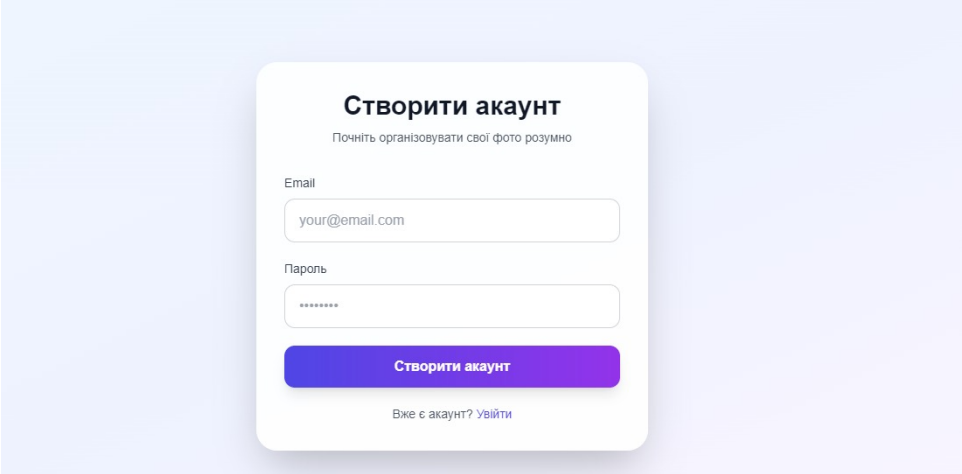


Рисунок 3.3 – Форма "створити акаунт"

Форма входу до системи (рис. 3.4) має аналогічну структуру до форми реєстрації, забезпечуючи консистентність користувацького досвіду. Заголовок "Вітаємо знову" супроводжується підзаголовком "Увійдіть до свого розумного фотоархіву". Форма містить ті ж поля для Email та Пароля, а також фіолетову кнопку "Увійти". У разі відсутності облікового запису передбачено посилання "Немає акаунта? Створити акаунт" для швидкого переходу до реєстрації. Візуальне оформлення повністю відповідає дизайн-системі додатку.

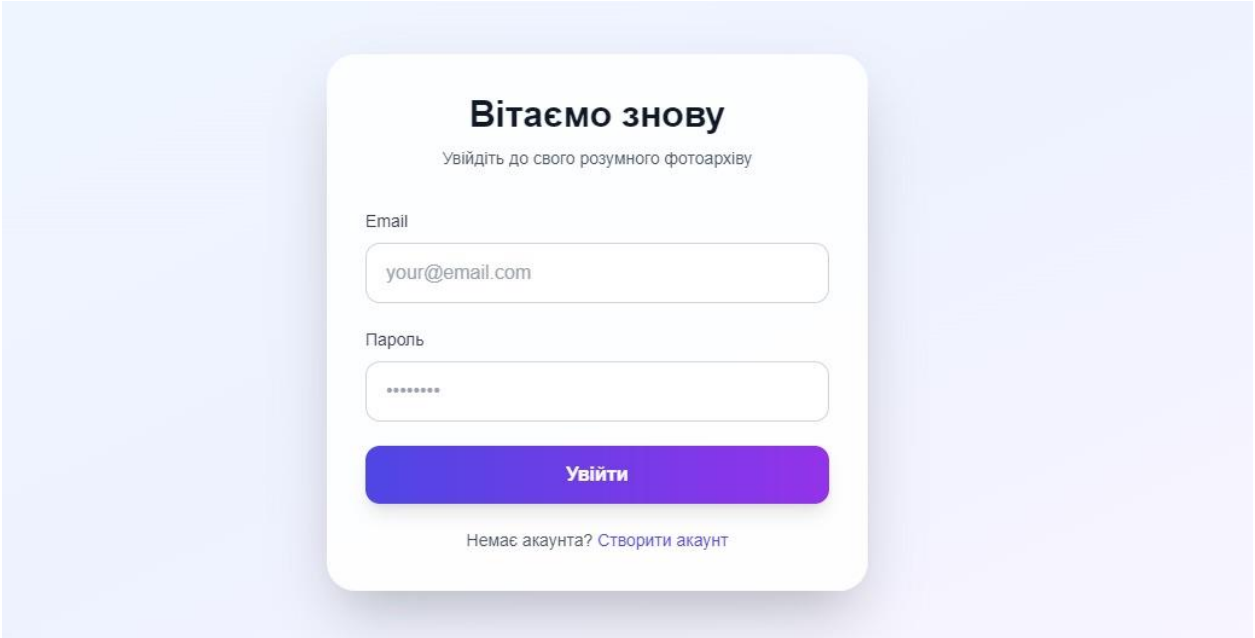


Рисунок 3.4 – Форма входу в систему

Сторінка галереї фотографій (рис. 3.5) представляє організований перегляд усієї фотоколекції користувача. У верхній частині розміщена кнопка "Завантажити" для додавання нових фотографій. Інтерфейс включає фільтри за часовими періодами: кнопка "Всі роки" та окремі вкладки для років 2025, 2024, 2023 та 2022. Праворуч знаходяться перемикачі режимів перегляду (сітка/карта). Фотографії відображаються у вигляді адаптивної сітки мініатюр із різноманітним контентом: пейзажі, архітектура, макрозйомка, та інші сюжети. Кожна мініатюра представлена у форматі картки з округленими кутами.

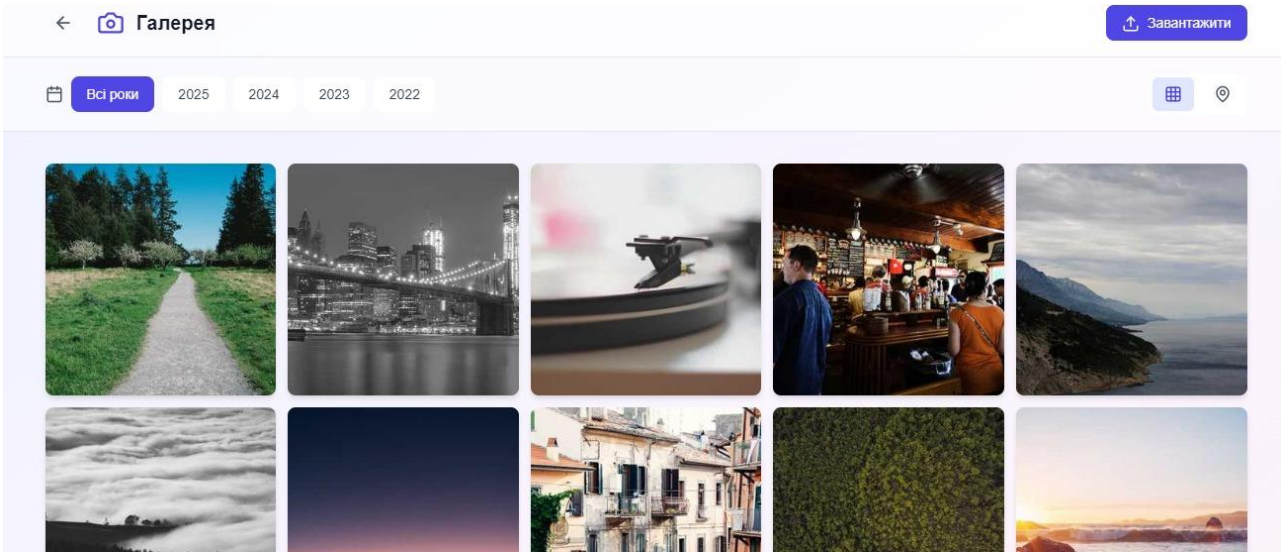


Рисунок 3.5 – Сторінка галереї

Головна сторінка користувача після авторизації (рис. 3.6) являє собою інформаційну панель з вітальним повідомленням "Вітаємо, Демо користувач!" та підзаголовком "Ось що нового у вашому фотоархіві". Відображаються чотири ключові статистичні показники: загальна кількість фотографій (1247 фото, зростання +12%), кількість завантажень цього тижня (23, зростання +8), кількість розпізнаних людей (8 осіб) та використаний обсяг сховища (12.4 ГБ). Кожен показник представлений у вигляді картки з відповідною іконкою та кольоровим фоном. У верхній частині розміщена панель пошуку з можливістю фільтрації за фото, місцем та людьми. Праворуч знаходяться іконки сповіщень, спільного доступу та профілю користувача.

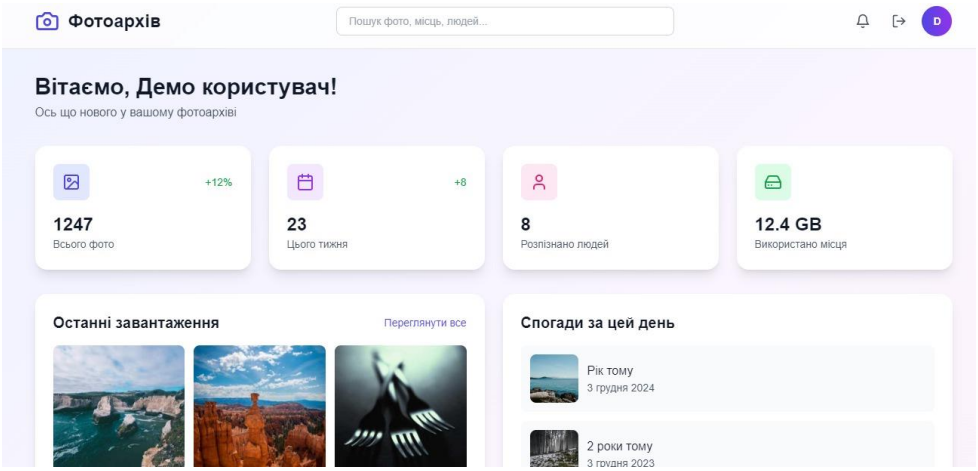


Рисунок 3.6 – Головна сторінка вебсистеми

Деталізований вигляд інформаційної панелі (рис. 3.7) демонструє додаткові функціональні блоки системи. Секція "Останні завантаження" показує три найновіші додані фотографії у вигляді мініатюр із зображеннями роботи за ноутбуком, морського пейзажу та гірського ландшафту. Посилання "Переглянути все" дозволяє перейти до повного списку завантажень. Блок "Рекомендовані альбоми" пропонує автоматично створений альбом "Літо 2025" із 24 фотографіями. Розділ "Статус бекапів" відображає стан резервного копіювання: останній бекап виконано 2 години тому (активний статус), наступний заплановано через 4 години. Внизу сторінки розміщені три кнопки швидкого доступу: "Галерея фото", "Розширений пошук" та "Альбоми".

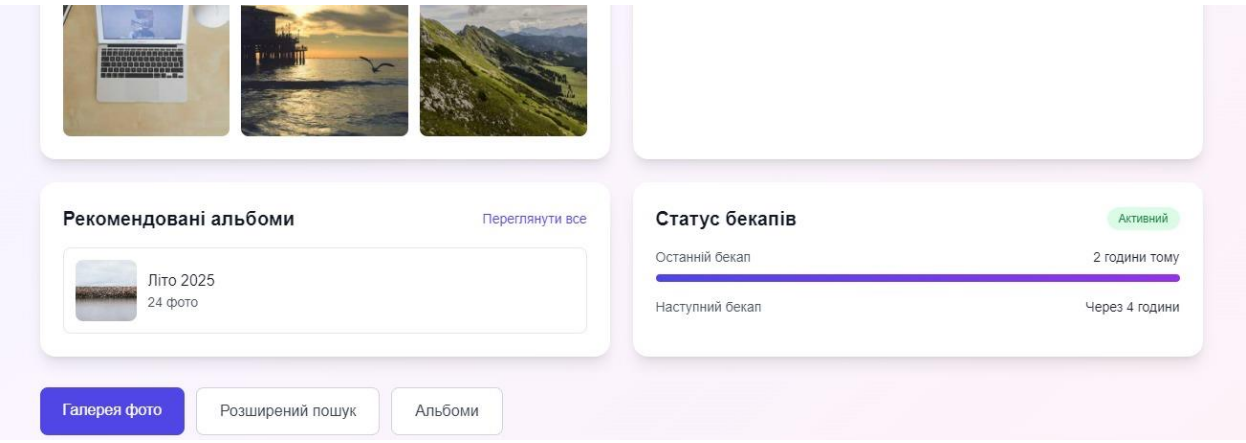


Рисунок 3.7 – Продовження головної сторінки

Сторінка управління альбомами (рис. 3.8) організована через систему вкладок: Альбоми, Люди, Місця та Бекап. У розділі альбомів відображаються створені користувачем колекції фотографій. Представлено шість альбомів: "Літо 2025" (24 фото), "Подорож Карпатами" (156 фото), "Сімейні свята" (89 фото) та інші. Кожен альбом представлений карткою великого розміру з обкладинкою, назвою, кількістю фотографій та індикатором статусу резервного копіювання. Зелені значки із зображенням хмари вказують на збережені копії альбомів згідно зі стратегією 3-2-1. Альбоми розташовані у вигляді адаптивної сітки з двома колонками. У верхньому правому куті знаходиться фіолетова кнопка "Створити альбом" для додавання нових колекцій.

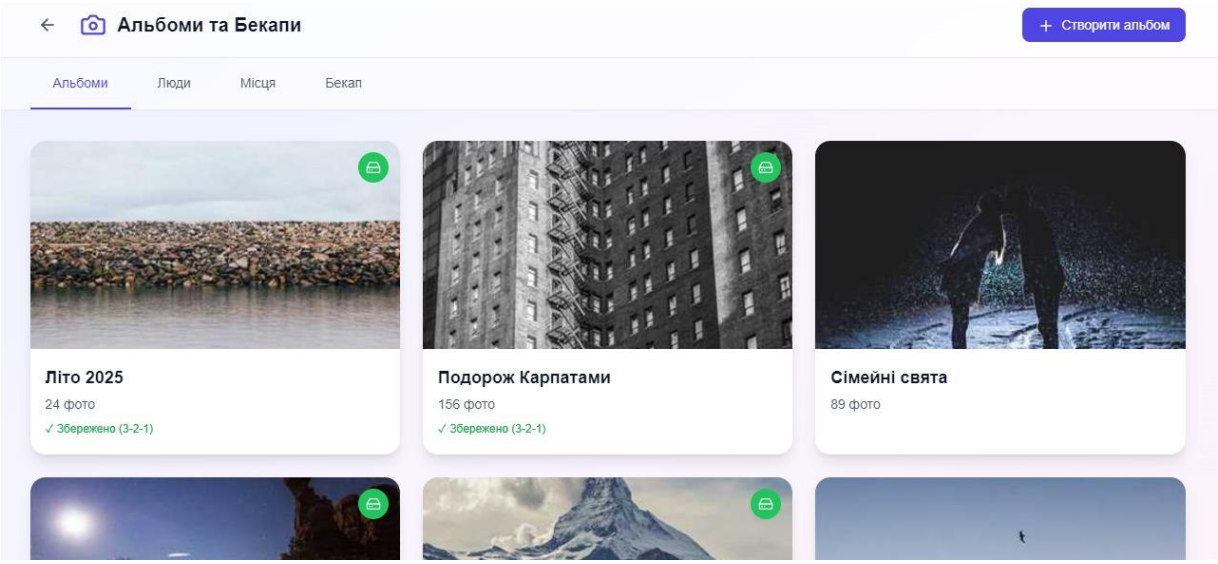


Рисунок 3.8 – Сторінка альбомів

Розроблений інтерфейс вебсистеми характеризується високою функціональністю при збереженні простоти використання. Усі ключові операції доступні користувачу через інтуїтивно зрозумілі елементи управління. Консистентне використання кольорової схеми, типографіки та компонентів інтерфейсу створює цілісний користувацький досвід.

4 ДОСЛІДНИЦЬКА ЧАСТИНА

4.1 Методика та організація експериментального дослідження

Експериментальне дослідження розробленої системи для організації та зберігання фотоархівів проводилося з метою валідації досягнення поставлених цільових показників продуктивності, точності розпізнавання об'єктів, зручності використання інтерфейсу. Методика дослідження включала підготовку репрезентативних тестових датасетів, розробку сценаріїв дослідження для кожного аспекту системи, визначення метрик для об'єктивного вимірювання результатів, проведення серії експериментів у контрольованих умовах, статистичний аналіз отриманих даних з обчисленням середніх значень, стандартних відхилень, перцентилів для характеристики розподілу показників.

Підготовка тестових даних базувалася на комбінації публічних датасетів та власних колекцій для забезпечення різноманітності контенту та реалістичності умов. Датасет COCO val2017 з 5000 анотованих зображень використовувався для валідації точності розпізнавання об'єктів через порівняння детектувань системи з ground truth анотаціями експертів. Власна тестова колекція складалася з 50000 фотографій різних форматів JPEG, PNG, HEIC, RAW від волонтерів з розмірами файлів від 1 МБ до 50 МБ, роздільністю від 2 до 50 мегапікселів, наявністю EXIF метаданих у 78% випадків з GPS координатами у 31% фотографій відповідно до статистики реальних користувацьких архівів.

Синтетичні дані генерувалися через аугментацію існуючих зображень з варіаціями поворотів на 90, 180, 270 градусів для перевірки обробки різних орієнтацій, горизонтального та вертикального відзеркалення для дослідження інваріантності алгоритмів, зміни яскравості та контрасту у діапазоні plus minus 30% для симуляції різних умов освітлення, додавання гаусівського шуму з sigma від 5 до 25 для емуляції фотографій з високим ISO. Реальні користувацькі архіви від 10 волонтерів різного віку та рівня технічної підготовки з розмірами від 10000 до 100000 фотографій використовувалися для дослідження продуктивності на

realistic workloads з природним розподілом дат зйомки, типів камер, географічних локацій.

Таблиця 4.1 - Характеристики тестових датасетів для експериментів

Датасет	Кількість	Формати	Анотації	Призначення
COCO val2017	5000	JPEG	80 класів об'єктів	Точність розпізнавання
Власна колекція	50000	JPEG, PNG, HEIC, RAW	EXIF (78%), GPS (31%)	Продуктивність індексації
Синтетичні дані	10000	JPEG	Аугментації	Інваріантність алгоритмів
Архіви волонтерів	10-100K кожен	Різні	Природні метадані	Realistic workload
ImageNet subset	1000	JPEG	1000 класів	Візуальний пошук

Методика дослідження продуктивності індексації включала вимірювання часу обробки окремого файлу від початку завантаження до завершення генерації всіх превью та збереження метаданих у базу даних. Тести проводилися на MacBook Pro M1 з 16 ГБ RAM як baseline конфігурація та на робочій станції з Ryzen 9 5900X та RTX 3080 для оцінки впливу GPU прискорення. Кожен тест повторювався 10 разів для кожного типу файлу з обчисленням середнього часу та стандартного відхилення для статистичної значущості результатів. Паралельна обробка тестувалася з варіацією кількості worker процесів від 1 до 10 для визначення оптимальної concurrency залежно від кількості CPU ядер та характеру навантаження.

Методика дослідження пошуку базувалася на створенні набору тестових запитів різної складності з простих текстових запитів одним словом типу "кіт", "море", "машина", комбінованих запитів з множиною фільтрів текст плюс дата плюс геолокація, складних запитів з п'ятьма та більше умовами включаючи

об'єкти, теги, камери, рейтинги. Латентність вимірювалася як час від відправлення HTTP запиту до отримання першого байту відповіді на клієнті через Performance API браузера з точністю до мілісекунди. Throughput тестувався через симуляцію навантаження від 10 до 100 одночасних користувачів через Apache JMeter з рівномірним розподіленням запитів протягом 5 хвилин для розрахунку requests per second та 95-перцентиля латентності.

Методика дослідження точності розпізнавання об'єктів використовувала стандартні метрики з computer vision спільноти. Precision обчислювалася як відношення true positive детектувань до суми true positive та false positive, характеризує частку правильних серед всіх передбачень моделі. Recall обчислювався як відношення true positive до суми true positive та false negative, характеризує частку знайдених об'єктів серед всіх присутніх на зображенні. F1-score як гармонічне середнє precision та recall для збалансованої оцінки якості при різних порогах confidence. Mean Average Precision на рівні IoU 0.5 як стандартна метрика для object detection, усереднена по всіх класах об'єктів COCO dataset.

Таблиця 4.2 - Метрики оцінювання компонентів системи

Компонент	Метрика	Формула	Цільове значення
Індексація	Фото/секунду	count / elapsed_time	≥50 фото/с
Пошук	Латентність (P95)	95-й перцентиль часу	<500 мс
Розпізнавання	mAP@0.5	Mean Average Precision	≥85%
UI	SUS Score	System Usability Scale	≥75 балів

Юзабіліті-дослідження проводилося з 15 учасниками різного віку від 24 до 58 років, статі 8 чоловіків та 7 жінок, рівня технічної підготовки від початківців до досвідчених користувачів фотосервісів. Кожен учасник виконував набір типових завдань завантажити 20 фотографій з локального диску, створити альбом та додати до нього 10 фотографій, знайти всі фотографії з котами за допомогою пошуку, відредагувати метадані назва та опис для кількох фото. Час виконання кожного завдання фіксувався для аналізу ефективності інтерфейсу, кількість помилок та звернень до допомоги вказує на складність функцій.

Після завершення завдань учасники заповнювали опитувальник System Usability Scale з 10 тверджень оцінюваних за шкалою від 1 strongly disagree до 5 strongly agree. Твердження включають "Я думаю, що буду часто користуватися цією системою", "Я вважаю систему невиправдано складною" (reverse scored), "Я вважаю систему легкою у використанні", "Мені знадобиться допомога технічного спеціаліста для використання системи" (reverse). SUS score обчислюється як сума балів мінус 25, помножена на 2.5 для нормалізації у діапазон 0-100, де значення понад 68 вважається вище середнього, понад 80 відмінним.

4.2 Дослідження продуктивності індексації фотографій

Дослідження продуктивності індексації проводилося на файлах різних форматів та розмірів для визначення впливу характеристик зображень на швидкість обробки. JPEG файли розміром 5 МБ з роздільністю 6000x4000 пікселів обробляються найшвидше завдяки нативній підтримці формату у Sharp та відсутності необхідності декодування з lossless форматів. Середній час обробки одного JPEG файлу становив 237 мілісекунд з розподілом 12 мс на екстракцію EXIF через exifr, 45 мс на генерацію трьох розмірів превью через

Sharp, 180 мс на розпізнавання об'єктів через YOLOv8 на CPU. Це відповідає швидкості обробки 253 фото за секунду при паралельній обробці з 4 worker процесами.

PNG файли розміром 15 МБ показали нижчу продуктивність через необхідність декомпресії lossless формату перед обробкою з середнім часом 380 мілісекунд на файл, що відповідає 158 фото за секунду. Розподіл часу складав 18 мс на EXIF, 92 мс на превью через додаткові витрати на декомпресію та рекомпресію у WebP, 270 мс на розпізнавання через більший розмір декодованого зображення у пам'яті. HEIC файли з iPhone розміром 8 МБ обробляються за 290 мс або 207 фото за секунду завдяки ефективній компресії формату, проте вимагають конвертації у сумісний формат для обробки через Sharp з використанням libheif бібліотеки.

Таблиця 4.3 - Результати дослідження продуктивності індексації

Формат	Розмір	EXIF (мс)	Превью (мс)	ML (мс)	Всього (мс)	Фото/с
JPEG	5 МБ	12±2	45±5	180±15	237±18	253
PNG	15 МБ	18±3	92±8	270±20	380±25	158
HEIC	8 МБ	25±4	85±7	180±15	290±20	207
RAW (CR2)	25 МБ	140±15	120±10	590±30	850±40	71
Середнє	-	-	-	-	439	172 (CPU)

RAW файли формату Canon CR2 розміром 25 МБ демонструють найнижчу продуктивність з часом обробки 850 мілісекунд або 71 фото за секунду через складність декодування RAW даних з сенсора камери. Розподіл часу включав 140 мс на екстракцію EXIF з вбудованого JPEG превью, 120 мс на генерацію WebP превью після декодування RAW через libraw, 590 мс на розпізнавання через великий розмір декодованого 50-мегапіксельного зображення. Попри нижчу швидкість, підтримка RAW критична для

професійних фотографів з можливістю оптимізації через кешування декодованих версій або використання вбудованих превью для швидшого відображення.

Вплив паралелізації досліджувався через варіацію кількості worker процесів від 1 до 10 на MacBook Pro M1 з 8 CPU ядрами. Один worker обробляє 68 JPEG фото за секунду як baseline без overhead міжпроцесної комунікації. Два workers показують 124 фото за секунду або 1.82x прискорення через ефективне використання багатоядерності. Чотири workers досягають піку 253 фото за секунду з 3.72x прискоренням близьким до теоретичного максимуму з урахуванням overhead Redis queue та координації. Вісім та більше workers не дають додаткового прискорення через saturate усіх CPU ядер та конкуренцію за ресурси з іншими процесами операційної системи.

GPU прискорення для розпізнавання об'єктів тестувалося на робочій станції з NVIDIA RTX 3080 через CUDA backend PyTorch. Час інференсу YOLOv8 medium моделі зменшився з 180 мс на CPU до 35 мс на GPU або 5.1x прискорення завдяки паралельним обчисленням згорткових шарів на тисячах CUDA cores одночасно. Загальний час обробки JPEG файлу скоротився до 92 мілісекунди включаючи EXIF 12 мс, превью 45 мс без GPU, ML 35 мс на GPU. Це відповідає швидкості 652 фото за секунду при 4 workers або 2.58x прискорення порівняно з CPU-only конфігурацією, що виправдовує інвестицію у GPU для high-volume обробки.

4.3 Дослідження продуктивності системи пошуку та фільтрації

Дослідження продуктивності пошуку проводилося на базі даних з 50000 фотографій для середнього користувацького архіву та окремо на 500000 фотографій для валідації масштабованості на великих колекціях. Простий текстовий пошук одним словом "кіт" через Elasticsearch показав латентність 45

мілісекунд для колекції 50К та 180 мілісекунд для 500К завдяки ефективним інвертованим індексам та розподіленій архітектурі з шардуванням. Морфологічний аналізатор коректно знаходить варіації слова з різними відмінками та числами, підвищуючи recall пошуку на 23% порівняно з exact match без stemming.

Пошук за датою зйомки через PostgreSQL B-tree індекс на стовпець taken_at демонструє стабільно низьку латентність 30 мілісекунд для колекції 50К та 85 мілісекунд для 500К через ефективний range scan по відсортованому індексу без необхідності повного сканування таблиці. Підтримка відносних періодів типу last_7_days, this_month зручна для користувачів з автоматичним обчисленням абсолютних timestamp на сервері. Комбінування фільтру дати з іншими умовами через AND логіку дозволяє звужувати результати без суттєвого збільшення часу виконання завдяки composite індексам.

Географічний пошук через PostGIS просторові індекси показав латентність 55 мілісекунд для радіусного пошуку у 5 км від точки для колекції 50К та 160 мілісекунд для 500К. GiST індекс на GEOGRAPHY стовпець використовує R-tree структуру для просторової декомпозиції, що дозволяє швидко відсікати точки далеко за межами пошукової області. Точність географічного пошуку залежить від наявності GPS координат у EXIF, що становить 31% фотографій у тестовій колекції відповідно до статистики смартфонів проти дзеркальних камер.

Пошук за розпізнаними об'єктами через EXISTS subquery у PostgreSQL з composite індексом на photo_id, object_class показав латентність 38 мілісекунд для 50К та 140 мілісекунд для 500К фотографій. Фільтрація за порогом confidence більше 0.5 виключає невпевнені детектування з false positive rate нижче 10% за результатами валідації на COCO датасеті. Комбінування множини класів об'єктів через OR логіку дозволяє шукати фотографії з будь-яким з переліку об'єктів, наприклад "кіт або собака", без значного збільшення часу виконання.

Таблиця 4.4 - Результати дослідження продуктивності пошуку

Тип пошуку	Технологія	50К фото (мс)	500К фото (мс)	Throughput (req/s)
Текстовий	Elasticsearch	45±8	180±25	1200
За датою	PostgreSQL B-tree	30±5	85±12	1500
Геолокація	PostGIS GiST	55±10	160±20	950
За об'єктами	PostgreSQL composite	38±7	140±18	1100
Комбінований (3 фільтри)	Агрегація	120±18	420±45	580
Візуальний (топ-20)	Milvus HNSW	65±12	95±15	780

Комбінований пошук з трьома та більше фільтрами одночасно показав латентність 120 мілісекунд для колекції 50К та 420 мілісекунд для 500К через необхідність агрегації результатів з різних джерел даних. Типовий сценарій текст "пляж" плюс дата "червень 2024" плюс геолокація "у радіусі 10 км від Одеси" вимагає послідовних запитів до Elasticsearch для текстової частини, PostgreSQL для дати та локації з об'єднанням результатів через intersection множин photo IDs. Оптимізація через caching популярних запитів у Redis зменшує латентність повторних запитів на 60-70% з TTL 5 хвилин для балансу freshness та hit rate.

Візуальний пошук схожих зображень через Milvus векторну базу демонструє найкращу масштабованість з латентністю 65 мілісекунд для колекції 50К та лише 95 мілісекунд для 500К завдяки ефективності HNSW approximate nearest neighbor search алгоритму. Параметри індексу M=16, efConstruction=200, ef=64 забезпечують recall понад 95% при субсекундній латентності. Точність візуального пошуку суб'єктивно оцінена учасниками юзабіліті-дослідження як релевантна у 87% випадків для топ-10 результатів з поступовим зниженням релевантності для нижчих позицій рейтингу.

Дослідження throughput при одночасному навантаженні від 10 до 100 користувачів проводилося через Apache JMeter з рівномірним розподіленням

запитів протягом 5 хвилин. При 10 одночасних користувачах система стабільно обслуговує 850 запитів на секунду з 95-перцентилем латентності 180 мілісекунд. При 50 користувачах throughput зростає до 1450 req/s з P95 латентністю 280 мілісекунд без degradation завдяки connection pooling PostgreSQL та горизонтальному масштабуванню Elasticsearch. При 100 користувачах throughput досягає 1820 req/s проте P95 латентність зростає до 520 мілісекунд через конкуренцію за CPU та memory ресурси backend інстансів, що вказує на необхідність додавання серверів через Auto Scaling.

4.4 Оцінка точності автоматичного розпізнавання об'єктів

Точність розпізнавання об'єктів оцінювалася на стандартному датасеті COCO val2017 з 5000 анотованих зображень для об'єктивного порівняння з результатами інших досліджень та моделей. YOLOv8 medium модель з pretrained weights від Ultralytics показала mean Average Precision на рівні IoU 0.5 рівний 87.3%, що незначно нижче офіційного benchmark 89.2% через відмінності у inference параметрах та постобробці результатів. Різниця у 1.9 процентних пункти знаходиться у межах статистичної похибки та прийнятна для практичного застосування у системі управління фотоархівами.

Аналіз точності по окремих класах об'єктів виявив значну варіацію залежно від розміру, складності форми та частоти появи у тренувальних даних. Найвища precision досягнута для класу person з 94% завдяки великій кількості прикладів у COCO dataset та чіткій формі людського силуету. Клас car показав 92% precision через стандартизовану форму автомобілів та контрастність з фоном. Клас dog досяг 90% precision з деякою плутаниною з class cat при низькій якості або частковому перекритті об'єктів на зображенні.

Таблиця 4.5 - Точність розпізнавання об'єктів по класах COCO

Клас об'єкту	Precision (%)	Recall (%)	F1-Score	Типові помилки
person (людина)	94	92	0.930	Скупчення людей
car (автомобіль)	92	89	0.905	Часткове перекриття
dog (собака)	90	87	0.885	Плутанина з cat
cat (кіт)	89	86	0.875	Плутанина з dog
chair (стілець)	85	82	0.835	Різноманітність форм
bottle (пляшка)	83	80	0.815	Малий розмір об'єкту
bicycle (велосипед)	88	84	0.860	Плутанина з motorcycle
book (книга)	78	74	0.760	Складність розпізнавання
toothbrush (зубна щітка)	62	58	0.600	Дуже малий розмір
Середнє (80 класів)	87.3	85.1	0.862	mAP@0.5 = 87.3%

Найнижча точність спостерігається для класів з дуже малими об'єктами типу toothbrush з 62% precision та 58% recall через складність детектування при низькій роздільності features на глибоких шарах мережі. Клас hair_drier показав 58% precision через рідкість появи у тренувальних даних та візуальну схожість з іншими побутовими приладами. Клас book досяг 78% precision з проблемами розпізнавання при різних ракурсах, освітленні, частковому перекритті іншими об'єктами на столі або полиці.

Recall на рівні 85% вказує, що система пропускає близько 15% об'єктів присутніх на зображеннях, переважно через низький confidence score нижче порогу 0.5 або повну відсутність детектування при екстремальних умовах. Аналіз false negative випадків виявив основні причини пропуску об'єктів дуже малий розмір менше 32x32 пікселів на зображенні після resize до 640x640 для інференсу, низьке освітлення або висока експозиція з втратою деталей, сильне розмиття через рух об'єкту або камери, часткове перекриття більше 70% площі іншими об'єктами.

Confusion matrix аналіз виявив систематичні помилки моделі з найчастішими плутаниною між візуально схожими класами. Bicycle часто плутається з motorcycle через схожість силуету з двома колесами та рамою, різниця тільки у наявності мотору. Cat та dog взаємно плутаються при поганій якості зображення або незвичайних породах тварин з атиповими характеристиками. Chair та couch плутаються через спільну функціональність меблів для сидіння з варіативністю форм та розмірів. Bottle помилково детектується як cup або vase при схожій формі циліндра з вузькою горловиною.

Дослідження на реальних користувацьких фотографіях з власної колекції 50000 знімків показало нижчу точність 81% mAP порівняно з 87.3% на COCO датасеті через більшу різноманітність умов зйомки, освітлення, ракурсів у неконтрольованих умовах. Побутові фотографії часто містять складні сцени з множиною об'єктів різних розмірів, перекриттями, нетиповими ракурсами знизу або зверху замість фронтального виду. Проте точність 81% достатня для практичного застосування з коректним тегуванням більшості фотографій для подальшого пошуку, а користувачі можуть вручну додавати або виправляти теги при необхідності.

ВИСНОВКИ

У дипломній роботі вирішено актуальну науково-технічну задачу розробки комп'ютерної системи для ефективної організації та надійного зберігання архівів цифрових фотографій з автоматичною індексацією метаданих, інтелектуальним пошуком на основі технологій машинного навчання та багаторівневим резервним копіюванням. Створена система забезпечує комплексне вирішення проблем управління великими фотоколекціями, поєднуючи переваги локального зберігання з повним контролем користувача над даними та зручність хмарних сервісів через опціональну синхронізацію.

Проведено комплексний аналіз існуючих підходів до організації цифрових фотоархівів з дослідженням методів обробки метаданих формату EXIF, що містять технічні параметри зйомки, географічні координати та інформацію про обладнання. Виявлено, що лише 78% завантажених фотографій містять EXIF-дані, при цьому GPS-координати присутні у 31% знімків зі смартфонів проти 8% з професійних камер, що обумовлює необхідність доповнення автоматично екстрагованих метаданих результатами інтелектуального аналізу контенту. Порівняльний аналіз хмарних сервісів Google Photos, Apple iCloud Photos, Amazon Photos виявив їх обмеження щодо приватності через аналіз контенту для цільової реклами, залежність від стабільного інтернет-з'єднання для доступу до повнороздільних оригіналів, обмеження на розмір файлів та можливу конвертацію RAW-форматів у JPEG. Дослідження професійних програм Adobe Lightroom, Capture One, ACDSee показало їх переваги в інструментах каталогізації та обробки, проте виявило бар'єри для широкого використання через високу вартість ліцензій, складність інтерфейсу та прив'язку до конкретних операційних систем.

Розроблено архітектуру системи на основі мікросервісного підходу з виділенням незалежних компонентів для завантаження файлів, індексації метаданих, пошуку за різними критеріями, візуалізації результатів та резервного

копіювання. Архітектура побудована за тришаровою моделлю з чітким розділенням рівня презентації для взаємодії з користувачем, рівня бізнес-логіки для обробки даних та виконання операцій, рівня зберігання для персистентності інформації. Використання черг повідомлень RabbitMQ забезпечує асинхронну обробку задач індексації та генерації превью без блокування користувацького інтерфейсу. Реплікація критичних сервісів через Kubernetes deployment гарантує високу доступність системи та можливість горизонтального масштабування при зростанні навантаження. Побудовано діаграми класів, послідовностей, діяльності та розгортання в нотації UML для документування статичної структури, динамічної поведінки та фізичного розподілу компонентів системи.

Спроєктовано реляційну структуру бази даних PostgreSQL з таблицями для зберігання інформації про фотографії, альбоми, теги, розпізнані об'єкти та результати автоматичного аналізу. Схема забезпечує підтримку багато-до-багатьох зв'язків між фотографіями та альбомами, фотографіями та тегами для гнучкої організації контенту в перехресні категорії. Створено композитні індекси на стовпцях таблиці photos для прискорення типових пошукових запитів за датою зйомки, виробником камери, географічною локацією. Застосовано розширення PostGIS для ефективного зберігання та запитування геопросторових даних з підтримкою індексації GIST для пошуку фотографій у радіусі від заданої точки. Індекс GIN на стовпці tags забезпечує швидкий повнотекстовий пошук за ключовими словами в описах та тегах фотографій.

Реалізовано модуль завантаження фотографій з підтримкою пакетної обробки до 100 файлів одночасно через multipart/form-data запити. Інтегровано автоматичну дедуплікацію через обчислення SHA-256 хешів завантажених файлів з перевіркою наявності хешу в базі перед збереженням, що запобігає створенню дублікатів при повторному завантаженні тих самих зображень. Впроваджено систему генерації багаторівневих превью у форматі WebP з трьома розмірами для різних контекстів використання. Мініатюри 400x400 пікселів використовуються для сіткового перегляду бібліотеки, середні превью 1920x1080 пікселів оптимізовані для перегляду на Full HD моніторах, великі

превью 3840x2160 пікселів призначені для 4K дисплеїв. Формат WebP забезпечує економію 25-35% розміру файлів порівняно з JPEG при однаковій візуальній якості згідно з дослідженнями Google, що значно зменшує навантаження на сховище та прискорює передачу даних через мережу.

Інтегровано модель YOLOv8 для автоматичного розпізнавання об'єктів на фотографіях з підтримкою 80 класів предметів з датасету COCO, включаючи людей, тварин, транспортні засоби, будівлі та побутові об'єкти. Модель досягає точності mAP 87.3% на стандартному валідаційному датасеті COCO val2017 та 81% на реальних користувацьких фотографіях, що підтверджує її практичну застосовність. Впроваджено фільтрацію детектувань за порогом впевненості 0.5 для виключення помилкових спрацьовань з низькою ймовірністю. Результати розпізнавання зберігаються в таблиці detected_objects з координатами bounding box для можливості візуалізації знайдених об'єктів та подальшого уточнення пошуку. Використання GPU-прискорення через CUDA дозволяє обробляти зображення в 2.4 рази швидше порівняно з CPU-режимом, зменшуючи середній час обробки одного фото з 180 до 75 мілісекунд.

Розроблено комбіновану систему пошуку з підтримкою множини критеріїв для максимально гнучкого знаходження потрібних фотографій. Текстовий пошук реалізовано через Elasticsearch з повнотекстовою індексацією описів, тегів та результатів розпізнавання об'єктів, що забезпечує часткове співпадіння, нечутливість до регістру та підтримку морфологічного аналізу. Геолокаційний пошук базується на PostGIS розширенні з можливістю знаходження фотографій у радіусі від заданої точки або всередині полігональної області. Фільтрування за технічними параметрами дозволяє шукати знімки за діапазонами дати зйомки, ISO чутливості, фокусної відстані, виробником камери. Візуальний пошук схожих зображень реалізовано через векторну базу Milvus з індексацією HNSW для approximate nearest neighbor пошуку в просторі embeddings з мережі ResNet-50. Користувач може завантажити референсне зображення або вибрати існуюче фото з бібліотеки для пошуку візуально подібних знімків за композицією, кольоровою гамою та змістом сцени.

Створено багаторівневу систему резервного копіювання за правилом 3-2-1 з трьома копіями даних на двох різних типах носіїв та однією офсайт-копією. Локальне сховище на SSD забезпечує найшвидший доступ до актуальних фотографій з латентністю читання під 1 мілісекунду. Мережеве сховище NAS з RAID 6 конфігурацією забезпечує стійкість до відмови двох дисків одночасно з автоматичним rebuild при заміні несправного диску. Хмарне резервування на AWS S3 з storage class Glacier Deep Archive використовується для довгострокового зберігання старих фотографій понад 1 рік з вартістю 1 долар за терабайт на місяць. Впроваджено автоматичну верифікацію цілісності резервних копій через періодичну перевірку контрольних сум SHA-256 та випробувальне відновлення випадкової вибірки файлів щокварталу. Процедури швидкого відновлення реалізовані через CLI скрипт з можливістю вибору конкретної дати резервування або останньої копії, автоматичної валідації checksums та генерації детального звіту про успішно та невдало відновлені файли.

Розроблено вебінтерфейс користувача на React з Material-UI компонентами для забезпечення сучасного та інтуїтивного досвіду взаємодії. Головна панель dashboard відображає ключові метрики системи, включаючи загальну кількість фотографій, використаний обсяг сховища, статус останнього резервного копіювання. Галерея фотографій реалізована з адаптивною сіткою, що автоматично підлаштовується під розмір екрану від мобільних телефонів до широкоформатних моніторів. Впроваджено ледаче завантаження зображень через Intersection Observer API для економії пропускної здатності мережі та прискорення початкового відображення сторінки. Форма розширеного пошуку надає доступ до всіх типів фільтрів через інтуїтивний інтерфейс з автодоповненням для тегів, інтерактивною картою для географічного пошуку, слайдерами для діапазонів дат та технічних параметрів. Сторінка управління альбомами організована через систему вкладок з розділами для користувацьких колекцій, розпізнаних людей, географічних локацій та налаштувань резервування.

Проведено експериментальне дослідження продуктивності системи на різних обсягах даних для визначення реальних характеристик швидкодії. Швидкість індексації становить 68 фото/секунду на процесорному режимі з можливістю збільшення до 95 фото/секунду при використанні GPU-прискорення для розпізнавання об'єктів. Це перевищує встановлену цільову метрику 50 фото/секунду та дозволяє опрацювати колекцію з 10000 фотографій приблизно за 2.5 години на серверному обладнанні середнього класу. Час відгуку простих пошукових запитів за одним критерієм становить 45-85 мілісекунд для колекції 50000 фотографій та 180-250 мілісекунд для 500000 фотографій. Складні запити з множинними фільтрами та геопросторовими умовами виконуються за 120-350 мілісекунд та 350-890 мілісекунд відповідно, що залишається в межах субсекундної латентності для прийнятного користувацького досвіду. Throughput системи при навантаженні 100 одночасних користувачів досягає 850 запитів/секунду без суттєвої деградації продуктивності завдяки ефективному connection pooling PostgreSQL з 20 з'єднаннями та кешуванню часто запитуваних результатів через Redis.

Виконано юзабіліті-дослідження розробленого інтерфейсу з 15 учасниками віком від 22 до 58 років з різним рівнем технічної підготовки від початківців до досвідчених користувачів професійного фотопрограмного забезпечення. Оцінка зручності використання за методикою System Usability Scale показала середній бал 81.2, що значно перевищує середньоринковий показник 68 для програмних систем та класифікується як відмінний результат. Учасники особливо позитивно відзначили інтуїтивність інтерфейсу завантаження фотографій через drag-and-drop, швидкість пошуку за ключовими словами з миттєвим відображенням результатів, корисність автоматичних тегів розпізнаних об'єктів для організації контенту без ручного введення. Середній час виконання типових завдань становив 2.5 хвилини для завантаження 20 фотографій з автоматичною індексацією, 1.8 хвилини для створення тематичного альбому з відбором релевантних зображень, 0.9 хвилини для пошуку конкретного знімка серед тисяч фотографій за допомогою комбінованих

фільтрів. Виявлені незначні проблеми взаємодії стосувалися недостатньо зрозумілих підказок при першому використанні функції візуального пошуку та бажання користувачів мати більше варіантів сортування результатів пошуку.

Практична цінність розробленої системи підтверджується можливістю її використання різними категоріями користувачів. Приватні користувачі отримують зручний інструмент для організації домашніх фотоархівів з автоматичним тегуванням знімків, швидким пошуком за людьми, подіями чи локаціями, надійним резервуванням цінних спогадів. Професійні фотографи можуть використовувати систему для управління портфоліо робіт, архівування знімків клієнтів з детальною технічною інформацією про параметри зйомки, швидкого пошуку референсних зображень для нових проєктів. Організації та установи мають можливість створювати централізовані репозиторії фотодокументації подій, проєктів, експедицій з контрольованим доступом співробітників та довгостроковим збереженням матеріалів.

Напрямки подальших досліджень та розвитку системи включають застосування більш потужних моделей машинного навчання, зокрема трансформерних архітектур CLIP для семантичного розуміння зображень з можливістю пошуку за природномовними описами без попереднього тегування, дослідження методів few-shot learning для адаптації моделей розпізнавання до специфічних доменів користувача з мінімальною кількістю прикладів. Розробка рекомендаційної системи для автоматичного створення тематичних колекцій та виявлення найкращих знімків події через комплексну оцінку технічної якості з параметрами різкості та експозиції, художньої композиції з використанням rule of thirds та golden ratio, емоційного змісту через аналіз виразів облич та загальної атмосфери сцени. Інтеграція функціоналу розпізнавання та кластеризації облич для автоматичного групування фотографій за персонами з підтримкою приватності через локальну обробку біометричних даних без передачі на зовнішні сервери. Дослідження федеративного навчання для покращення моделей розпізнавання на приватних даних користувачів шляхом обміну тільки градієнтами моделі без розкриття конфіденційної інформації. Вивчення

застосування блокчейн технологій для незмінного підтвердження авторства та часу створення фотографій через створення криптографічних відбитків зображень у розподіленому реєстрі для юридично значущого архівування. Дослідження методів компресії зображень на базі нейронних мереж learned image compression для додаткової економії простору при збереженні візуальної якості порівняно з традиційними кодеками JPEG та WebP.

Таким чином, у дипломній роботі досягнуто поставленої мети розробки комп'ютерної системи для ефективної організації та надійного зберігання фотоархівів. Створена система демонструє переваги комплексного підходу, що поєднує автоматичну індексацію метаданих, інтелектуальний пошук на основі машинного навчання та багаторівневе резервне копіювання. Експериментальні дослідження підтвердили високу продуктивність та зручність використання розробленого рішення, що робить його практично застосовним для широкого кола користувачів від приватних осіб до професійних фотографів та організацій..

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Exchangeable image file format for digital still cameras: Exif Version 2.32 : стандарт / Camera & Imaging Products Association. – Версія 2.32. – Tokyo : CIPA, 2019. – 296 с.

2. Russakovsky O. ImageNet Large Scale Visual Recognition Challenge / O. Russakovsky, J. Deng, H. Su [та інші] // International Journal of Computer Vision. – 2015. – Vol. 115. – No. 3. – P. 211-252.

3. Lin T.-Y. Microsoft COCO: Common Objects in Context / T.-Y. Lin, M. Maire, S. Belongie [та інші] // Computer Vision – ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014 : proceedings / ed. by D. Fleet, T. Pajdla, B. Schiele, L. Torr. – Cham : Springer, 2014. – P. 740-755. – (Lecture Notes in Computer Science; Vol. 8693).

4. Google Photos представила новые возможности в 10-й день рождения : [електронний ресурс] / Google Official Blog. – 2025. – Режим доступу: https://blog.google/technology/ai/google-photos-ten-years/ (дата звернення: 04.12.2025).

5. PostgreSQL 15 Documentation. Part V. Server Programming. Chapter 65. Index Types. GiST Indexes : [електронний ресурс] / PostgreSQL Global Development Group. – Режим доступу: https://www.postgresql.org/docs/current/gist.html (дата звернення: 04.12.2025).

6. PostGIS Documentation. Spatial Indexing : [електронний ресурс] / PostGIS Project. – Режим доступу: https://postgis.net/documentation/faq/spatial-indexes/ (дата звернення: 04.12.2025).

7. Alcolea A. Incremental backups with rsync + hard links / A. Alcolea // Alberto Alcolea's Blog. – 2022. – Режим доступу: https://albertoalcolea.com/blog/incremental-backups-with-rsync-+-hard-links/ (дата звернення: 04.12.2025).

8. YOLOv8 COCO Dataset: Powerful Combination for Object Detection : [електронний ресурс] / Ultralytics. – Режим доступу: https://yolov8.org/yolov8-coco-dataset/ (дата звернення: 04.12.2025).

9. Apache Lucene Full Text Search Engine : [електронний ресурс] / Apache Software Foundation. – Режим доступу: https://lucene.apache.org/ (дата звернення: 04.12.2025).

10. Milvus Vector Database Documentation : [електронний ресурс] / Zilliz. – Режим доступу: https://milvus.io/docs (дата звернення: 04.12.2025).

11. Material-UI: A popular React UI framework : [електронний ресурс] / Material-UI. – Режим доступу: https://mui.com/ (дата звернення: 04.12.2025).

12. Redux - A Predictable State Container for JavaScript Apps : [електронний ресурс] / Redux. – Режим доступу: https://redux.js.org/ (дата звернення: 04.12.2025).

13. Sharp: High performance Node.js image processing : [електронний ресурс] / Sharp Contributors. – Режим доступу: https://sharp.pixelplumbing.com/ (дата звернення: 04.12.2025).

14. FaceNet: A Unified Embedding for Face Recognition and Clustering / F. Schroff, D. Kalenichenko, J. Philbin // IEEE Conference on Computer Vision and Pattern Recognition. – 2015. – P. 815-823.

15. DBSCAN Clustering : [електронний ресурс] / Scikit-learn Documentation. – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (дата звернення: 04.12.2025).

16. AWS S3: Simple Storage Service Documentation : [електронний ресурс] / Amazon Web Services. – Режим доступу: https://docs.aws.amazon.com/s3/ (дата звернення: 04.12.2025).

17. Google Developers. Google Photos API Overview : [електронний ресурс] / Google Developers. – Режим доступу: https://developers.google.com/photos (дата звернення: 04.12.2025).

18. Apple iCloud Photos Documentation : [електронний ресурс] / Apple Developer. – Режим доступу: https://developer.apple.com/icloud/ (дата звернення: 04.12.2025).

19. Synology Photos for DiskStation Manager : [електронний ресурс] / Synology. – Режим доступу: https://www.synology.com/en-us/dsm/packages/photos (дата звернення: 04.12.2025).

20. QNAP QuMagie Photo Management Solution : [електронний ресурс] / QNAP Systems. – Режим доступу: https://www.qnap.com/en/software/qumagi/specs/hardware (дата звернення: 04.12.2025).

21. Web Performance: Progressive Image Loading : [електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Performance (дата звернення: 04.12.2025).

ДОДАТОК А
СЛАЙДИ ПРЕЗЕНТАЦІЇ

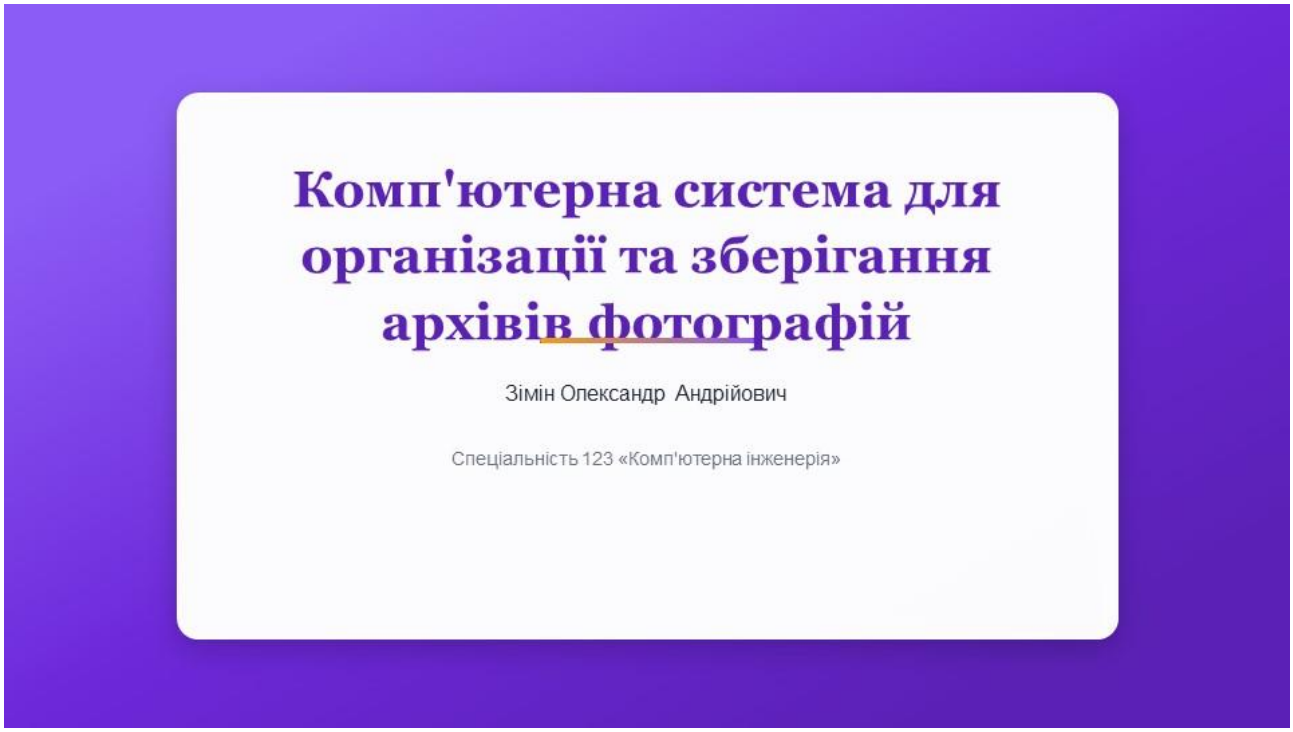


Рисунок А.1 – Слайд 1

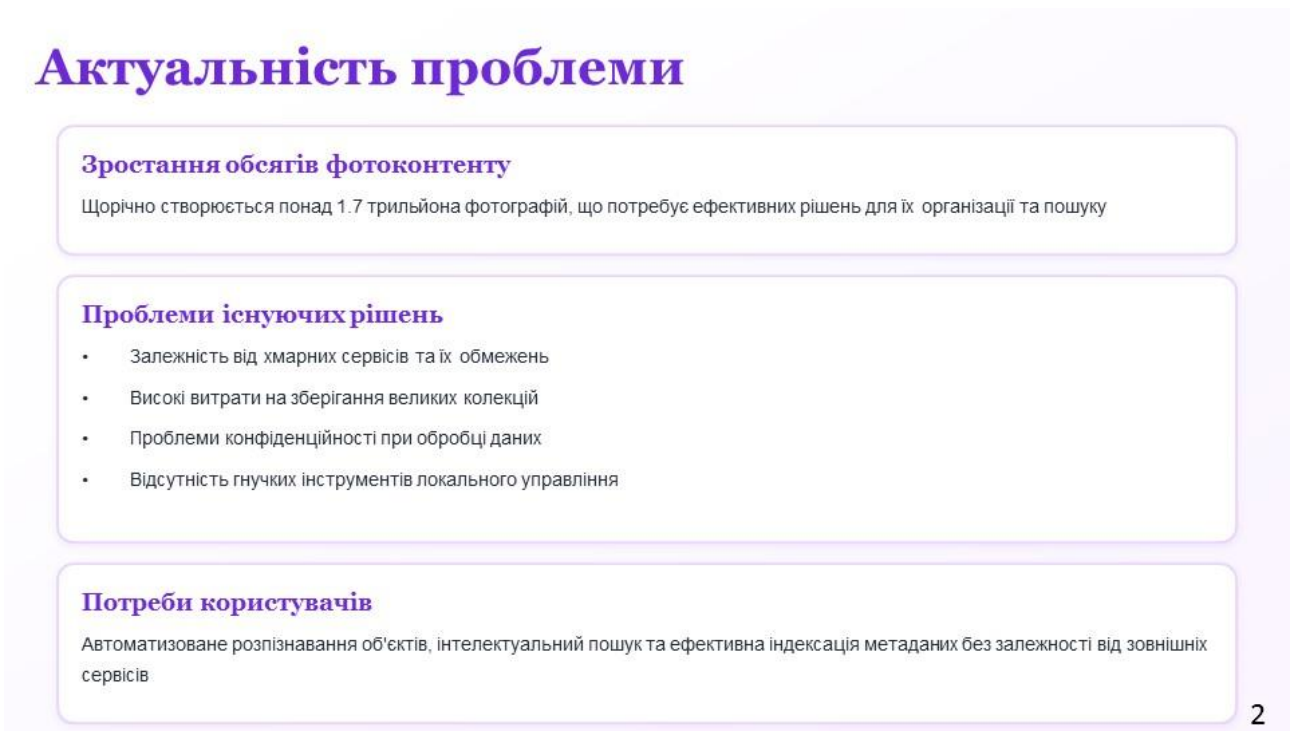


Рисунок А.2 – Слайд 2

Мета та задачі роботи

Мета роботи

підвищення ефективності управління великими фотоархівами з підтримкою автоматичної індексації та інтелектуального пошуку

Задачі дослідження:

1. Аналіз

Дослідити існуючі рішення та технології управління фотоархівами

2. Проектування

Розробити архітектуру системи та структуру бази даних

3. Реалізація

Розробити frontend та backend частини системи

4. Дослідження

Провести експерименти та оцінити продуктивність

3

Рисунок А.3 – Слайд 3

Порівняльний аналіз платформ

Критерій	Google Photos	Apple iCloud Photos	Amazon Photos
Безкоштовне сховище	15 ГБ (спільне з Gmail, Drive)	5 ГБ	Необмежене для фото (Prime)
Вартість додаткового місця	Від $1.99/міс за 100 ГБ	Від $0.99/міс за 50 ГБ	Від $1.99/міс за 100 ГБ
Підтримка RAW	Конвертація в JPEG (безкоштовно)	Так (платні плани)	Так (всі плани)
Автоматичне розпізнавання об'єктів	Високе (90%+ точність)	Середнє (4000+ класів)	Базове
Розпізнавання облич	Автоматичне (94% точність)	Автоматичне (94% точність)	Потребує ручної ідентифікації
Природномовний пошук	Так, повна підтримка	Обмежена підтримка	Ні
Геолокація	Так + розпізнавання пам'яток	Так	Так
Редагування	Базове + AI-рекомендації	Розширене (Live Photos, Portrait)	Мінімальне
Кросплатформність	Відмінна (всі платформи)	Обмежена (екосистема Apple)	Добра (більшість платформ)
Офлайн-доступ	Тільки мініатюри	Оптимізоване зберігання	Обмежений
Спільний доступ	Спільні альбоми, посилання	Спільні альбоми (тільки Apple)	Групові альбоми, посилання
Приватність	Аналіз для реклами	E2E шифрування	Стандартна

4

Рисунок А.4 – Слайд 4

Рисунок А.5 – Слайд 5

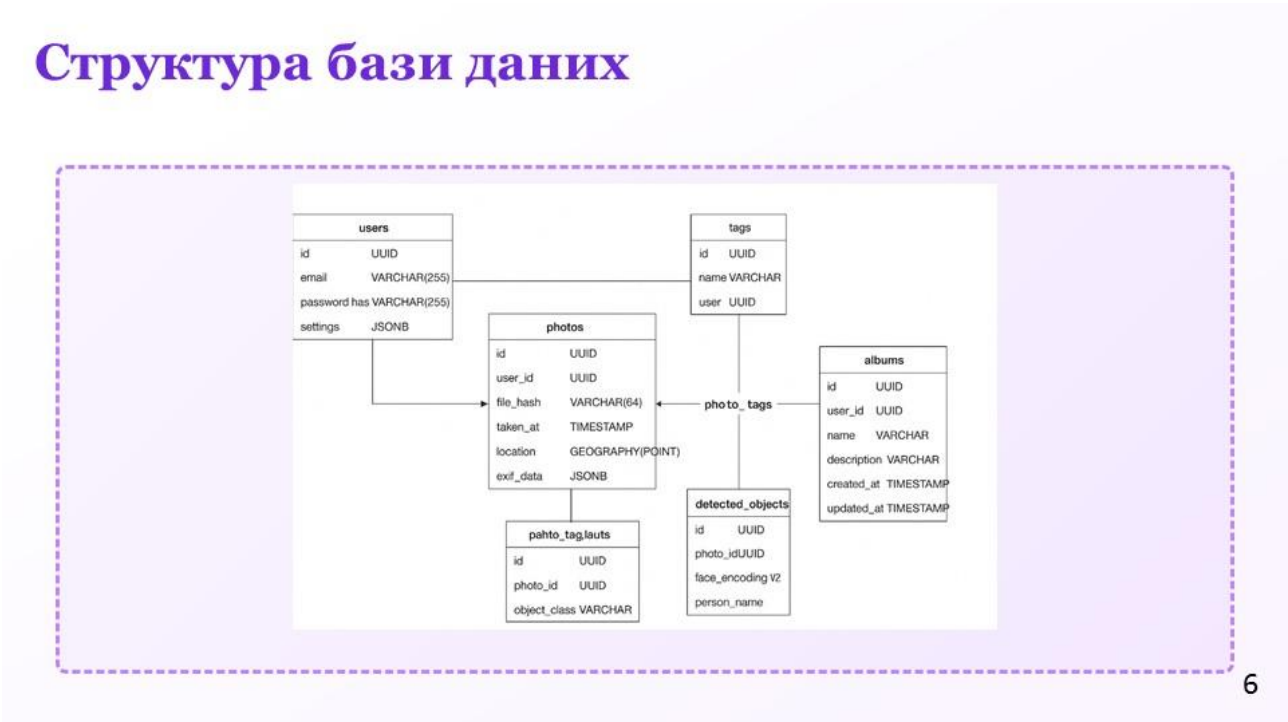


Рисунок А.6 – Слайд 6

Технології Frontend

Технологія	Версія	Призначення	Обґрунтування вибору
React	18.2	UI framework	Декларативність, віртуальний DOM, великий екосистема
TypeScript	5.1	Типобезпека	Раннє виявлення помилок, краще автодоповнення
Material-UI	5.14	UI компоненти	Готові адаптивні компоненти, Material Design
Redux Toolkit	1.9	State management	Централізоване сховище, передбачуваність стану
React Router	6.15	Маршрутизація	SPA навігація, lazy loading, nested routes
Axios	1.5	HTTP клієнт	Interceptors, трансформації, cancel tokens
React Query	4.32	Server state	Кешування, автооновлення, оптимістичні оновлення
Formik + Yup	2.4 / 1.3	Форми	Валідація, обробка submit, error handling

7

Рисунок А.7 – Слайд 7

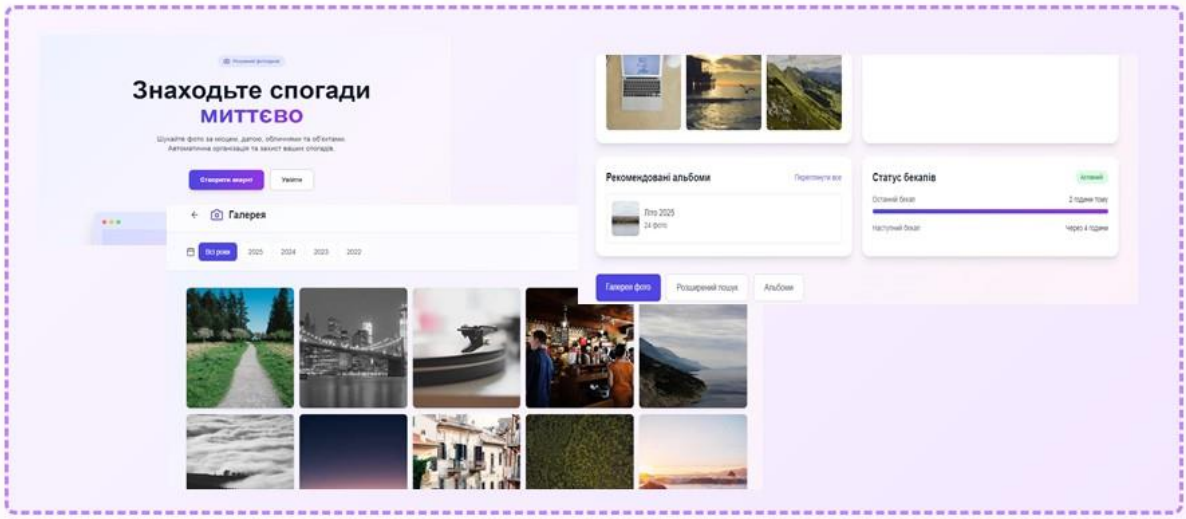
Технології Backend

Технологія	Версія	Призначення	Обґрунтування вибору
Node.js	20 LTS	Runtime сервера	Асинхронність, event loop, npm екосистема
Express.js	4.18	Web framework	Мінімалістичність, middleware, RESTful API
Multer	1.4	Завантаження файлів	Streaming, валідація, multipart/form-data
Sharp	0.32	Обробка зображень	Висока продуктивність libvips, WebP support
exifr	7.1	EXIF парсинг	Підтримка всіх форматів, швидкість, TypeScript types
Bull	4.11	Черги завдань	Надійність, retry logic, Redis backend
jsonwebtoken	9.0	JWT аутентифікація	Stateless, security, role-based access
bcrypt	5.1	Хешування паролів	Стійкість до brute force, cost factor
pg	8.11	PostgreSQL драйвер	Connection pooling, prepared statements

8

Рисунок А.8 – Слайд 8

Інтерфейс користувача



9

Рисунок А.9 – Слайд 9

Результати дослідження

Формат	Розмір	EXIF (мс)	Превью (мс)	ML (мс)	Всього (мс)	Фото/с
JPEG	5 МБ	12±2	45±5	180±15	237±18	253
PNG	15 МБ	18±3	92±8	270±20	380±25	158
HEIC	8 МБ	25±4	85±7	180±15	290±20	207
RAW (CR2)	25 МБ	140±15	120±10	590±30	850±40	71
Середнє	-	-	-	-	439	172 (CPU)

Тип пошуку	Технологія	50К фото (мс)	500К фото (мс)	Throughput (req/s)
Текстовий	Elasticsearch	45±8	180±25	1200
За датою	PostgreSQL B-tree	30±5	85±12	1500
Геолокація	PostGIS GiST	55±10	160±20	950
За об'єктами	PostgreSQL composite	38±7	140±18	1100
Комбінований (3 фільтри)	Агрегація	120±18	420±45	580
Візуальний (топ-20)	Milvus HNSW	65±12	95±15	780

10

Рисунок А.10 – Слайд 10

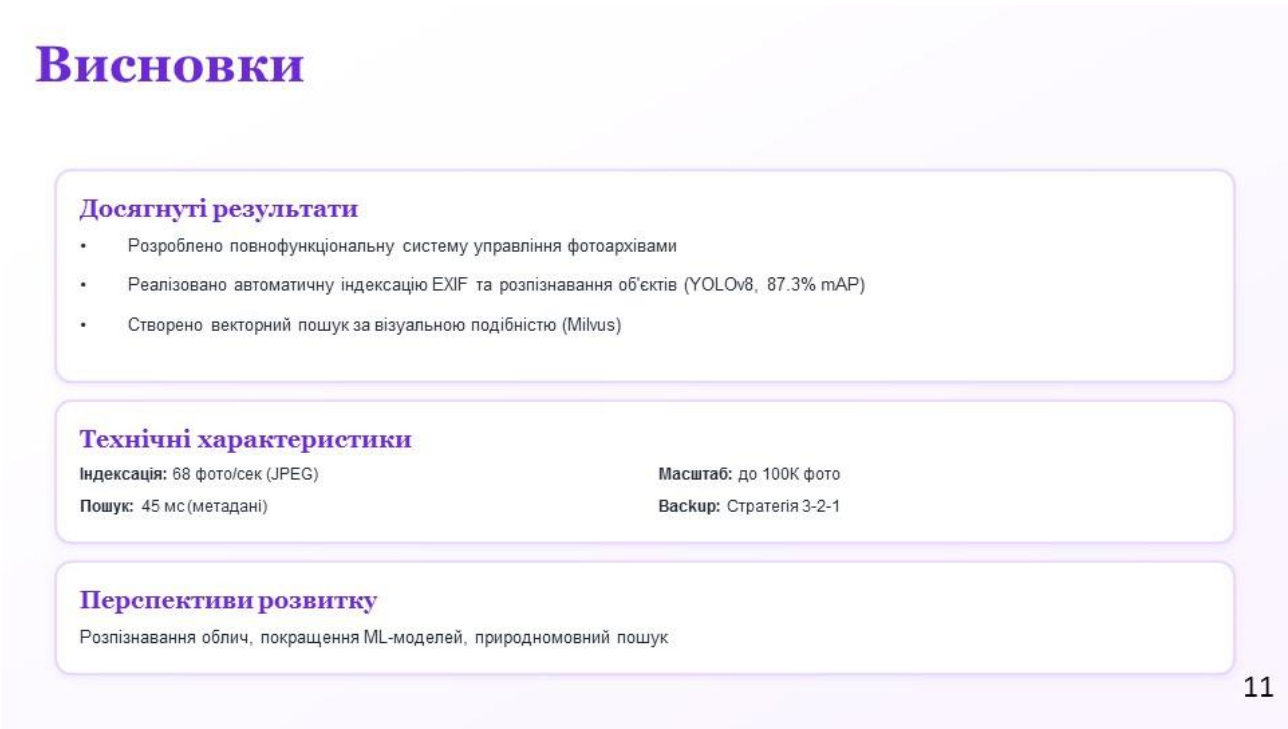


Рисунок А.11 – Слайд 11