Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Веб-орієнтована система гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Front-end частина
(тема)

Виконав:

студент 4 курсу, групи ПЗПІ-20-8

Долготер М. С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. Терещенко Г.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Долготеру Михайлу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-орієнтована система гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Front-end частина
затверджена наказом університету від 20.05. 2024. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 14.06. 2024 р.
3. Вихідні дані до роботи *Front-end частина веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Використовуючи сучасні веб-технології, такі як React.js та HTML/CSS, система надає зручний інтерфейс для ефективного управління зображеннями.*
4. Перелік питань, що потрібно опрацювати у роботі *Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, висновки, додатки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2024	виконано
2	Створення специфікації програмного забезпечення	22.05.2024	виконано
3	Проєктування програмного забезпечення	24.05.2024	виконано
4	Розробка програмного забезпечення	28.05.2024	виконано
5	Тестування програмного забезпечення	30.05.2024	виконано
6	Оформлення пояснювальної записки	05.06.2024	виконано
7	Підготовка доповіді та презентації	06.06.2024	виконано
8	Перевірка на нормоконтроль	09.06.2024	виконано
9	Попередній захист	06.06.2024	виконано
10	Здача роботи у електронний архів	10.06.2024	виконано
11	Допуск до захисту у зав. кафедри	11.06.2024	виконано

Дата видачі завдання 16 травня 2024 р.

Студент _____ Долготер М.С.

(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Терещенко Г.Ю.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 105 стор., 25 рис., 1 табл., 16 джерел.

АНАЛІЗ, ВЕБ-ОРІЄНТОВАНА СИСТЕМА, ВЕЛИКІ ДАНІ, ГІБРИДНИЙ МЕХАНІЗМ УПРАВЛІННЯ, ЗОБРАЖЕННЯ, ОБРОБКА, CSS, FRONT-END, HTML, REACT.

Об’єкт розробки – front-end частина веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, яка зосереджена на користувацьких аспектах системи, забезпечуючи інтуїтивно зрозумілий та візуально привабливий інтерфейс для взаємодії з користувачами.

Мета розробки – створити інтуїтивно зрозумілий та ефективний користувацький інтерфейс для веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, який спрощує процеси керування зображеннями, сприяє ефективній обробці та аналізу зображень та задовольняє потреби користувачів, які взаємодіють з великими обсягами зображень в умовах великих даних.

Метод рішення – React для створення компонентів інтерфейсу користувача, HTML для структурування вмісту веб-сторінок, CSS для стилізації та середовище розробки VS Code.

У результаті розробки створено front-end частину веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Front-end частина пропонує зручний інтерфейс з широким спектром функціональних можливостей. Користувачі можуть легко шукати файли за різними критеріями та метриками для зберігання, аналізу та класифікації зображень. Вони можуть додавати нові файли до системи та переглядати їх у різних форматах. Система надає детальну інформацію про обсяг завантаження, а користувачі можуть відстежувати час і швидкість завантаження та вивантаження файлів. Крім того, користувачі можуть виконувати завдання управління файлами, такі як копіювання, вирізання, вставка, перейменування та

видалення файлів і папок, що забезпечує комплексне рішення для організації та управління ресурсами зображень. Система також відображає інформацію про використання пам’яті в режимі реального часу, щоб користувачі знали про використання ресурсів і ємність сховища.

ANALYSIS, WEB-ORIENTED SYSTEM, BIG DATA, HYBRID CONTROL MECHANISM, IMAGES, PROCESSING, CSS, FRONT-END, HTML, REACT.

The object of development is the front-end part of a web-oriented system of a hybrid image management mechanism for efficient processing and analysis in big data, which focuses on the user aspects of the system, providing an intuitive and visually appealing interface for user interaction.

The development goal is to create an intuitive and efficient user interface for a web-based system of a hybrid image management engine for efficient image processing and analysis in a big data environment that simplifies image management processes, facilitates efficient image processing and analysis, and meets the needs of users who interact with large volumes of images in a big data environment.

The solution method is React for creating user interface components, HTML for structuring the content of web pages, CSS for styling, and the VS Code development environment.

As a result of the development, the front-end part of a web-oriented system of a hybrid image management mechanism for efficient processing and analysis in big data conditions was created. The front-end part offers a user-friendly interface with a wide range of functionalities. Users can easily search for files by various criteria and metrics to store, analyze, and classify images. They can add new files to the system and view them in various formats. The system provides detailed information about the volume of uploads, and users can track the time and speed of uploading and downloading files. Additionally, users can perform file management tasks such as copying, cutting, pasting, renaming, and deleting files and folders, providing a comprehensive solution for organizing and managing image resources. The system also displays real-time storage usage information so that users are aware of resource utilization and storage capacity.

Я, Долготер Михайло Сергійович, студент гр. ПЗПІ-20-8, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Веб-орієнтована система гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Front-end частина», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У наш час поширення цифрових зображень змінило ландшафт управління даними, вимагаючи розробки систем, здатних ефективно обробляти величезні обсяги візуальних даних. Оскільки цифрова сфера продовжує розвиватися, потреба в надійних механізмах управління зображеннями стає дедалі виразнішою. Запропоноване рішення являє собою front-end частину веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, пристосованої для управління та взаємодії з великими наборами даних зображень. Використовуючи сучасні веб-технології та принципи дизайну, front-end частина системи має на меті надати користувачеві зручний інтерфейс, що полегшує навігацію та взаємодію з файлами зображень.

Розробка інтерфейсу користувача відіграє ключову роль у формуванні користувацького досвіду в цифрових системах. Оскільки обсяг цифрових зображень продовжує зростати, зростає попит на інтерфейсні рішення, які можуть забезпечити інтуїтивно зрозумілі інтерфейси для навігації та взаємодії зі сховищами зображень. Традиційним системам управління зображеннями часто бракує гнучкості та орієнтованого на користувача дизайну, необхідного для задоволення різноманітних потреб користувачів, які взаємодіють з великими наборами даних зображень [1]. Тому метою цієї роботи є розробка front-end частини, яка зосереджуватиметься на зручності використання, ефективності та естетичній привабливості, щоб задовольнити потреби користувачів.

Основною метою цієї роботи є розробка та впровадження front-end частини, яка дозволить користувачам ефективно керувати та взаємодіяти з великомасштабними наборами даних зображень. Зокрема, завдання цієї роботи полягають у наступному:

- розробити інтуїтивно зрозумілий та естетично привабливий користувацький інтерфейс, який надає пріоритет простоті навігації та доступності, тим самим підвищуючи залученість та задоволеність користувачів;

- реалізувати основні функції, такі як автентифікація та реєстрація користувачів, пошук файлів за певними критеріями та метриками для пошуку, зберігання, аналізу та класифікації зображень, додавання, перегляд у різних форматах, відображення метаданих, завантаження з детальною інформацією про об’єм завантаження, моніторинг часу та швидкості завантаження/вивантаження, а також основні операції редагування;
- оптимізувати інтерфейс системи для швидкої реакції та продуктивності на різних пристроях і розмірах екранів, забезпечуючи послідовний та безперебійний користувацький досвід;
- інтегрувати механізми моніторингу використання системних ресурсів і надати користувачам у режимі реального часу інформації про використання пам’яті та ємності сховища, що дозволяє приймати обґрунтовані рішення щодо розподілу та оптимізації ресурсів.

Обсяг цієї кваліфікаційної роботи обмежується розробкою інтерфейсу системи управління зображеннями, зосереджуючись виключно на проектуванні та реалізації користувацького інтерфейсу та компонентів взаємодії. Передбачувана система знайде застосування в різних сферах, де ефективне управління та взаємодія з даними зображень є незамінними. Приклади включають управління особистими фотографіями, професійні фотографічні портфоліо, платформи електронної комерції та системи управління контентом, що вимагають інтерфейсів, орієнтованих на зображення. Надання надійного та зручного інтерфейсного рішення спрямоване на підвищення продуктивності у різних застосунках, що покладаються на ефективне управління зображеннями.

Отже, запропонована front-end частина система має на меті полегшити безперешкодну взаємодію з великими наборами даних зображень. Завдяки інтуїтивно зрозумілим інтерфейсам та інтерактивним елементам користувачі отримають інструменти, які необхідні для навігації, керування та взаємодії з файлами зображень відповідно до їхніх різноманітних потреб та вподобань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Індустрія управління зображеннями охоплює широкий спектр технологій, послуг і додатків, спрямованих на ефективну організацію, обробку та аналіз візуальних даних. В останні роки поширення цифрових зображень у різних секторах сприяло зростанню цієї галузі, каталізуючи розробку інноваційних рішень для задоволення зростаючих потреб користувачів та організацій.

Поява цифрової фотографії, платформ соціальних мереж та інструментів для створення мультимедійного контенту призвела до експоненціального зростання обсягу візуального контенту, що генерується та споживається в усьому світі. Таке поширення візуального контенту охоплює різні сектори, включаючи соціальні мережі, електронну комерцію, розваги, охорону здоров’я, спостереження та наукові дослідження.

Швидке зростання цифрових зображень породило супутнє зростання обсягу, швидкості та різноманітності даних зображень, що створює значні проблеми з точки зору зберігання, обробки та аналізу. Традиційні системи управління зображеннями часто не справляються з масштабом і складністю великих даних, що призводить до зниження продуктивності, проблем з масштабуванням і неефективного використання ресурсів [2].

У світлі викликів, пов’язаних з великими даними, зростає попит на ефективні рішення для управління зображеннями, здатні працювати з великими наборами даних зображень, забезпечуючи при цьому зручну взаємодію з користувачем і надійні аналітичні можливості. Організації з різних секторів шукають рішення, які можуть оптимізувати робочі процеси обробки зображень, полегшити прийняття рішень на основі даних і підвищити продуктивність.

Індустрія обробки зображень є свідком конвергенції технологій з різних галузей, включаючи штучний інтелект, машинне навчання, комп’ютерний зір, хмарні обчислення та розподілені системи. Ці технології використовуються для розробки передових алгоритмів обробки зображень, інтелектуальних систем

тегування та категоризації, а також хмарних платформ для зберігання та пошуку зображень.

Для вирішення проблем, пов’язаних з великими обсягами даних, зростає тенденція до впровадження гібридних підходів в управлінні зображеннями. Ці підходи поєднують переваги централізованих і розподілених архітектур, використовуючи масштабованість розподілених систем і простоту централізованих рішень для зберігання даних. Гібридні механізми управління зображеннями спрямовані на оптимізацію продуктивності, масштабованості та використання ресурсів, забезпечуючи при цьому безперебійну взаємодію з користувачами та доступність даних.

Оскільки системи керування зображеннями стають все більш невід’ємною частиною повсякденних робочих процесів, все більше уваги приділяється дизайну користувацького інтерфейсу. Сучасні інтерфейси управління зображеннями надають перевагу інтуїтивно зрозумілій навігації, адаптивному дизайну, робочим процесам, що налаштовуються, та інтерактивним функціям для підвищення рівня залученості та задоволеності користувачів. Юзабіліті-тестування, механізми зворотного зв’язку з користувачами та методології ітеративного дизайну відіграють вирішальну роль у вдосконаленні та оптимізації користувацького досвіду.

З поширенням даних про зображення зростає усвідомлення проблем безпеки та конфіденційності, пов’язаних з системами управління зображеннями. Організації повинні впроваджувати надійні заходи безпеки, щоб захистити конфіденційні дані зображень від несанкціонованого доступу, витоку даних і зловмисних атак. Дотримання правил захисту даних, технології шифрування, контроль доступу та надійні механізми автентифікації мають першорядне значення для забезпечення конфіденційності, цілісності та доступності даних зображень.

Застосування систем керування зображеннями охоплює різні галузі та сфери. У сфері охорони здоров’я ці системи полегшують управління та аналіз медичних зображень для діагностики, планування лікування та наукових досліджень. В електронній комерції платформи управління зображеннями дозволяють

роздрібним торговцям демонструвати товари за допомогою високоякісних зображень, покращуючи досвід покупок для клієнтів. У сфері спостереження та безпеки ці системи використовуються для моніторингу в реальному часі, виявлення загроз і криміналістичного аналізу. Крім того, в наукових дослідженнях рішення для обробки зображень підтримують аналіз даних, візуалізацію та співпрацю між дослідниками.

У динамічному ландшафті управління зображеннями кілька аналогів виділяються як видатні рішення, кожне з яких пропонує унікальні можливості та функції, пристосовані до різноманітних потреб користувачів. Від хмарних сервісів зберігання до платформ для обміну фотографіями, керованих спільнотами, ці аналоги відіграють ключову роль у полегшенні ефективної обробки, організації та аналізу зображень. Нижче наведено аналіз чотирьох аналогів у сфері управління зображеннями.

Google Photos - це широко використовуваний хмарний сервіс для зберігання фотографій та відео, розроблений компанією Google. Запущений у 2015 році, він пропонує користувачам широкий набір функцій для зберігання, впорядкування, редагування та обміну цифровими медіа (див. рис. 1.1).

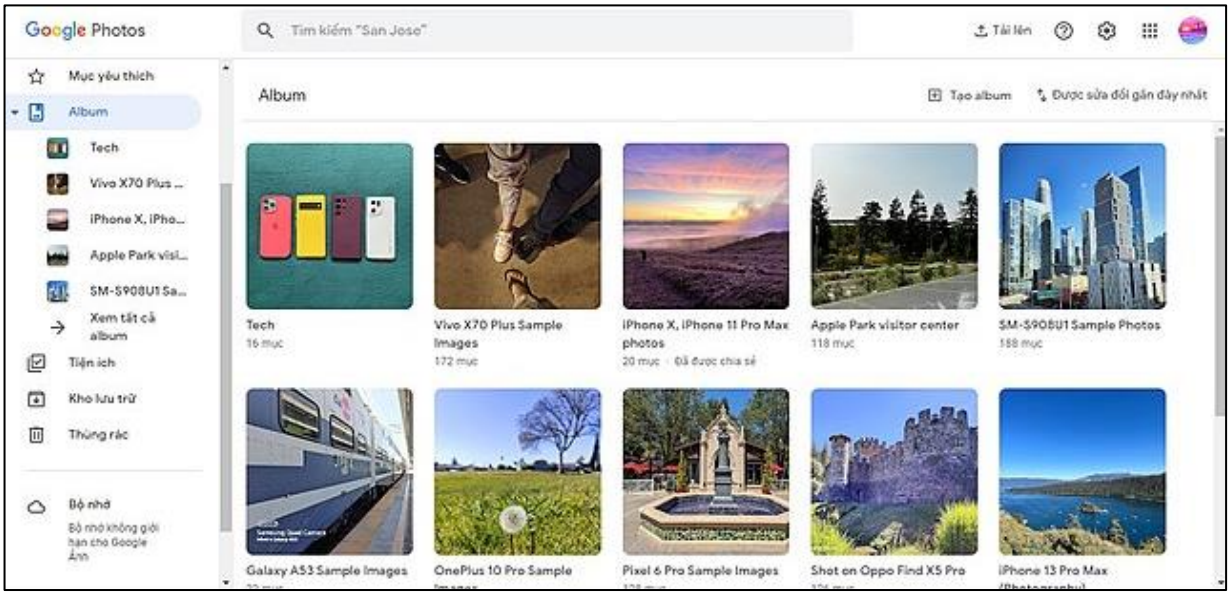


Рисунок 1.1 – Google Photos (рисунок створено самостійно)

Google Photos надає користувачам низку функцій, спрямованих на спрощення управління та організації їхніх фото- та відеоколекцій:

- Google Photos автоматично створює резервні копії фотографій та відео з пристроїв користувачів у хмарі, гарантуючи, що їхні спогади надійно зберігаються та доступні з будь-якого підключеного пристрою;
- сервіс впорядковує фотографії та відео на основі різних критеріїв, таких як дата, місцезнаходження, люди та об’єкти, виявлені на зображеннях за допомогою алгоритмів на основі штучного інтелекту;
- користувачі можуть легко шукати певні фотографії або відео за ключовими словами, людьми, місцями або об’єктами завдяки передовій технології розпізнавання зображень від Google;
- Google Photos пропонує набір інструментів для редагування, які дозволяють користувачам покращувати свої фотографії, застосовувати фільтри, налаштовувати кольори, обрізати та багато іншого;
- користувачі можуть ділитися своїми фотографіями та відео з іншими за допомогою прямих посилань, електронної пошти, соціальних мереж або спільних альбомів.

Google Photos слугує багатьом користувачам для різних цілей:

- це цифрове сховище, де користувачі можуть зберігати свої спогади, а також мати доступ до своїх фотографій та відео з будь-якого місця;
- Google Photos допомагає користувачам ефективно організовувати та керувати своїми великими колекціями цифрових медіа, полегшуючи пошук і переживання особливих моментів;
- користувачі можуть легко співпрацювати з іншими, обмінюючись альбомами або окремими фотографіями/відео, сприяючи безперешкодному обміну та співпраці між друзями, родиною та колегами.

Переваги Google Photos:

- Google Photos пропонує безкоштовне необмежене сховище для фотографій та відео (з деякими обмеженнями щодо якості відео), що робить його привабливим варіантом для користувачів, які прагнуть заощадити місце на своїх пристроях;
- легко інтегрується з іншими сервісами Google, такими як Gmail, Google

Drive і Google Assistant, що підвищує продуктивність та зручність для користувачів;

- розширені функції пошуку та впорядкування на основі алгоритмів штучного інтелекту дозволяють користувачам легко знаходити та впорядковувати свої фотографії та відео без зайвих зусиль;
- Google Photos доступний на різних пристроях і платформах, включаючи смартфони, планшети, настільні комп'ютери та веб-браузери, що забезпечує повсюдний доступ до медіаколекцій користувачів.

Недоліки Google Photos:

- Google Photos викликає занепокоєння щодо конфіденційності через великий обсяг збору та аналізу даних, пов'язаних з його функціями на основі штучного інтелекту, такими як розпізнавання облич і виявлення об'єктів;
- хоча Google Photos пропонує безкоштовне необмежене сховище, він може стискати фотографії та відео для зменшення розміру файлів, що потенційно може погіршити якість зображень, особливо для професійних фотографів та ентузіастів;
- користувачі потребують стабільного інтернет-з'єднання для доступу до своїх фотографій та відео, що зберігаються в хмарі, що може створювати проблеми в автономних сценаріях або в місцях з обмеженим підключенням;
- хоча Google Photos надає базові інструменти для редагування, йому бракує розширених функцій та можливостей, які пропонує спеціалізоване програмне забезпечення для редагування фотографій.

Google Photos слугує еталоном для систем керування зображеннями, демонструючи потенціал хмарних рішень з розширеними функціями, керованими штучним інтелектом. Акцент на безперебійному користувацькому досвіді, потужних можливостях пошуку та організації, а також повсюдному доступі є прикладом найкращих практик у сучасному управлінні зображеннями. Однак його вплив на конфіденційність та обмеження можливостей редагування підкреслюють

важливість адаптації систем управління зображеннями до конкретних потреб і вподобань користувачів, надаючи при цьому пріоритет безпеці даних і конфіденційності користувачів.

Microsoft OneDrive – це хмарне сховище, розроблене корпорацією Microsoft, що пропонує користувачам платформу для зберігання, синхронізації та спільного використання файлів і папок на різних пристроях (див. рис. 1.2). Спочатку запущений у 2007 році як Windows Live Folders, він був пізніше перейменований на OneDrive у 2014 році.

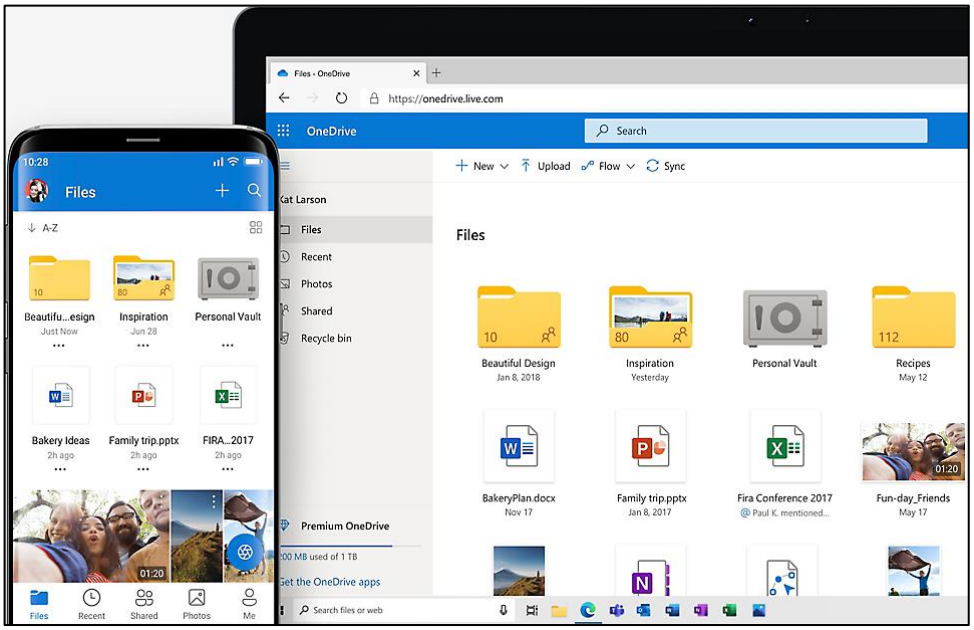


Рисунок 1.2 – Microsoft OneDrive (рисунок створено самостійно)

Microsoft OneDrive надає користувачам комплексний набір функцій для зберігання та керування файлами та папками в хмарі. Ключові функціональні можливості включають в себе:

– OneDrive дозволяє користувачам зберігати документи, фотографії, відео та інші файли в хмарі, забезпечуючи централізоване сховище, доступне з будь-якого пристрою, підключеного до Інтернету;
– пропонує автоматичну синхронізацію файлів і папок між пристроями, гарантуючи, що зміни, внесені на одному пристрої, відображаються на всіх інших пристроях, пов’язаних з тим самим обліковим записом OneDrive;

- OneDrive полегшує співпрацю між користувачами, дозволяючи в режимі реального часу спільно працювати над документами Office, спільними папками та посиланнями для спільного доступу до файлів із дозволами, що налаштовуються;
- легко інтегрується з пакетом Microsoft Office, дозволяючи користувачам створювати, редагувати та спільно працювати над документами Word, Excel, PowerPoint та іншими документами Office безпосередньо в OneDrive.

Microsoft OneDrive слугує багатьом користувачам для різних цілей:

- це рішення для резервного копіювання, що дає змогу користувачам захищати свої файли та отримувати до них доступ з будь-якого місця, де є підключення до Інтернету;
- OneDrive сприяє співпраці та підвищенню продуктивності, забезпечуючи безперешкодний обмін документами, презентаціями та проектами між окремими користувачами й командами та спільну роботу над ними;
- задовольняє потреби користувачів різних платформ, зокрема Windows, macOS, iOS, Android і веб-браузерів, забезпечуючи сумісність та доступність на різних пристроях і в різних операційних системах.

Переваги Microsoft OneDrive:

- OneDrive тісно інтегровано з операційною системою Windows, що забезпечує вбудовану підтримку та безперешкодне керування файлами за допомогою провідника файлів Windows;
- пропонує глибоку інтеграцію з пакетом Office 365, підвищуючи продуктивність завдяки таким функціям, як співавторство в режимі реального часу, історія версій та розширені можливості редагування;
- OneDrive пропонує щедрі плани зберігання, зокрема безкоштовну квоту за замовчуванням і додаткові можливості зберігання для передплатників Office 365, що робить його привабливим варіантом як для приватних осіб, так і для підприємств;

- корпорація Microsoft надає пріоритет безпеці та відповідності галузевим стандартам, пропонуючи такі функції, як шифрування, багатофакторна автентифікація, запобігання втраті даних та сертифікати відповідності, щоб захистити дані користувачів і забезпечити відповідність нормативним вимогам.

Недоліки Microsoft OneDrive:

- хоча OneDrive за замовчуванням пропонує безкоштовну квоту на зберігання даних, її може бути недостатньо для користувачів із великими потребами у сховищі, що призведе до необхідності придбання додаткових планів на зберігання;
- користувачі можуть зіткнутися з проблемами синхронізації, зокрема затримками або конфліктами при синхронізації файлів, особливо коли мають справу з великими файлами або ненадійним підключенням до Інтернету;
- доступ до файлів і папок OneDrive залежить від стабільного підключення до Інтернету, що може створювати проблеми в автономному режимі або в місцях з обмеженим підключенням;
- деяким користувачам інтерфейс користувача OneDrive може здатися надто складним, особливо під час навігації розширеними функціями або налаштуваннями.

Хоча Microsoft OneDrive позиціонується насамперед як хмарне сховище загального призначення, його функції та можливості роблять його актуальним у контексті систем керування зображеннями. Завдяки бездоганній інтеграції з Windows, офісним пакетом Office і крос-платформенній сумісності він підходить для зберігання та керування файлами зображень, а також іншими типами документів і медіафайлів. Крім того, функції для спільної роботи та покращення безпеки роблять його корисним у робочих процесах, орієнтованих на роботу з зображеннями, дозволяючи користувачам безпечно та ефективно співпрацювати над проектами зображень.

Dropbox – це хмарний файловий хостинг, який дозволяє користувачам зберігати, синхронізувати та обмінюватися файлами та папками на різних пристроях і платформах (див. рис. 1.3). Заснований у 2007 році Дрю Х'юстоном та Арашем Фердоусі, Dropbox перетворився на одного з провідних провайдерів хмарних сховищ, що обслуговує приватних осіб, компанії та команди по всьому світу.

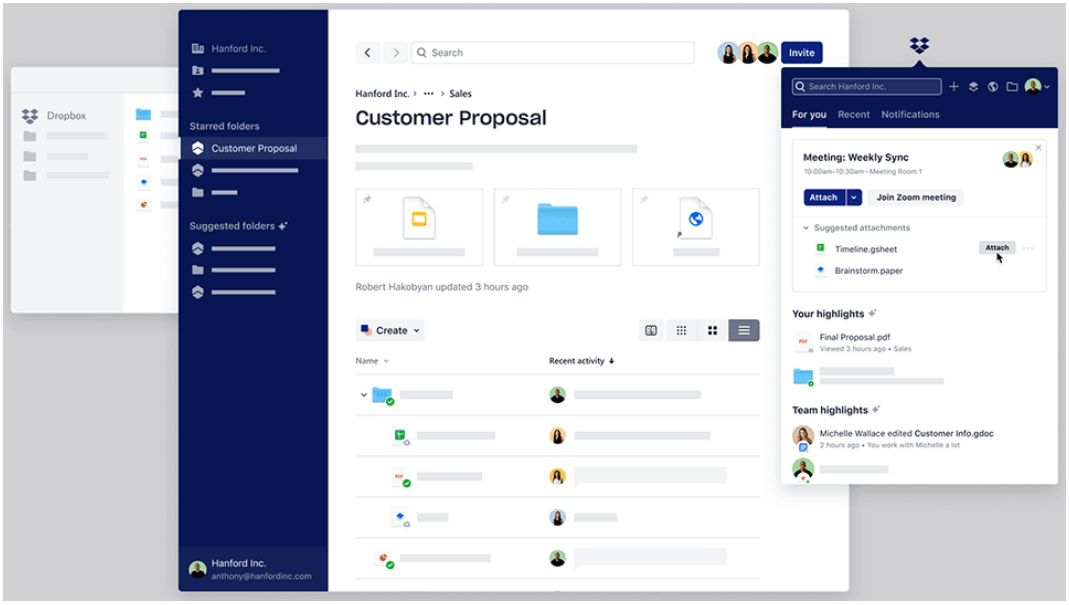


Рисунок 1.3 – Dropbox (рисунок створено самостійно)

Dropbox пропонує низку функцій, спрямованих на полегшення зберігання, синхронізації та спільної роботи з файлами. Основні функції включають:

– користувачі можуть завантажувати файли та папки до своїх акаунтів Dropbox, які потім надійно зберігаються у хмарі;
– Dropbox синхронізує файли та папки між пристроями, прив'язаними до одного облікового запису, гарантуючи, що зміни, зроблені на одному пристрої, відображаються на всіх інших пристроях;
– користувачі можуть ділитися файлами та папками з іншими за допомогою прямих посилань або спільних папок, що дозволяє безперешкодно співпрацювати та обмінюватися файлами між окремими особами та командами;

– Dropbox надає інструменти для спільної роботи, такі як коментарі до файлів, історія версій та попередній перегляд документів, що підвищує продуктивність та комунікацію між користувачами.

Dropbox слугує різним цілям для користувачів:

– це надійне рішення для резервного копіювання, що дозволяє користувачам зберігати свої файли та отримувати до них доступ з будь-якого місця, де є Інтернет-з'єднання;
– Dropbox сприяє співпраці та продуктивності, надаючи інструменти для безперешкодного обміну файлами, синхронізації та співпраці між окремими особами та командами;
– Dropbox обслуговує користувачів різних платформ, включаючи Windows, macOS, Linux, iOS, Android і веб-браузери, забезпечуючи сумісність та доступність на різних пристроях і операційних системах.

Переваги Dropbox:

– Dropbox може похвалитися зручним інтерфейсом, який є інтуїтивно зрозумілим і простим у навігації, що робить його доступним для користувачів усіх рівнів кваліфікації;
– Dropbox легко інтегрується з популярними інструментами та додатками для підвищення продуктивності, такими як Microsoft Office, Google Workspace, Slack та Adobe Creative Cloud, що підвищує ефективність робочого процесу та продуктивність;
– Dropbox сумісний з широким спектром пристроїв і платформ, що дозволяє користувачам отримувати доступ до своїх файлів зі стаціонарних комп'ютерів, ноутбуків, смартфонів і планшетів;
– Dropbox пропонує функції керування версіями файлів та їх відновлення, що дозволяє користувачам повернутися до попередніх версій файлів або відновити видалені файли протягом певного періоду часу.

Недоліки Dropbox:

– хоча Dropbox пропонує безкоштовний базовий план з обмеженим обсягом пам'яті, користувачам може знадобитися перейти на платний план, щоб

отримати доступ до додаткового обсягу пам’яті, що може спричинити додаткові витрати;

- користувачі можуть зіткнутися з проблемами синхронізації, такими як затримки або конфлікти при синхронізації файлів, особливо при роботі з великими файлами або ненадійним інтернет-з’єднанням;
- доступ до файлів і папок Dropbox залежить від стабільного Інтернет-з’єднання, що може створювати проблеми в автономному режимі або в місцях з обмеженим підключенням;
- Dropbox зіткнувся з ретельною перевіркою своїх практик безпеки та конфіденційності, включаючи витоки даних і занепокоєння щодо шифрування даних і конфіденційності користувачів.

Хоча Dropbox позиціонується насамперед як універсальний хмарний сервіс зберігання даних, його функції та можливості роблять його актуальним у контексті систем управління зображеннями. Безперебійна синхронізація файлів, інструменти для спільної роботи та крос-платформенна сумісність роблять його придатним для зберігання та управління файлами зображень поряд з іншими типами документів і медіа. Крім того, інтеграція з інструментами та програмами для підвищення продуктивності підвищує його корисність у робочих процесах, орієнтованих на зображення, дозволяючи користувачам ефективно співпрацювати над проектами зображень.

Flickr – це популярна онлайн-платформа для розміщення фото- та відеофайлів, яка дозволяє користувачам завантажувати, ділитися та впорядковувати свій візуальний контент (див. рис. 1.4). Заснований у 2004 році компанією Ludicorp, Flickr був придбаний Yahoo у 2005 році, а згодом – SmugMug у 2018 році. За ці роки Flickr перетворився на надійну платформу, керовану спільнотою, яка пропонує широкий спектр можливостей для фотографів, ентузіастів і творців контенту.

Flickr пропонує безліч функціональних можливостей, призначених для задоволення потреб фотографів і творців візуального контенту. Основні функції включають:

– користувачі можуть завантажувати фотографії та відео у свої акаунти на Flickr, де вони надійно зберігаються у хмарі;
– Flickr надає інструменти для впорядкування та тегування фотографій, що дозволяє користувачам класифікувати свій візуальний контент за темами, подіями та місцезнаходженням;
– користувачі можуть ділитися своїми фотографіями та відео з іншими за допомогою прямих посилань, платформ соціальних мереж або спільних альбомів, сприяючи залученню спільноти та взаємодії;
– Flickr дозволяє користувачам ліцензувати свої фотографії під різними ліцензіями Creative Commons, що дозволяє іншим використовувати та реміксувати їхній контент з належним зазначенням авторства;
– на Flickr існує жвава спільнота фотографів та ентузіастів, яка пропонує можливості для зворотного зв’язку, співпраці та відкриттів через групи, форуми та кураторські галереї.

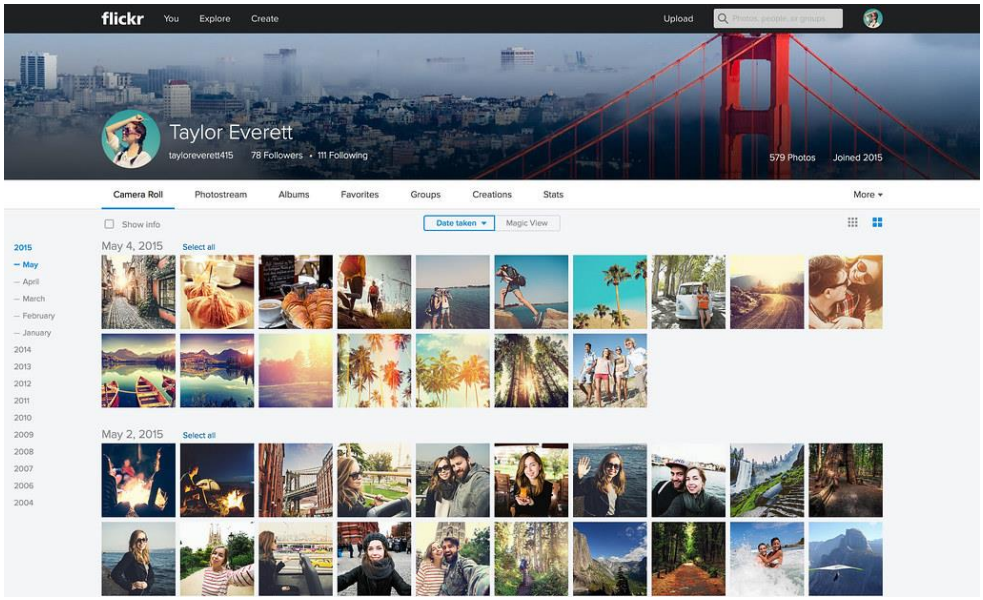


Рисунок 1.4 – Flickr (рисунок створено самостійно)

Flickr слугує різним цілям для користувачів:

– це платформа для фотографів і візуальних митців, на якій вони можуть демонструвати свої роботи, створювати портфоліо та здобувати популярність у фотоспільноті;

– Flickr сприяє створенню соціальних мереж серед фотографів та ентузіастів, дозволяючи їм спілкуватися, ділитися та взаємодіяти з однодумцями з усього світу;
– Flickr сприяє творчому самовираженню та дослідженню за допомогою фотографії та відео, надаючи користувачам платформу для експериментів з різними стилями, техніками та темами;
– це цифровий архів, в якому користувачі можуть зберігати свої візуальні спогади, що дозволяє їм переглядати та переживати минулий досвід за допомогою фотографій та відеозаписів.

Переваги Flickr:

– Flickr може похвалитися великою та різноманітною спільнотою фотографів, ентузіастів і творців контенту, пропонуючи можливості для нетворкінгу, співпраці та натхнення;
– Flickr пропонує варіанти відображення фотографій і відео у високій роздільній здатності, що дозволяє користувачам демонструвати свої роботи з приголомшливою деталізацією та чіткістю;
– можливість ліцензувати фотографії на умовах Creative Commons дозволяє користувачам ділитися своїми роботами з іншими, зберігаючи при цьому контроль над тим, як вони будуть використані та атрибутовані;
– Flickr пропонує надійні інструменти для впорядкування та тегування фотографій, що полегшує користувачам класифікацію та пошук контенту відповідно до їхніх інтересів та вподобань.

Недоліки Flickr:

– хоча Flickr пропонує безкоштовний базовий план з обмеженим обсягом пам’яті, користувачам може знадобитися перейти на платний план, щоб отримати доступ до додаткового обсягу пам’яті, що може спричинити додаткові витрати;
– в останні роки Flickr зазнав зниження популярності та залучення користувачів, зіткнувшись з конкуренцією з боку інших соціальних мереж та сервісів для обміну фотографіями;

- деякі користувачі можуть вважати користувацький інтерфейс Flickr перевантаженим або складним, особливо при навігації розширеними функціями або налаштуваннями;
- користувачі можуть мати сумніви щодо права власності та контролю над своїм контентом, особливо у світлі зміни власника платформи та умов надання послуг.

Хоча Flickr позиціонується насамперед як соціальна платформа для обміну фотографіями, її функції та можливості роблять її актуальною в контексті систем управління зображеннями. Його акцент на залученні спільноти, високоякісному відображенні та інструментах організації відповідає потребам користувачів, які прагнуть керувати та демонструвати свої колекції зображень. Крім того, варіанти ліцензування Creative Commons та інтеграція з платформами соціальних мереж підвищують її корисність у робочих процесах, орієнтованих на зображення, дозволяючи користувачам ділитися своїм контентом і поширювати його серед ширшої аудиторії.

Отже, індустрія управління зображеннями переживає стрімкий розвиток, зумовлений поширенням візуального контенту, проблемами великих даних, технологічним прогресом і зміною потреб користувачів. Щоб орієнтуватися в цьому динамічному ландшафті, потрібно надавати пріоритет дизайну, орієнтованому на користувача, щоб розробляти та впроваджувати ефективні, масштабовані та безпечні рішення для управління зображеннями, які дають можливість користувачам і організаціям використовувати весь потенціал візуальних даних.

Аналізуючи функціональні можливості, сильні та слабкі сторони чотирьох найвідоміших рішення для управління зображеннями: Google Photos, Microsoft OneDrive, Dropbox і Flickr, прагну забезпечити комплексне розуміння того, як ці сервіси відповідають вимогам сучасного управління зображеннями, особливо в контексті великих даних. Тому було проведено порівняння, яке висвітлить сфери, в яких кожен сервіс перевершує інші, а також ті, що потребують вдосконалення, надаючи цінну інформацію для розробки front-end частини нової та надійної веб-

орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних (див. табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця існуючих додатків та сервісів (таблиця створено самостійно)

Функція	Google Photos	Microsoft OneDrive	Dropbox	Flickr
Сховище для зберігання	Безкоштовне необмежене (стиснене) сховище	Безкоштовні та платні варіанти зберігання	Безкоштовні та платні варіанти зберігання	Безкоштовні та платні варіанти зберігання
Об’єм завантаження	Необмежений (стиснутий)	Залежить від плану зберігання	Залежить від плану зберігання	Залежить від плану зберігання
Час/швидкість завантаження/ вивантаження	Швидкий, залежить від швидкості Інтернету	Швидкий, інтегрований з Windows	Швидкий, залежить від швидкості Інтернету	Помірний, залежить від швидкості Інтернету
Критерії (метрики) пошуку	Дата, місце, люди, об’єкти	Назва файлу, тип, дата	Назва файлу, тип, дата	Теги, дата, місце, групи
Організація	Організація на основі штучного інтелекту	Папки, теги	Папки, теги	Альбоми, теги, групи
Інструменти редагування	Базове редагування)	Інтеграція з Office 365 (базове редагування)	Базове редагування	Базове редагування

Продовження таблиці 1.1

Параметри спільного доступу (варіанти обміну)	Прямі посилання, електронна пошта, соціальні мережі, спільні альбоми	Співпраця в режимі реального часу, спільні посилання, папки	Прямі посилання, спільні папки	Прямі посилання, соціальні мережі, спільні альбоми
Співпраця	Обмежена	Міцна інтеграція з офісом, співпраця в режимі реального часу	Коментарі, спільні папки, базова співпраця в режимі реального часу	Коментарі, групи, форуми
Безпека та конфіденційність	Проблеми зі збором даних, стиснені завантаження	Шифрування, багатофакторна автентифікація, запобігання втраті даних	Шифрування, минулі витоки даних	Занепокоєння щодо конфіденційності, зміна умов надання послуг
Платформи доступу	Web, iOS, Android, Windows, macOS	Web, iOS, Android, Windows, macOS, інтегрований в Windows	Web, iOS, Android, Windows, macOS	Web, iOS, Android, Windows, macOS

Продовження таблиці 1.1

Відстеження використання пам’яті	Відображає використання пам’яті	Відображає використання пам’яті	Відображає використання пам’яті	Відображає використання пам’яті
Переваги	Безкоштовне зберігання, пошук на основі штучного інтелекту, безшовна інтеграція з Google	Інтеграція з Windows та Office 365, надійні функції безпеки	Зручність у користуванні, інтеграція з інструментами підвищення продуктивності, керування версіями файлів	Дисплей з високою роздільною здатністю, ліцензія Creative Commons
Недоліки	Побоювання щодо конфіденційності, потенційна втрата якості зображення, потрібен доступ до Інтернету	Обмежене безкоштовне сховище, проблеми з синхронізацією	Обмеження на безкоштовне зберігання, проблеми з безпекою в минулому, потрібен доступ до Інтернету	Зниження популярності, складний інтерфейс, зміна власника

Отже, розроблена програмна система повинна усунути обмеження існуючих рішень, пропонуючи надійні функції, розширені можливості пошуку та організації,

безперешкодну інтеграцію з внутрішніми сервісами та надійні заходи безпеки. Співпраця, увага до деталей та дотримання найкращих практик мають вирішальне значення для створення високоякісного інтерфейсу, який покращить загальний користувацький досвід і функціональність системи.

1.2 Виявлення проблем та актуалізація рішень

При розробці інтерфейсу для веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, може виникнути кілька специфічних проблем. Ефективне вирішення цих проблем має важливе значення для забезпечення інтуїтивно зрозумілого, ефективного та зручного інтерфейсу системи. Нижче наведено поширені проблеми розробки інтерфейсу та відповідні рішення.

При обробці великих обсягів даних зображень може виникати низька швидкість рендерингу або неадекватна реакція інтерфейсу користувача. Тому можна впровадити такі методи, як ліниве завантаження, стиснення зображень та асинхронне завантаження для оптимізації продуктивності. Також можна використати такий фронтенд-фреймворк, як React, для ефективного управління рендерингом компонентів і оновленням станів, підвищуючи адаптивність користувацького інтерфейсу.

Невідповідність у рендерингу та поведінці різних веб-браузерів може призвести до проблем з юзабіліті для користувачів. Тому потрібно ретельно протестувати інтерфейс в різних браузерах і версіях, щоб виявити і вирішити проблеми сумісності. Також можна використати фреймворки CSS, такі як Bootstrap або Tailwind CSS, які пропонують вбудовану підтримку кросбраузерної сумісності, щоб забезпечити однаковий користувацький досвід на різних платформах.

Недостатня підтримка різних розмірів екранів і пристроїв може призвести до неоптимального користувацького досвіду на мобільних пристроях або планшетах. Тому потрібно застосувати мобільний підхід до розробки інтерфейсу, розробляючи інтерфейси з пріоритетом на зручність використання та читабельність на менших екранах. Також можна використати методи адаптивного дизайну, такі як медіа-

запити та гнучкі макети, щоб динамічно адаптувати інтерфейс до різних розмірів та роздільної здатності екрану.

Недотримання стандартів доступності може позбавити користувачів з обмеженими можливостями доступу до інтерфейсу та його ефективного використання. Тому потрібно дотримуватись найкращих практик, викладених у рекомендаціях щодо доступності, таких як WCAG (Web Content Accessibility Guidelines), щоб забезпечити доступність інтерфейсу для користувачів з різними потребами. Також можна використати семантичні елементи HTML, атрибути ARIA (Accessible Rich Internet Applications) та підтримку клавіатурної навігації для покращення доступності.

Надто складний або захаращений користувацький інтерфейс може заплутати користувачів і перешкоджати їхньому використанню. Тому потрібно спростити користувацький інтерфейс, визначивши пріоритети основних функцій та можливостей, видаливши непотрібні елементи та логічно організувавши контент..

Надмірне використання пам’яті або збої в роботі браузера можуть виникати при обробці великих наборів даних зображень або складних користувацьких взаємодій. Тому потрібно оптимізувати код інтерфейсу для ефективного використання пам’яті та продуктивності браузера, мінімізуючи непотрібні маніпуляції з DOM і витоки пам’яті. Впровадити такі методи, як віртуальна прокрутка та пагінація, щоб ефективно керувати великими наборами даних, не перевантажуючи браузер.

Отже, вирішуючи ці проблеми розробки інтерфейсу, можна створити надійний, ефективний та зручний інтерфейс для веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, покращуючи загальний користувацький досвід і зручність використання програми.

1.3 Постановка задачі

Стрімке зростання обсягів цифрових зображень в останні роки зумовило необхідність розробки надійних систем управління зображеннями, здатних

ефективно обробляти великі набори зображень. В умовах великих обсягів даних зростає попит на веб-рішення, які поєднують можливості обробки та аналізу зображень зі зручним інтерфейсом, доступним через веб-браузери. Існуючі рішення, такі як Google Photos, Microsoft OneDrive, Dropbox і Flickr, встановили високі стандарти завдяки таким функціям, як розширені можливості пошуку, безперешкодна інтеграція з іншими сервісами та надійні заходи безпеки. Однак ці рішення також мають обмеження, такі як конфіденційність, ліміти зберігання, проблеми синхронізації та складні користувацькі інтерфейси.

Задачею є спроектувати та реалізувати front-end частину для веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, зосередившись на наступних ключових аспектах:

- створити візуально привабливий та інтуїтивно зрозумілий користувацький інтерфейс, який полегшить ефективну навігацію та взаємодію з даними зображень;
- використовувати принципи адаптивного дизайну для забезпечення сумісності з різними розмірами екранів і пристроями, надаючи пріоритет зручності та доступності;
- реалізувати функції для завантаження, впорядкування та управління файлами зображень, включаючи перетягування та організацію папок;
- надати візуальний зворотній зв’язок та індикатори прогресу, щоб інформувати користувачів про стан завантаження зображень;
- додати метрики обсягу завантаження, часу та швидкості завантаження/вивантаження, а також використання сховища;
- розробити функції попереднього перегляду для перегляду файлів зображень з прев’ю;
- дозволити користувачам ефективно орієнтуватися у великих колекціях зображень;
- впровадити критерії для пошуку, зберігання, аналізу та класифікації зображень;

- реалізувати механізми автентифікації та авторизації користувачів для контролю доступу до даних зображень і функцій системи;
- надати функції для реєстрації користувачів, керування логінами та паролями, а також керування доступом на основі ролей, щоб забезпечити безпеку даних і конфіденційність користувачів;
- співпрацювати з бекенд-розробником для інтеграції фронтенд-компонентів з бекенд-сервісами, API та базами даних для безперебійного обміну даними та виконання функцій;
- забезпечити належну обробку помилок, перевірку даних та синхронізацію даних між фронтенд- та бекенд-компонентами;
- провести ретельне тестування фронтенду, щоб виявити та вирішити будь-які помилки та проблеми з продуктивністю або юзабіліті.

Отже, задача полягає в розробці та впровадженні зручного інтерфейсу для ефективної обробки та аналізу зображень. Співпраця, увага до деталей та дотримання найкращих практик мають вирішальне значення для створення високоякісного інтерфейсу, який покращує загальний користувацький досвід і функціональність системи. Враховуючи ці ключові аспекти, інтерфейсна частина веб-орієнтованої гібридної системи управління зображеннями може значно підвищити ефективність та результативність управління великомасштабними наборами даних зображень в контексті великих даних.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Інтерфейс слугує основною точкою взаємодії між користувачами та системою, відіграючи ключову роль у забезпеченні інтуїтивно зрозумілої навігації, безперешкодної роботи користувача та ефективного доступу до даних зображень [3]. У веб-орієнтованій системі гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних front-end частина слугує провідником, за допомогою якого користувачі можуть легко завантажувати, впорядковувати, переглядати, обробляти та аналізувати файли зображень. Функціональні вимоги наведені нижче.

Авторизація/реєстрація:

- користувачі повинні мати можливість зареєструвати обліковий запис і пройти безпечну автентифікацію для доступу до системи;
- процес реєстрації повинен збирати основну інформацію про користувача, забезпечуючи при цьому конфіденційність та безпеку;
- після успішної реєстрації користувачі повинні отримати підтвердження та мати можливість увійти в систему.

Пошук файлів:

- користувачі повинні мати можливість шукати конкретні файли зображень на основі таких критеріїв, як назва файлу, теги, метадані або дата завантаження;
- функція пошуку має бути інтуїтивно зрозумілою, швидко реагувати на запити та швидко надавати релевантні результати пошуку.

Додавання файлів:

- користувачі повинні мати можливість завантажувати файли зображень до системи зі своїх локальних пристроїв або зовнішніх джерел;
- процес завантаження повинен підтримувати різні формати файлів, включаючи популярні формати зображень, такі як JPEG, PNG і GIF;
- під час завантаження файлів повинні бути передбачені індикатори прогресу, щоб інформувати користувачів про статус завантаження.

Перегляд файлів:

- користувачі повинні мати можливість переглядати свої файли зображень як в режимі плиткового перегляду, так і в режимі списку, що дозволяє налаштувати різні параметри перегляду;
- для кожного файлу повинні відображатися мініатюри або прев'ю зображень, що дають візуальне уявлення про вміст;
- елементи керування навігацією повинні дозволяти користувачам ефективно переглядати великі колекції файлів зображень.

Перегляд інформації про файл:

- користувачі повинні мати доступ до детальної інформації про кожен файл зображення, включаючи метадані, такі як розмір файлу, роздільна здатність, дата створення та теги;
- інформація про файл має бути представлена у чіткий та організований спосіб, що полегшує її розуміння та пошук.

Завантажити файл:

- користувачі повинні мати можливість завантажувати файли зображень із системи на свої локальні пристрої для автономного доступу або зовнішнього використання;
- опції завантаження повинні бути легкодоступними та дозволяти користувачам вибирати декілька файлів для пакетного завантаження, якщо це необхідно.

Відкрити файл:

- користувачі повинні мати можливість відкривати файли зображень в інтерфейсі системи для перегляду та подальшої взаємодії;
- система повинна підтримувати різні режими перегляду зображень, включаючи повноекранний режим, масштабування та панорамування для детального ознайомлення з вмістом зображень.

Копіювання, вирізання, вставка файлів:

- користувачі повинні мати можливість копіювати, вирізати та вставляти файли зображень між директоріями в системі;
- операції копіювання, вирізання та вставки мають бути інтуїтивно зрозумілими та підтримувати вибір як одного, так і декількох файлів.

Редагування назви теки, видалення теки:

- користувачі повинні мати можливість перейменовувати теки та видаляти теки, якщо це необхідно для впорядкування їхніх файлів зображень;
- дії з редагування та видалення папок мають супроводжуватися запитами на підтвердження, щоб запобігти випадковій втраті даних.

Перейменування та видалення файлів:

- користувачі повинні мати можливість перейменовувати файли зображень та видаляти файли з системи за необхідності;
- процедури перейменування та видалення файлів мають бути простими та включати діалогові вікна підтвердження, щоб запобігти ненавмисним змінам.

Відображення використання пам’яті дисплея:

- система повинна надавати користувачам інформацію про обсяг пам’яті, використаний для зберігання файлів зображень;
- статистика використання пам’яті повинна відображатися у чіткому та зрозумілому форматі, динамічно оновлюючись при додаванні або видаленні файлів.

Реалізація гібридного механізму управління зображеннями:

- система повинна підтримувати одночасне завантаження великого обсягу зображень з можливістю витримувати високі навантаження трафіку;
- зображення повинні завантажуватися та вивантажуватися ефективно, з мінімальною затримкою та високою швидкістю передачі;
- система повинна надавати масштабовані рішення для зберігання зображень, що зростають, з можливістю зберігання як у хмарі, так і в локальному сховищі;
- зображення повинні зберігатися на основі відповідних критеріїв, таких як дата, місцезнаходження, категорія та визначені користувачем теги. Слід враховувати показники ефективності зберігання, такі як використання простору сховища та надмірність даних.

Реалізація обробки та аналізу:

- впровадити розширені можливості пошуку на основі атрибутів зображень, таких як колір, розмір, роздільна здатність і метадані. Слід враховувати

показники точності, швидкості та релевантності пошуку;

- реалізувати алгоритми аналізу зображень для таких завдань, як виявлення об’єктів, класифікація зображень та сегментація зображень. Слід оцінити показники точності аналізу, часу обробки та використання ресурсів;
- розробити моделі класифікації зображень на основі алгоритмів машинного навчання, навчених на маркованих наборах даних. Слід оцінити метрики точності класифікації, продуктивності моделі та часу навчання;
- інтеграція зі сторонніми інструментами і бібліотеками обробки та аналізу зображень, такими як OpenCV, TensorFlow та PyTorch, для використання наявних можливостей та розширення функціональності;
- впровадити можливості обробки та аналізу в реальному часі, щоб обробляти потокові дані зображень та надавати своєчасну інформацію. Слід вимірювати показники швидкості обробки, затримки та пропускної здатності;
- потрібно переконатися, що компоненти обробки та аналізу масштабуються та можуть ефективно справлятися зі зростаючими робочими навантаженнями. Показники масштабованості, продуктивності та використання ресурсів слід відстежувати та оптимізувати.

Нефункціональні вимоги наведені нижче.

Дизайн інтерфейсу користувача:

- інтерфейс повинен мати чистий, сучасний дизайн з інтуїтивно зрозумілою навігацією та візуально привабливими елементами;
- елементи інтерфейсу повинні бути адаптивними та оптимізованими для зручності використання на різних пристроях та розмірах екранів.

Продуктивність:

- інтерфейс повинен бути чуйним і продуктивним, забезпечувати швидке завантаження та безперебійну взаємодію;
- операції обробки зображень повинні бути ефективними, мінімізуючи затримки і затримки у відображенні контенту.

Безпека:

- механізми автентифікації та авторизації користувачів повинні бути надійними та безпечними, захищаючи дані користувачів від несанкціонованого доступу або маніпуляцій;
- процеси завантаження та вивантаження файлів повинні бути зашифровані для забезпечення цілісності та конфіденційності даних.

Масштабованість:

- інтерфейс повинен бути спроектований таким чином, щоб його можна було легко масштабувати в міру зростання системи, пристосовуючи до зростаючої кількості користувачів і файлів зображень;
- міркування масштабованості повинні бути враховані, щоб запобігти погіршенню продуктивності або збоям системи під великим навантаженням.

Сумісність:

- інтерфейс повинен бути сумісним з широким спектром веб-браузерів і пристроїв, забезпечуючи однаковий користувацький досвід на різних платформах;
- слід провести тестування сумісності, щоб виявити та вирішити будь-які проблеми, пов’язані з поведінкою браузера або пристрою.

Юзабіліті: інтерфейс повинен бути зручним для користувача, з інтуїтивно зрозумілими робочими процесами та мінімальною тривалістю навчання для користувачів з різним рівнем кваліфікації.

Отже, формування детальних вимог до розробки front-end частинb веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних гарантує ефективне задоволення потреб користувачів при дотриманні високих стандартів зручності використання, продуктивності, безпеки, масштабованості та сумісності. Розмежувавши функціональні та нефункціональні вимоги, можна створити надійний та зручний інтерфейс, який легко інтегрується з внутрішніми сервісами, забезпечує оптимальну продуктивність та інтуїтивно зрозумілу роботу на різних пристроях і платформах. Дотримання цих вимог є важливим для успішної реалізації цілей системи та задоволення очікувань користувачів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Діаграма варіантів використання допомагає окреслити різні функціональні можливості та взаємодії, які користувачі матимуть із системою (див. рис. 3.1). Вона гарантує, що розробник має чітке розуміння вимог до системи та допомагає керувати процесом проектування та впровадження [4].

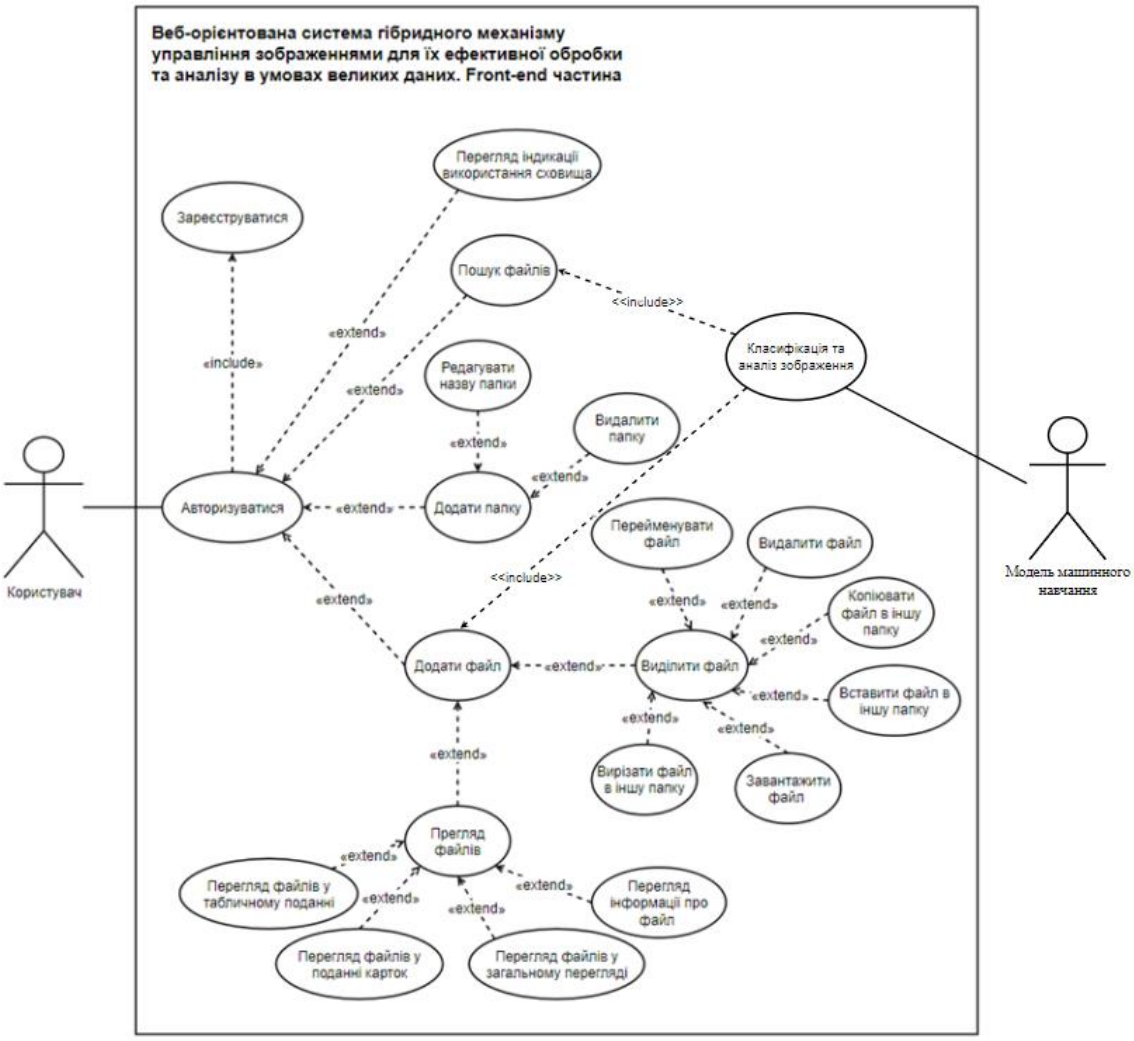


Рисунок 3.1 – Діаграма варіантів використання (рисунок створено самостійно)

Користувачі можуть зареєструвати обліковий запис або увійти в систему, щоб отримати доступ до системи. Вони можуть шукати певні файли зображень на основі різних критеріїв. Користувачі можуть завантажувати файли зображень до системи. Користувачі можуть переглядати та візуалізувати свої файли зображень у різних режимах. Можна отримати детальну інформацію про конкретний файл зображення.

Користувачі мають можливість завантажити файли зображень на свої локальні пристрої. Файли зображень можна відкрити в інтерфейсі системи для перегляду або подальшої взаємодії. Підтримуються операції керування файлами, такі як копіювання, вирізання та вставка між папками. Користувачі можуть редагувати назви папок або видаляти папки за потреби. Є можливість перейменовувати або видаляти окремі файли зображень. Система відображає інформацію про обсяг пам’яті, використаний для зберігання файлів зображень.

Діаграма розгортання необхідна для розуміння того, як різні компоненти веб-орієнтованої системи розподіляються між різними вузлами або серверами в мережевому середовищі (див. рис. 3.2). Вона допомагає зацікавленим сторонам зрозуміти фізичне розміщення компонентів, канали зв’язку, можливості масштабування, вимоги до інфраструктури та конфігурації розгортання, що полегшує ефективне розгортання та обслуговування системи [5].

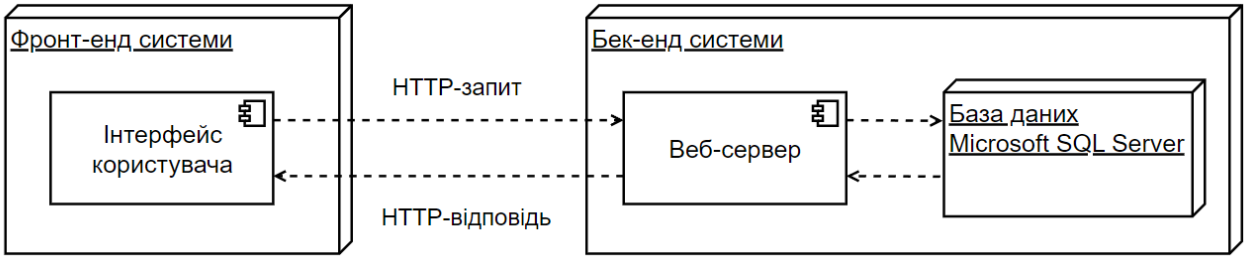


Рисунок 3.2 – Діаграма розгортання (рисунок створено самостійно)

Діаграма розгортання складається з двох основних частин: інтерфейсу та серверної частини. Фронт-енд представляє користувацький аспект системи, включаючи користувацький інтерфейс. Цей інтерфейс дозволяє користувачам взаємодіяти з функціональними можливостями системи. HTTP-запити надходять від компонента користувацького інтерфейсу та надсилаються до внутрішньої частини системи для обробки. Внутрішня частина системи складається з декількох компонентів, що відповідають за обробку, зберігання даних та надання відповідей на запити клієнтів. База даних і компоненти Microsoft SQL Server зберігають та керують даними зображень та пов’язаною з ними інформацією. Веб-сервер містить логіку програми та забезпечує

зв’язок між зовнішнім і внутрішнім компонентами. HTTP-відповіді генеруються бекенд-компонентами та надсилаються назад на фронт-енд у відповідь на запити клієнтів.

Діаграма діяльності відображає послідовність дій та взаємодій у системі, я також включає кілька компонентів, таких як клієнт, веб-сервер і сервер бази даних (див. рис. 3.3). Вона забезпечує чітке зображення хронологічного порядку дій, що виконуються різними компонентами системи, від ініціації запиту до передачі даних назад клієнту [6].

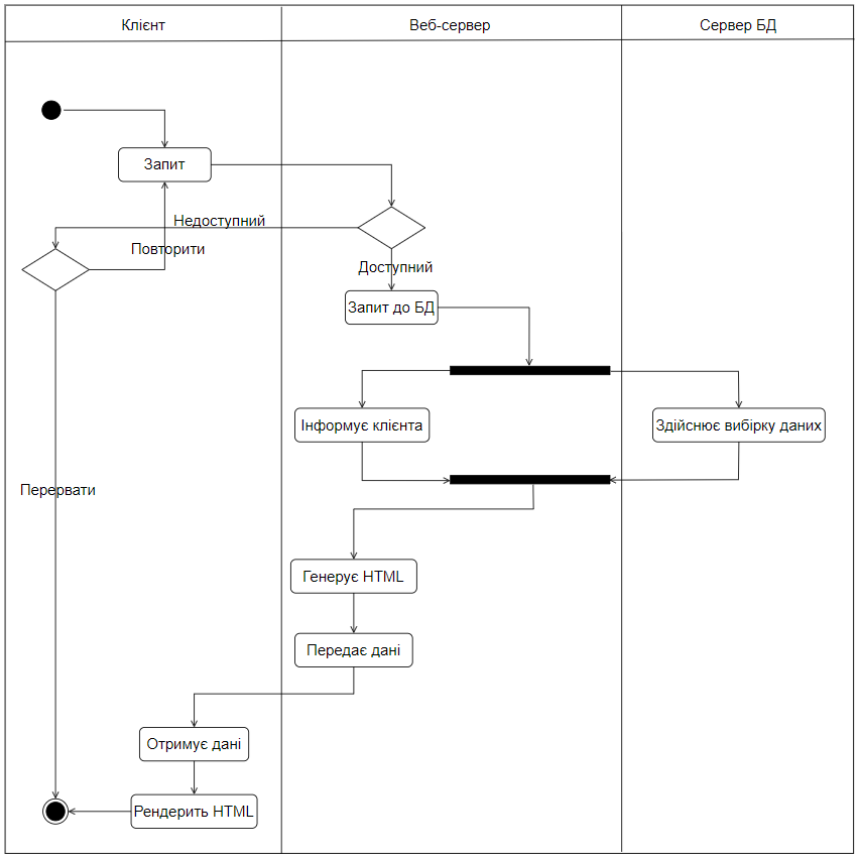


Рисунок 3.3 – Діаграма діяльності (рисунок створено самостійно)

Процес починається з того, що клієнт ініціює запит до веб-сервера. Веб-сервер отримує запит і перевіряє його доступність. Якщо запитувані дані недоступні, він переходить у стан переривання, вказуючи на те, що дані недоступні в даний момент. Як тільки дані стають доступними, веб-сервер продовжує запитувати сервер бази даних, щоб отримати необхідну інформацію. У відповідь сервер бази даних передає дані веб-серверу. Отримавши дані, веб-сервер обробляє їх, створюючи HTML-контент, який потім надсилається назад клієнту. Клієнт отримує HTML-контент і повторює процес за потреби. Під час цього процесу можуть виникати перерви або затримки, наприклад, коли

запитувані дані спочатку недоступні або коли необхідна вибірка або обробка даних. Однак після завершення необхідних дій веб-сервер інформує клієнта і передає необхідні дані, забезпечуючи безперебійну взаємодію між клієнтом і системою.

Отже, при UML проєктування ПЗ ці діаграми в сукупності дають комплексне уявлення про програмну систему, охоплюючи її функціональні вимоги, статичну структуру та фізичну архітектуру розгортання. Вони слугують цінними інструментами для проектування та розробки.

3.2 Проєктування архітектури ПЗ

Клієнт-серверна архітектура – це фундаментальна концепція мережевих обчислень, де завдання та ресурси розподіляються між клієнтами та серверами [7].

В архітектурі клієнт-сервер клієнт – це будь-який комп'ютер або пристрій, який запитує послуги або ресурси у сервера. Це може включати доступ до веб-сторінок, завантаження файлів або запит даних. І навпаки, сервер – це спеціальний комп'ютер або програмне забезпечення, призначене для надання послуг або ресурсів клієнтам [8]. Сервери обробляють клієнтські запити, обробляють дані та надають відповідні відповіді (див. рис. 3.4).

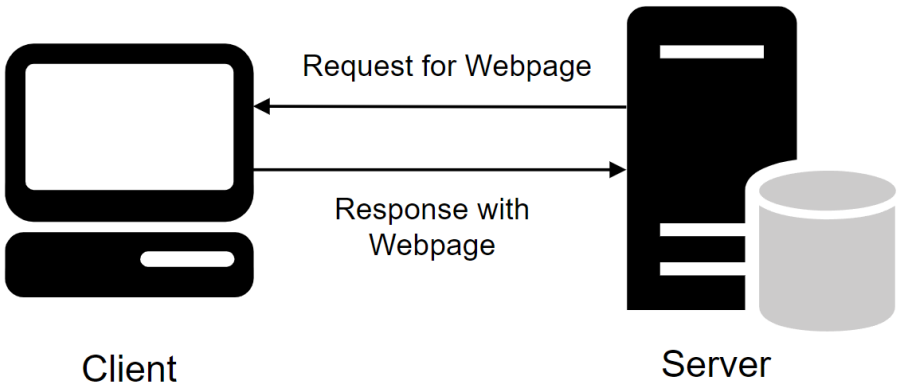


Рисунок 3.4 – Архітектура клієнт-сервер (рисунок створено самостійно)

Архітектура клієнт-сервер – це мережева модель, яка розподіляє завдання між клієнтами та серверами, як в межах однієї системи, так і в мережі. Клієнти ініціюють запити на послуги, а сервери виконують ці запити, надаючи ресурси та обробляючи дані. Ця архітектура полегшує комунікацію та спільне використання ресурсів між кількома клієнтами та серверами.

Коли клієнту потрібні дані або послуги від сервера, він надсилає запит на IP-адресу сервера. Система доменних імен (DNS) перетворює доменні імена на IP-адреси, дозволяючи клієнтам знаходити сервери. Запит містить конкретний номер порту, пов’язаний з потрібною програмою на сервері. Сервер обробляє запит і надсилає клієнту відповідь, яка використовується відповідною програмою.

Переваги клієнт-серверної архітектури:

- централізоване зберігання на серверах спрощує управління файлами та їх пошук, підвищуючи організованість та ефективність;
- клієнти можуть отримати доступ до послуг і ресурсів з будь-якого місця, використовуючи будь-який сумісний пристрій, що забезпечує гнучкість і зручність;
- клієнт-серверні мережі можна легко масштабувати, щоб пристосувати до зростаючої кількості користувачів і ресурсів без значних перебоїв у роботі, що сприяє масштабованості та адаптивності;
- централізоване адміністрування забезпечує ефективне управління та обслуговування мережевих ресурсів, полегшуючи оновлення, резервне копіювання та заходи безпеки;
- централізована архітектура забезпечує надійні заходи безпеки, включаючи аутентифікацію користувачів, контроль доступу та шифрування даних, захищаючи конфіденційну інформацію від несанкціонованого доступу.

Отже, клієнт-серверна архітектура відіграє ключову роль в мережах і комунікаціях, забезпечуючи ефективний обмін ресурсами, обробку даних і надання послуг. Масштабованість, доступність і централізоване управління роблять її незамінною для організацій будь-якого розміру, сприяючи підвищенню продуктивності, співпраці та цифровій трансформації. Розуміючи принципи та переваги архітектури клієнт-сервер, її можливості можна використовувати для покращення своєї діяльності та збереження конкурентоспроможності в цифровому середовищі.

3.3 Проєктування структури зберігання даних

Структура зберігання даних передбачає проектування комплексної схеми бази даних та її реалізацію за допомогою скриптів Python для взаємодії з базою даних MSSQL.

Схема бази даних – це план, який визначає структуру бази даних, включаючи таблиці, стовпці, типи даних, зв'язки та обмеження. Схема бази даних включає такі таблиці:

- таблиця "Зображення" зберігає інформацію про завантажені зображення, таку як назва файлу, шлях до файлу, розміри, дата створення та інші відповідні метадані;
- таблиця "Користувачі" містить інформацію про користувачів, зокрема імена користувачів, паролі, які хешовані для безпеки та адреси електронної пошти;
- таблиця "Теги" містить список тегів або категорій, пов'язаних із зображеннями, для їх упорядкування та класифікації;
- таблиця "Метадані" зберігає додаткові метадані, витягнуті із зображень, такі як модель камери, GPS-координати, роздільна здатність зображення та інші дані EXIF.

Скрипти Python встановлюють з'єднання з базою даних MSSQL за допомогою відповідної бібліотеки з'єднувачів баз даних, наприклад, pyodbc або SQLAlchemy [9]. Рядок з'єднання містить важливі деталі, такі як ім'я сервера, ім'я бази даних, облікові дані для автентифікації та інші параметри з'єднання.

Скрипти Python виконують команди SQL для створення необхідних таблиць у базі даних MSSQL. Кожне визначення таблиці включає специфікації для стовпців, типів даних, первинних ключів, зовнішніх ключів, обмежень та індексів [10].

Скрипти Python взаємодіють з базою даних для вставки нових записів та отримання існуючих даних за допомогою команд SQL.

Скрипти Python використовують бібліотеки обробки зображень, такі як Pillow або OpenCV, для вилучення метаданих або виконання завдань аналізу зображень. Витягнуті метадані або результати аналізу можна зберігати в таблиці "Метадані" або інших відповідних таблицях для подальшої обробки або аналізу.

Скрипти Python реалізують заходи безпеки, такі як параметризовані запити та перевірка вхідних даних, щоб запобігти атакам SQL-ін'єкцій. Також реалізовано механізми автентифікації та авторизації для контролю доступу до конфіденційних даних і функцій.

Схема бази даних і SQL-запити оптимізовані для підвищення продуктивності шляхом створення відповідних індексів на часто запитувані стовпці, розбиття великих таблиць та оптимізації планів виконання SQL-запитів для забезпечення ефективного пошуку і обробки даних [11].

Скрипти Python включають механізми обробки помилок та ведення журналів, щоб ефективно перехоплювати та обробляти винятки. Помилки та попередження реєструються для налагодження та усунення несправностей, забезпечуючи надійність та цілісність системи.

Система розроблена з можливістю масштабування, що дозволяє збільшувати або зменшувати обсяг даних і трафік користувачів у міру необхідності. Для забезпечення постійної надійності та доступності системи впроваджено автоматизовані процедури резервного копіювання та обслуговування.

Отже, структура зберігання даних у веб-орієнтованій системі гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних ретельно розроблена та впроваджена для ефективного зберігання, пошуку та управління даними зображень, забезпечуючи при цьому безпеку, продуктивність та масштабованість.

3.4 Приклади найцікавіших алгоритмів та методів

Завантаження файлів є поширеною вимогою, з якою стикаються різні додатки. Важливо забезпечити користувачам безперебійний та інтуїтивно зрозумілий процес завантаження файлів, а також зворотній зв'язок про хід процесу завантаження.

Цей фрагмент коду демонструє завантаження файлів на сервер з індикатором прогресу для відстеження прогресу завантаження:

```
<input type="file" id="fileInput">
<div id="progressBar"></div>
```

HTML-частина складається з елемента input типу "file" з ідентифікатором "fileInput" та елемента div з ідентифікатором "progressBar". Елемент input дозволяє користувачам вибрати файл з локального пристрою, в той час як елемент div слугуватиме індикатором прогресу для візуального відображення прогресу завантаження.

Частина JavaScript починається з вибору елемента введення файлу та елемента смуги прогресу за їхніми відповідними ідентифікаторами:

```
const fileInput = document.getElementById('fileInput');
const progressBar = document.getElementById('progressBar');
```

Потім до елемента введення файлу додається слухач події "change", який спрацьовує, коли користувач вибирає файл:

```
fileInput.addEventListener('change', (event) => {
  const file = event.target.files[0];
```

Усередині функції-слухача події вибраний файл отримується з об'єкта події за допомогою властивості "files". Для зберігання даних файлу створюється об'єкт FormData, і вибраний файл додається до нього за допомогою методу append:

```
  const formData = new FormData();
  formData.append('file', file);
```

Далі створюється об'єкт XMLHttpRequest для асинхронного POST-запиту до сервера:

```
  const xhr = new XMLHttpRequest();
```

У запиті вказується метод HTTP ("POST"), URL-адреса "/upload" і прапорець асинхронності, встановлений у значення "true":

```
  xhr.open('POST', '/upload', true);
```

Для події "progress" процесу завантаження визначено приймач подій, який оновлює ширину індикатора прогресу на основі відсотка завершення, розрахованого за допомогою властивостей "progressEvent.loaded" і "progressEvent.total":

```
xhr.upload.onprogress = (progressEvent) => {
   if (progressEvent.lengthComputable) {
     const percentCompleted = (progressEvent.loaded / progressEvent.total)
* 100;
     progressBar.style.width = percentCompleted + '%';}};
```

Для події "load" визначено ще один слухач подій, який обробляє завершення процесу завантаження. Якщо статус об'єкта XMLHttpRequest дорівнює 200, що означає успішне завантаження, на консоль виводиться повідомлення. В іншому випадку – повідомлення про помилку:

```
xhr.onload = () => {
  if (xhr.status === 200) {
    console.log('File uploaded successfully');
  } else {
    console.error('File upload failed');}  };
```

Потім об'єкт FormData, що містить дані файлу, надсилається на сервер за допомогою методу "send" об'єкта XMLHttpRequest:

```
xhr.send(formData);});
```

Отже, цей фрагмент коду демонструє, як реалізувати функцію завантаження файлів з індикатором виконання за допомогою об'єкта XMLHttpRequest в JavaScript, що дозволяє користувачам завантажувати файли на сервер і відстежувати хід завантаження в режимі реального часу.

Цей код цікавий з кількох причин:

– він надає користувачам зворотній зв'язок у реальному часі про хід завантаження їхніх файлів за допомогою індикатора прогресу. Це покращує користувацький досвід, надаючи користувачам візуальну інформацію про те, який файл було завантажено;

- код використовує XMLHttpRequest для асинхронних запитів до сервера. Це означає, що користувач може продовжувати взаємодіяти з веб-сторінкою, не чекаючи завершення завантаження, що покращує швидкість відгуку та зручність використання;
- ширина індикатора виконання динамічно оновлюється залежно від відсотка завантаженого файлу. Така динамічна поведінка додає користувацькому інтерфейсу інтерактивності, роблячи процес завантаження більш цікавим та інформативним;
- код включає обробку помилок для випадків, коли завантаження не вдається. Це гарантує, що користувачі отримують повідомлення про будь-які проблеми із завантаженням, що дозволяє їм вжити відповідних заходів або повторити спробу завантаження, якщо це необхідно;
- код відносно простий та зрозумілий, що робить його легким для розуміння та інтеграції в існуючі веб-додатки. Він демонструє фундаментальний аспект фронтенд-розробки – обробку завантаження файлів у чіткій та лаконічній формі.

Також цей код демонструє практичний та ефективний підхід до реалізації завантаження файлів із зворотним зв’язком у реальному часі у веб-орієнтованій системі гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, покращуючи користувацький досвід та зручність використання додатку.

У сучасних веб-додатках надання користувачам ефективної пошукової функціональності має важливе значення для підвищення зручності використання та забезпечення швидкого доступу до потрібного контенту. Цей фрагмент коду демонструє реалізацію функції пошуку зображень за допомогою JavaScript.

HTML-частина складається з елемента input типу "text" з ідентифікатором "searchInput" та елемента div з ідентифікатором "imageGallery". Елемент input дозволяє користувачам вводити пошукові терміни, тоді як елемент div слугуватиме контейнером для відображення відфільтрованих зображень:

```
<input type="text" id="searchInput">
<div id="imageGallery"></div>
```

Частина JavaScript починається з вибору елемента введення та контейнера галереї зображень за їхніми відповідними ідентифікаторами. Потім до елемента вводу додається слухач події "input", який спрацьовує щоразу, коли користувач вводить або змінює текст у полі вводу:

```
const searchInput = document.getElementById('searchInput');
const imageGallery = document.getElementById('imageGallery');
const images = ['image1.jpg', 'image2.jpg', 'image3.jpg', 'image4.jpg'];
```

Усередині функції-слухача події код отримує пошуковий термін, введений користувачем, і перетворює його в нижній регістр, щоб виконати пошук без урахування регістру.

```
searchInput.addEventListener('input', (event) => {
  const searchTerm = event.target.value.toLowerCase();
```

Потім він фільтрує список зображень на основі того, чи містить URL-адреса зображення пошуковий запит, за допомогою методу Array.filter():

```
    const filteredImages = images.filter(image =>
image.toLowerCase().includes(searchTerm));
```

Відфільтрований список зображень передається функції renderImages(), яка очищає вміст контейнера галереї зображень та повторно переглядає список відфільтрованих зображень. Для кожного зображення вона створює новий елемент img, встановлює його атрибут src на URL-адресу зображення, додає клас CSS для стилізації і додає його до контейнера галереї зображень:

```
  renderImages(filteredImages);
});
function renderImages(imageList) {
  imageGallery.innerHTML = '';
  imageList.forEach(image => {
    const imgElement = document.createElement('img');
```

```
imgElement.src = image;
  imgElement.classList.add('image-thumbnail');
  imageGallery.appendChild(imgElement);});}
```

Отже, цей фрагмент коду демонструє простий, але ефективний спосіб реалізації функції пошуку зображень у фронтенд-частині, що дозволяє користувачам фільтрувати та відображати зображення на основі їхніх пошукових запитів.

Цей код цікавий з кількох причин:

- він дозволяє користувачам динамічно шукати зображення під час введення тексту, забезпечуючи миттєвий зворотній зв'язок і покращуючи користувацький досвід. Функція пошуку в реальному часі дозволяє користувачам швидко знаходити зображення, які вони шукають, без необхідності перезавантаження сторінки або додаткових кліків;
- код ефективно фільтрує список зображень на основі пошукового запиту користувача за допомогою вбудованого в JavaScript методу Array.filter(). Це гарантує, що користувачеві будуть показані лише релевантні зображення, що відповідають критеріям пошуку, зменшуючи когнітивне навантаження та покращуючи зручність використання;
- функціонал пошуку легко інтегрується у зовнішній інтерфейс, що дозволяє користувачам безперешкодно взаємодіяти з додатком. Інтерфейс миттєво реагує на введення даних, забезпечуючи плавний та чуйний досвід, що підвищує задоволеність та залученість користувачів;
- код лаконічний, добре структурований та легкий для розуміння, що робить його доступним. Завдяки використанню простої логіки JavaScript та елементів HTML, він демонструє ефективний підхід до реалізації функціональності пошуку зображень без зайвої складності;
- код можна легко розширити та адаптувати для роботи з більшими наборами даних або додатковими функціями за потреби. Незалежно від того, чи це інтеграція з внутрішньою базою даних, чи реалізація більш досконалих алгоритмів пошуку, код забезпечує міцну основу для майбутнього розвитку та розширення.

Також цей код демонструє практичний та ефективний підхід до реалізації функціональності динамічного пошуку зображень у фронтенд-частині. Його простота, гнучкість та масштабованість роблять його цінним доповненням до веб-орієнтованих систем гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних, покращуючи зручність використання та полегшуючи ефективний пошук зображень для користувачів.

Отже, наведені фрагменти коду ілюструють реалізацію динамічних та інтерактивних функцій у фронтенд-середовищі для веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних

Перший фрагмент коду демонструє можливість завантажувати файли зі зворотним зв’язком у реальному часі, покращуючи взаємодію з користувачем шляхом надання візуальних підказок під час процесу завантаження. Ця функціональність є важливою для додатків, що мають справу з великими файлами зображень, забезпечуючи інформування користувачів про хід завантаження та зменшуючи невизначеність.

Другий фрагмент коду демонструє функцію динамічного пошуку зображень, що дозволяє користувачам фільтрувати зображення в режимі реального часу на основі їхніх пошукових запитів. Ця функція підвищує зручність використання, дозволяючи користувачам швидко знаходити конкретні зображення в колекції, підвищуючи ефективність та задоволеність користувачів.

Обидва фрагменти підкреслюють важливість адаптивності та інтерактивності при розробці інтерфейсу, що сприяє безперебійному користувацькому досвіду. Завдяки включенню цих функцій, система може краще відповідати потребам користувачів і спростити завдання обробки та аналізу зображень.

3.5 Створення UI / UX дизайну системи

UI/UX дизайн веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних має вирішальне значення для надання користувачам безперешкодного та інтуїтивно зрозумілого досвіду під час взаємодії з платформою.

Сторінка "Створити обліковий запис" є важливим компонентом процесу автентифікації користувачів у веб-орієнтованій системі гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних (див. рис. 3.5).

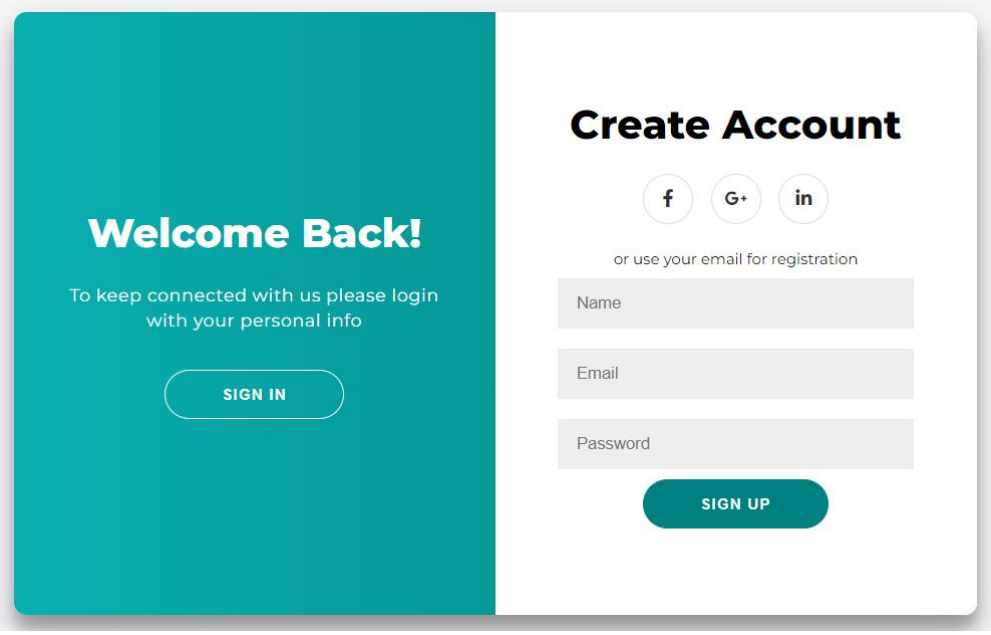


Рисунок 3.5 – Сторінка "Створити обліковий запис" (рисунок створено самостійно)

Основна частина сторінки містить форму створення облікового запису, за допомогою якої користувачі можуть зареєструвати новий обліковий запис. Форма містить такі поля:

- користувачам пропонується ввести своє повне ім’я або псевдонім, якому вони надають перевагу. Це додає їхньому обліковому запису індивідуальності та полегшує персоналізовану взаємодію в системі;
- користувачі повинні ввести дійсну адресу електронної пошти, яка слугує їхнім унікальним ідентифікатором для входу в систему та отримання повідомлень;
- користувачам пропонується створити пароль для свого облікового запису. Поле для введення пароля містить такі вимоги, як мінімальна довжина, спеціальні символи або комбінація букв і цифр для забезпечення надійності пароля;

– помітна кнопка "Зареєструватися" дозволяє користувачам заповнити форму і створити свій обліковий запис. Натискання цієї кнопки ініціює процес реєстрації облікового запису.

На цій сторінці використовуються персоналізовані повідомлення, щоб привітати користувачів і заохотити їх до створення облікового запису. Ці повідомлення мають бути доброзичливими, заохочувальними та переконливими, щоб спонукати користувачів до дії і завершити процес реєстрації.

Сторінка включає в себе елементи візуального дизайну, такі як кольори та типографіка, щоб створити візуально привабливий та цілісний користувацький досвід. Дизайн узгоджений з усією веб-орієнтованою системою, використовуючи кольори та зображення, які відображають ідентичність та призначення системи.

Під формою створення облікового запису користувачам пропонується альтернативний варіант входу. Це дозволяє користувачам, які вже мають обліковий запис, увійти, використовуючи свої існуючі облікові дані, замість того, щоб створювати новий обліковий запис (див. рис. 3.6).

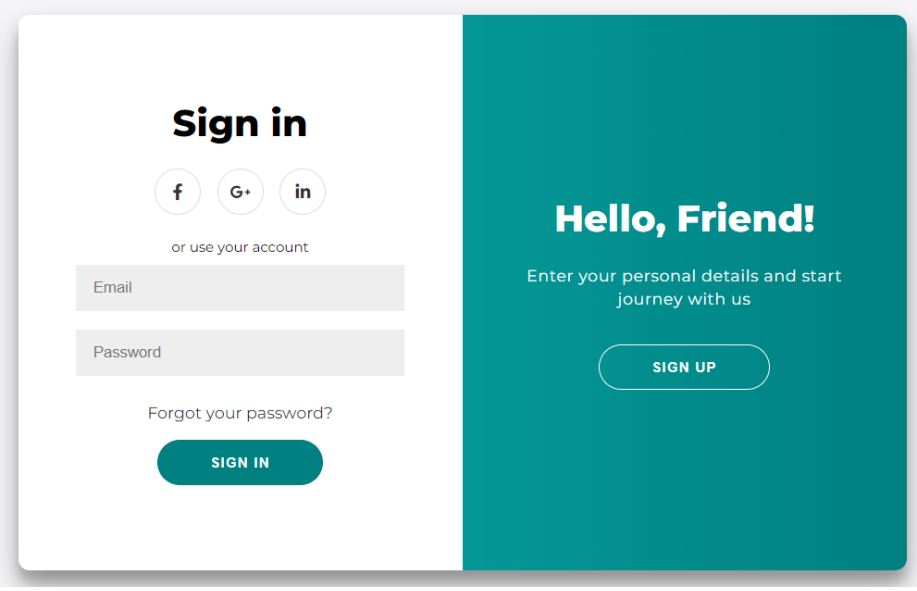


Рисунок 3.6 – Сторінка "Входу" (рисунок створено самостійно)

Альтернативний варіант входу включає в себе:

– посилання або кнопка з написом "Увійти" перенаправляє користувачів на сторінку входу, де вони можуть ввести свою електронну пошту і пароль для входу;

– посилання з написом "Забули пароль?" надає користувачам можливість скинути пароль, якщо вони його забули. Натиснувши на це посилання, користувач перенаправляється на сторінку скидання пароля, де він може відновити доступ до свого облікового запису.

Для організації елементів на сторінках використовуються чіткі та інтуїтивно зрозумілі макети та інтервали, що гарантують, що користувачі можуть легко розуміти та взаємодіяти з контентом.

Отже, сторінка "Створити обліковий запис" призначена для полегшення процесу реєстрації облікового запису у веб-орієнтованій системі гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Вона надає користувачам простий та інтуїтивно зрозумілий інтерфейс для створення нового облікового запису, а також пропонує альтернативні варіанти входу для користувачів, які вже мають обліковий запис. Завдяки персоналізованим повідомленням та елементам візуального дизайну сторінка має на меті залучити користувачів і заохотити їх до ефективного завершення процесу реєстрації.

Також була створена анімація завантаження, яка призначена для надання користувачам візуального зворотного зв’язку під час очікування завантаження сторінки або виконання певної дії (див. рис. 3.7).

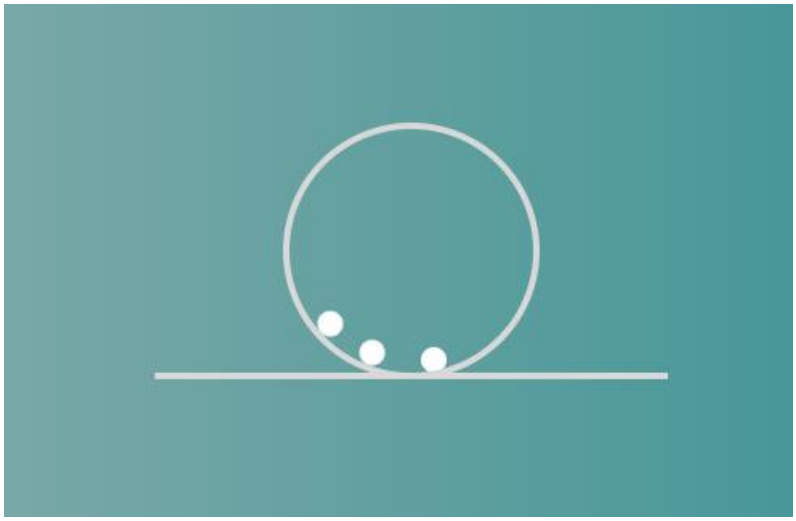


Рисунок 3.7 – Анімація завантаження (рисунок створено самостійно)

Головна сторінка є інтерфейсом керування зображеннями у веб-орієнтованій системі (див. рис. 3.8).

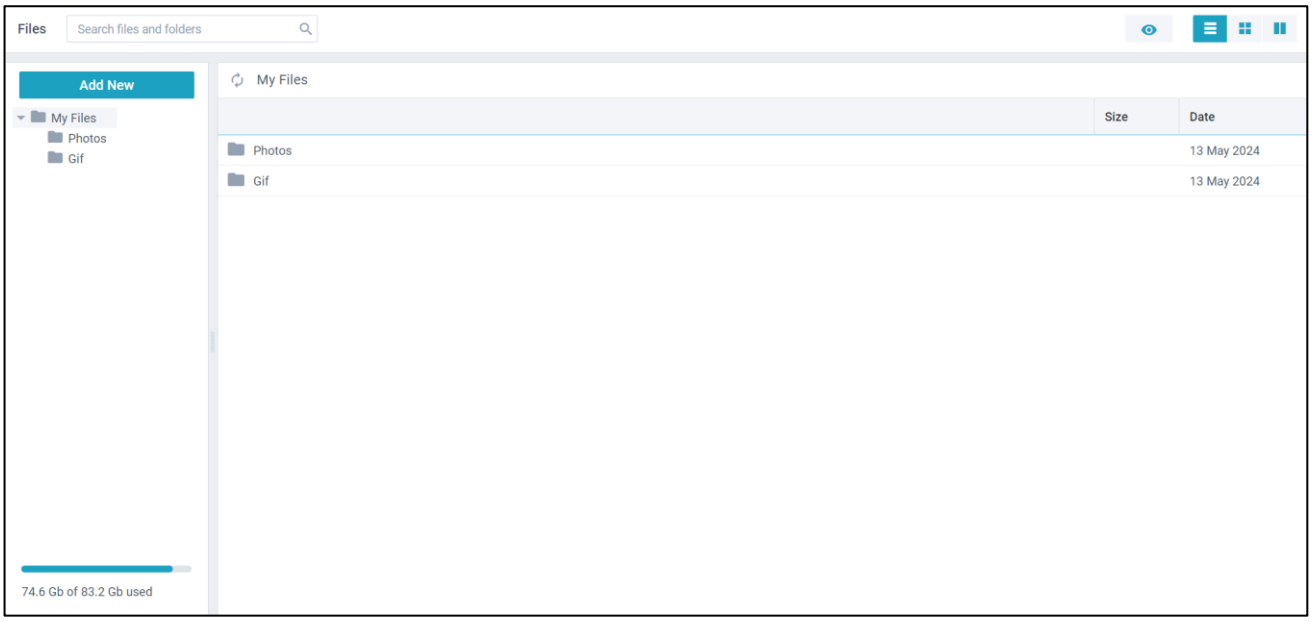


Рисунок 3.8 – Головна сторінка (рисунок створено самостійно)

Передбачено рядок пошуку, що дозволяє користувачам шукати певні файли або папки у своєму сховищі. Ця функція підвищує ефективність роботи користувача, забезпечуючи швидкий доступ до потрібних елементів.

Під заголовком знаходяться опції навігації, представлені у вигляді вкладок або кнопок. Ці опції дозволяють користувачам фільтрувати файли на основі різних категорій або типів, що дозволяє користувачам впорядковувати файли відповідно до їхнього змісту або формату.

Індикація використання сховища надає користувачам інформацію про поточний обсяг сховища та його використання. Це допомагає користувачам ефективно керувати своїми файлами та уникати перевищення лімітів зберігання.

Кнопка "Add New" надає користувачам опції для додавання нових файлів або папок до веб-орієнтованої системи (див. рис. 3.9).

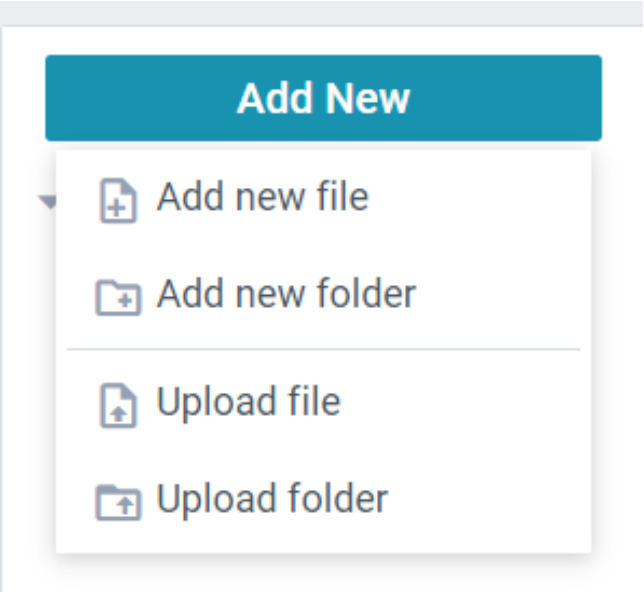


Рисунок 3.9 – Кнопка "Add New" з випадаючим списком (рисунок створено самостійно)

Опція додати новий файл дозволяє користувачам завантажувати окремі файли до системи. При виборі цієї опції користувачам може бути запропоновано вибрати файл з локального пристрою за допомогою діалогового вікна вибору файлу. Після вибору файлу користувачі зазвичай ініціюють процес завантаження, натискаючи кнопку або підтверджуючи свій вибір. Завантажений файл буде перенесено до сховища системи.

За допомогою опції додати нову папку користувачі можуть створювати нові папки у своєму каталозі файлів. Створення папок допомагає користувачам організувати свої файли в логічні групи або категорії, що полегшує навігацію та управління ними. Після вибору цього пункту користувачам може бути запропоновано ввести назву для нової теки. Після введення назви теки користувачі зазвичай підтверджують створення теки, і вона додається до їхнього каталогу файлів.

Опція завантажити файл може надати користувачам альтернативний метод завантаження файлів до системи. На відміну від "Додати новий файл", який передбачає вибір окремих файлів для завантаження, "Завантажити файл" може дозволити користувачам перетягнути кілька файлів або вибрати кілька файлів одночасно для пакетного завантаження. Після вибору файлів для завантаження користувачі зазвичай ініціюють процес завантаження, натискаючи кнопку або підтверджуючи свій вибір. Після цього вибрані файли переносяться до сховища системи.

Подібно до "Завантажити файл", опція завантажити папку дозволяє користувачам завантажувати до системи цілі папки, що містять кілька файлів. Ця функція спрощує процес завантаження, особливо для користувачів, яким потрібно завантажити великі партії файлів, організованих у певні папки. Користувачеві може бути запропоновано вибрати папку на локальному пристрої, і система завантажить всі файли з обраної папки у вказане місце у файловому каталозі користувача.

Розділ "Мої файли" надає користувачам комплексний набір дій для ефективної організації, маніпулювання та підтримки своїх файлів у веб-орієнтованій системі. Користувачі можуть взаємодіяти зі своїми файлами та папками безпосередньо з цього розділу, забезпечуючи зручну навігацію та маніпуляції з контентом (див. рис. 3.10).

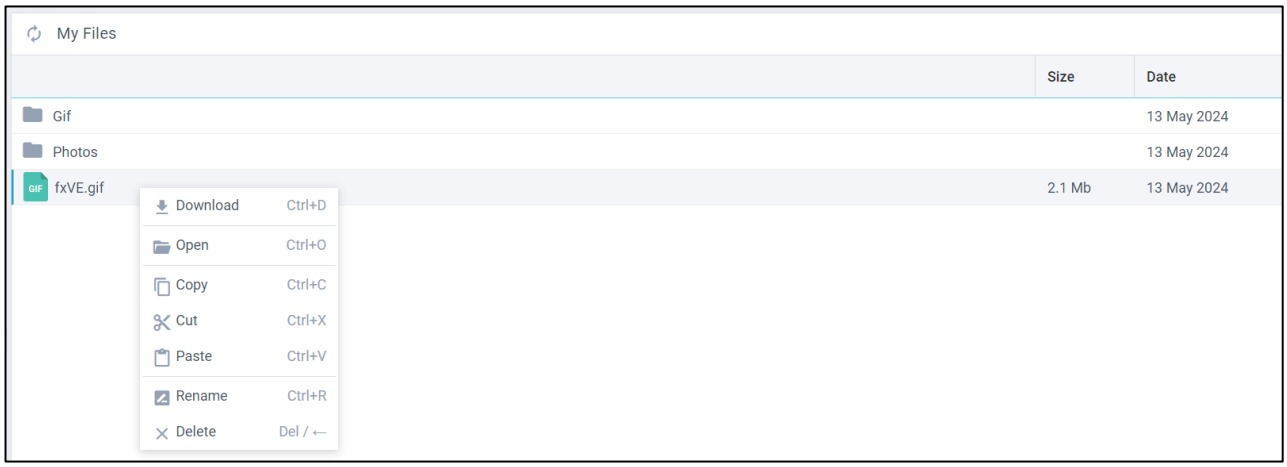


Рисунок 3.10 – Розділ "Мої файли" (рисунок створено самостійно)

Дія "Завантажити" дозволяє користувачам завантажити вибрані файли з системи на свій локальний пристрій. Після вибору одного або декількох файлів і запуску дії завантаження, система зазвичай ініціює процес передачі файлів, пропонуючи користувачеві вибрати місце для збереження завантажених файлів.

Дія "Відкрити" дозволяє користувачам переглядати або отримувати доступ до вибраних файлів в інтерфейсі системи. Залежно від типу файлу та конфігурації системи, відкриття файлу може відображати його вміст безпосередньо в браузері або запускати зовнішню програму, пов’язану з форматом файлу.

За допомогою дії "Копіювати" користувачі можуть копіювати вибрані файли в системі. Копіювання файлів створює дублікати вибраних об’єктів, що дозволяє

користувачам переміщувати або дублювати їх у різні місця у файловому каталозі або в різних теках.

Подібно до дії "Копіювати", дія "Вирізати" дозволяє користувачам переміщувати вибрані файли у системі. Коли файли вирізаються, вони тимчасово видаляються з початкового розташування та зберігаються у системному буфері обміну, готові до вставки у нове місце призначення.

Дія "Вставити" завершує процес переміщення або копіювання, ініційований діями "Вирізати" або "Копіювати". Після вибору папки призначення та запуску дії "Вставити" файли, що зберігаються в буфері обміну, переносяться у вказане місце у файловому каталозі користувача.

За допомогою дії "Перейменувати" можна перейменувати вибрані файли. Перейменування файлів дозволяє користувачам налаштовувати назви файлів, щоб краще відображати їхній вміст або організаційну структуру. При виконанні дії перейменування користувачі зазвичай вводять нову назву для вибраного файла, а система відповідно оновлює метадані файла.

Дія "Видалити" видаляє вибрані файли з системи назавжди.

Отже, ця сторінка пропонує користувачам комплексний інтерфейс для керування своїми файлами і папками у веб-орієнтованій системі. Він надає інтуїтивно зрозумілі можливості навігації, ефективні функції керування файлами та чіткі індикатори використання сховища, що дозволяє користувачам легко впорядковувати свої файли зображень та отримувати до них доступ з мінімальними зусиллями.

Розділ "Попередній перегляд" надає користувачам можливість побачити певний файл зображення разом з детальною інформацією про його атрибути (див. рис. 3.11).

У верхній частині сторінки попереднього перегляду відображається попередній перегляд або ескіз файлу зображення. Таке візуальне представлення дозволяє користувачам швидко визначити вміст зображення.

Інформація про файл:

- тип показує тип файлу, у цьому випадку – файл зображення;
- розмір вказує розмір файлу зображення, зазвичай вимірюється в мегабайтах (Мб) або кілобайтах (Кб);

– дата відображає дату та час створення або останньої зміни файлу зображення.

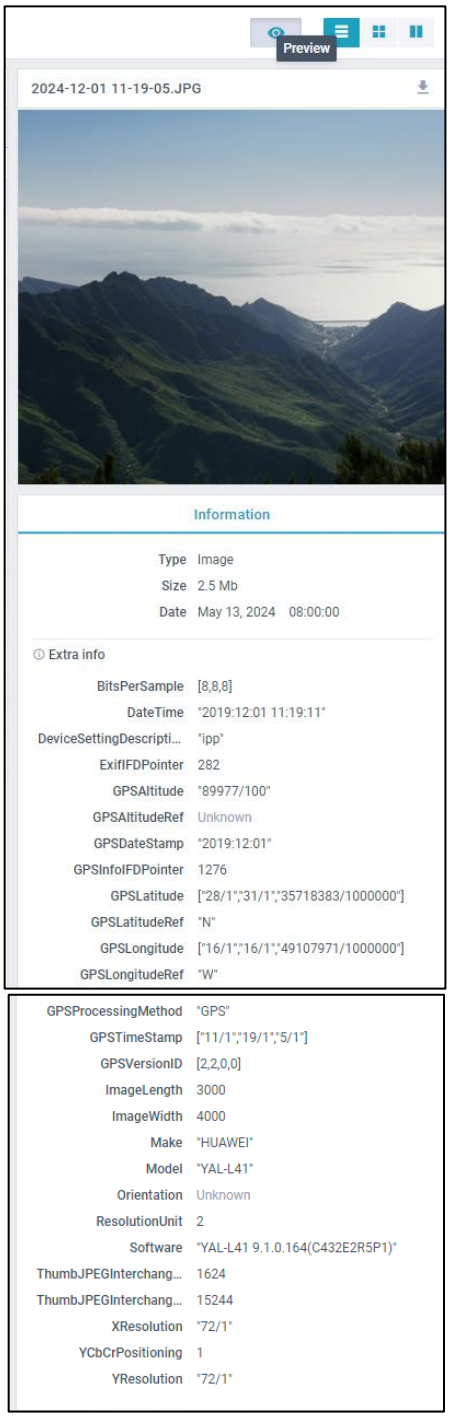


Рисунок 3.11 – Розділ "Попередній перегляд" (рисунок створено самостійно)

Розділ додаткової інформації містить додаткові відомості про файл зображення, часто витягнуті з його метаданих. Нижче наведено деякі поширені атрибути метаданих та їхні значення:

– BitsPerSample вказує кількість бітів на вибірку для кожного компонента пікселя;

- DateTime вказує на дату та час, коли було зроблено зображення;
- Виробник вказує на виробника пристрою, яким було зроблено знімок (наприклад, "HUAWEI");
- модель зазначає модель пристрою, яким було зроблено знімок (наприклад, "YAL-L41");
- одиниця виміру роздільної здатності зазначає одиницю виміру роздільної здатності зображення (наприклад, "пікселі на дюйм");
- програмне забезпечення вказує на програмне забезпечення або програму, за допомогою якої було створено або змінено файл зображення;
- роздільна здатність за вертикаллю, за горизонталлю вказує горизонтальну та вертикальну роздільну здатність зображення відповідно;
- ImageWidth, ImageLength вказує ширину та висоту зображення у пікселях;
- орієнтація вказує орієнтацію зображення (наприклад, книжкову або альбомну);
- GPS-широта, GPS-довгота вказують географічні координати місця, де було зроблено знімок;
- GPSВисота вказує висоту над рівнем моря, на якій було зроблено знімок;
- GPSDateStamp, GPSTimeStamp вказують дату і час, коли було записано дані GPS.

Загалом, розділ "Попередній перегляд " слугує для всебічного огляду вибраного файлу зображення, надаючи користувачам як візуальне представлення зображення, так і детальну інформацію про його атрибути та метадані. Це дозволяє користувачам швидко та ефективно оцінити вміст і властивості зображення.

Розділи "Перегляд файлів" пропонують користувачам різні варіанти відображення для перегляду їх колекції файлів у веб-орієнтованій системі.

У табличному поданні файли представлені у структурованому форматі, як правило, у вигляді рядків і стовпців. Кожен рядок представляє окремий файл, тоді як стовпці відображають різні атрибути або метадані, пов’язані з файлами (див. рис. 3.12).

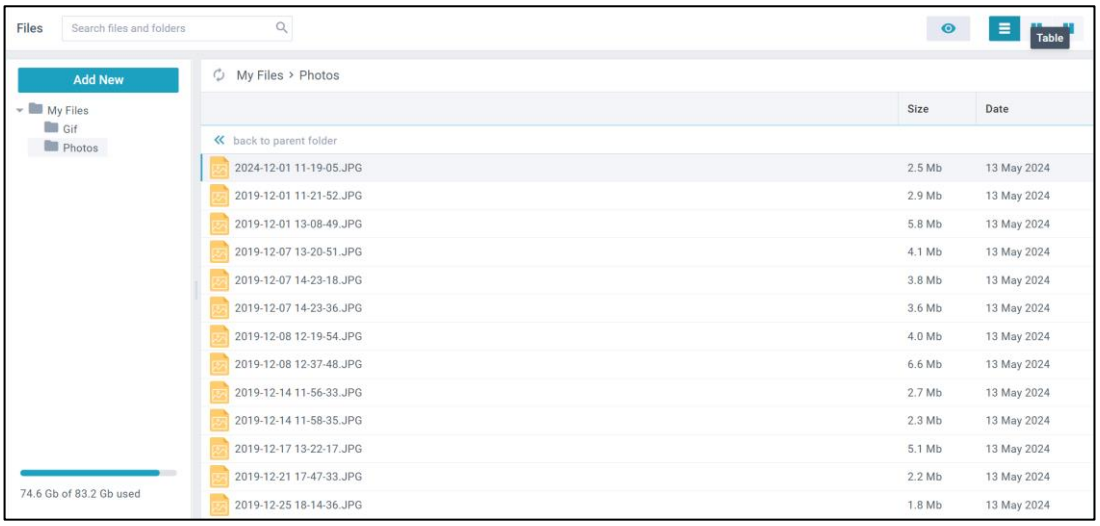


Рисунок 3.12 – Перегляд файлів у табличному поданні (рисунок створено самостійно)

У поданні карток файли відображаються у вигляді окремих карток або плиток, кожна з яких представляє один файл. Картки зазвичай містять мініатюру або зображення попереднього перегляду файлу, а також ключову інформацію, таку як ім’я, тип і розмір файлу (див. рис. 3.13).

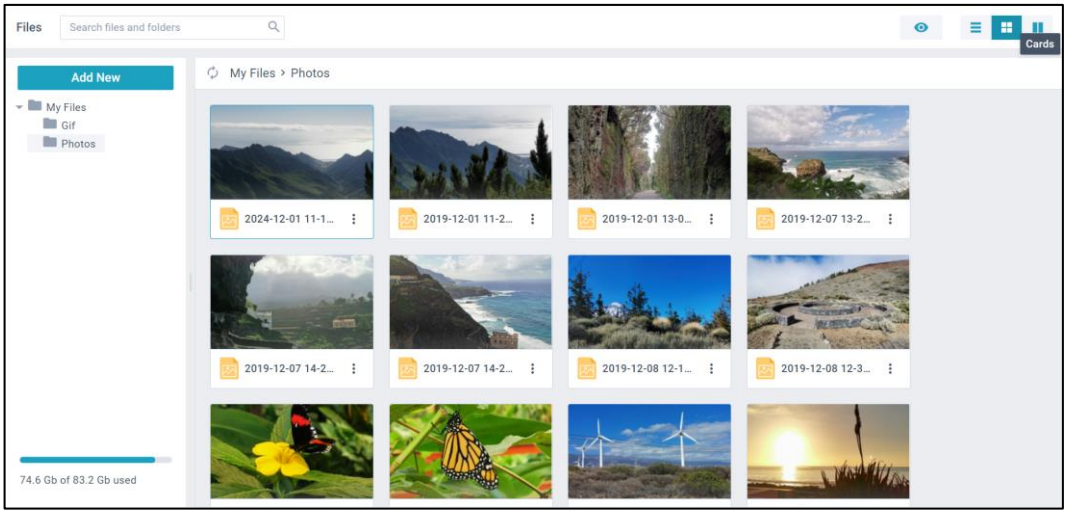


Рисунок 3.13 – Перегляд файлів у поданні карток (рисунок створено самостійно)

Таке подання забезпечує візуально привабливий та інтуїтивно зрозумілий спосіб перегляду файлів, що полегшує ідентифікацію файлів на основі їхнього візуального вмісту. Користувачі можуть швидко прокручувати картки і натискати на окремі картки, щоб переглянути більше деталей або виконати дії.

Загальний перегляд надає користувачам підсумок або огляд всієї їхньої колекції файлів (див. рис. 3.14).

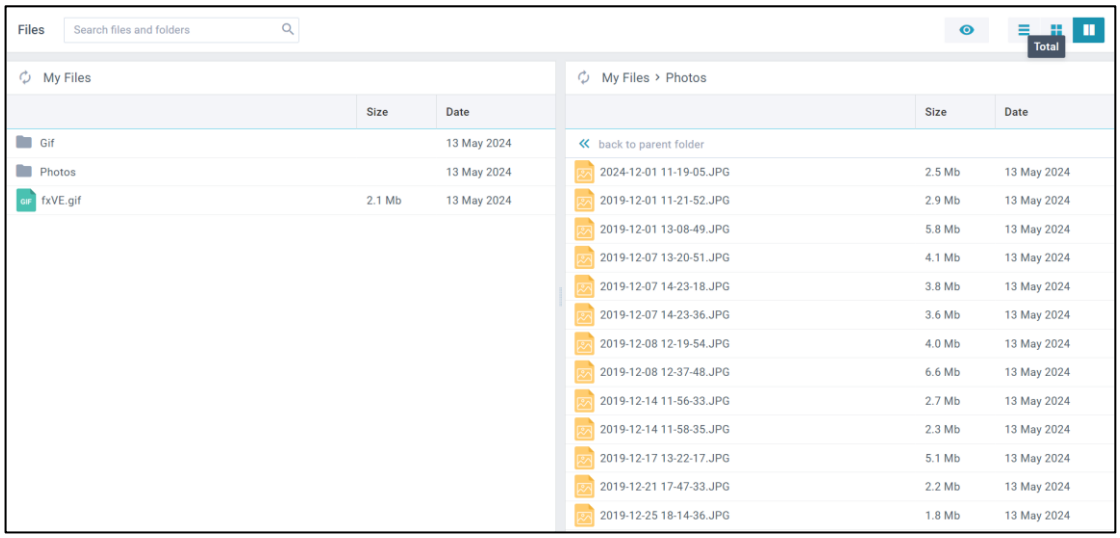


Рисунок 3.14 – Перегляд файлів у загальному перегляді (рисунок створено самостійно)

Користувачі можуть використовувати цей перегляд, щоб отримати уявлення про загальний склад і обсяг своєї колекції файлів, що допоможе їм ефективніше керувати файлами і приймати обґрунтовані рішення щодо використання та організації сховища.

Отже, розділ "Перегляд файлів" пропонує користувачам гнучкість у візуалізації та взаємодії зі своїми файлами у веб-орієнтованій системі. Незалежно від того, чи надають користувачі перевагу структурованим таблицям, візуально насиченим карткам або повному огляду своєї колекції файлів, вони можуть вибрати варіант відображення, який найкраще відповідає їхнім уподобанням і робочому процесу. Це покращує користувацький досвід і забезпечує ефективне управління файлами та навігацію в межах системи.

Функція зберігання дозволяє користувачам безпечно завантажувати та зберігати зображення в системі. Користувачі можуть організовувати зображення в папки або категорії для полегшення управління. Система надає варіанти масштабованих рішень для зберігання, включаючи хмарне та локальне зберігання, залежно від уподобань та вимог користувача. Зображення зберігаються ефективно, враховуючи такі фактори, як розмір файлу, формат і метадані, для оптимізації використання місця в сховищі. Система забезпечує надлишковість даних і механізми резервного копіювання для запобігання їх втрати.

Користувачі можуть шукати зображення за різними критеріями, такими як ім’я файлу, дата, місцезнаходження, теги та метадані. Розширені можливості пошуку дозволяють користувачам фільтрувати зображення на основі певних атрибутів, таких як колір, розмір, роздільна здатність і орієнтація. Система надає результати пошуку в реальному часі з високою точністю та релевантністю до запитів користувача. Для оцінки ефективності пошукового алгоритму використовуються пошукові метрики, зокрема точність, повторюваність та швидкість пошуку.

Зображення зберігаються на основі попередньо визначених критеріїв, таких як дата, місцезнаходження, категорія та визначені користувачем теги. Система автоматично організовує зображення в логічні папки або колекції, щоб полегшити пошук і керування ними. Метрики зберігання використовуються для оцінки ефективності зберігання, включаючи використання простору сховища, коефіцієнт стиснення даних і рівень надмірності даних.

Система включає алгоритми аналізу зображень для таких завдань, як виявлення об’єктів, класифікація зображень і сегментація зображень. Користувачі можуть аналізувати зображення для вилучення значущої інформації, наприклад, ідентифікації об’єктів, виявлення закономірностей або вимірювання атрибутів зображення. Метрики аналізу використовуються для оцінки точності та продуктивності алгоритмів аналізу зображень, включаючи точність, розпізнавання, оцінку F1 і час обробки.

Система підтримує класифікацію зображень на основі моделей машинного навчання, навчених на мічених наборах даних. Користувачі можуть класифікувати зображення за попередньо визначеними категоріями або мітками, або створювати власні мітки на основі певних критеріїв. Метрики класифікації використовуються для оцінки точності та продуктивності моделей класифікації, включаючи точність класифікації, точність моделі, запам’ятовування та час навчання.

Таким чином, надаючи пріоритет ясності, інтуїтивності та функціональності різних розділів, сторінок, кнопок і функцій, система підвищує залученість, продуктивність та задоволеність користувачів, зрештою, досягаючи мети ефективної обробки та аналізу зображень в умовах великих даних.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Гібридний механізм управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних передбачає інтеграцію різних технологій та методологій для роботи з великими обсягами даних зображень. Ця система поєднує хмарне зберігання, локальну обробку та передові алгоритми аналізу зображень, щоб забезпечити надійне рішення. Нижче наведено детальний план того, як гібридний механізм управління зображеннями реалізовано у front-end частині з використанням Python як для зовнішньої взаємодії, так і для внутрішньої підтримки.

Ключові функції для реалізації завантаження та зберігання зображень:

- підтримка завантаження зображень за допомогою перетягування або вибору файлу;
- зберігання зображення локально для швидкого доступу та в хмарному сховищі для масштабування.

Ключові функції для реалізації організації зображень:

- впорядкування зображення за папками або категоріями;
- керування тегами та метаданими.

Ключові функції для реалізації пошуку зображень:

- пошук зображення на основі метаданих, тегів і вмісту;
- впровадження розширених алгоритмів пошуку за допомогою розпізнавання зображень.

Ключові функції для реалізації обробки та аналізу зображень:

- застосування алгоритму обробки зображень (наприклад, зміна розміру, фільтрація);
- аналіз зображення для виявлення об’єктів, класифікації та інших метрик.

Аутентифікація та авторизація користувачів:

- забезпечення безпечного доступу за допомогою аутентифікації користувачів;
- керування доступом на основі ролей для різних дозволів користувачів.

Спочатку були встановлені необхідні пакунки Python. Буду використовувати Flask для веб-сервера, SQLAlchemy для взаємодії з базами даних та opencv-python для обробки зображень.

Flask – це мікро веб-фреймворк для Python, що дозволяє легко масштабувати його та є достатньо гнучким для підтримки як простих, так і складних додатків. Flask надає основні інструменти та функції, необхідні для створення веб-додатків, такі як маршрутизація, обробка запитів та шаблонування [12].

Ключові особливості:

- просте ядро та модульний дизайн Flask дозволяють швидко розпочати роботу, а розширення дозволяють додавати функціональність за потреби;
- Flask підтримує маршрутизацію URL-адрес, що дозволяє створювати чисті та читабельні URL-адреси для різних частин веб-додатку;
- Flask використовує Jinja2 як движок шаблонів, що дозволяє динамічно генерувати HTML з успадкуванням шаблонів та іншими можливостями;
- Flask надає інструменти для обробки HTTP запитів та відповідей, включаючи підтримку GET, POST, PUT, DELETE та інших методів;
- Flask має багату екосистему розширень для додавання таких функцій, як обробка форм, інтеграція з базами даних та автентифікація.

SQLAlchemy – це інструментарій SQL та бібліотека об’єктно-реляційного відображення (ORM) для Python. Вона надає повний набір відомих шаблонів збереження даних корпоративного рівня, призначених для ефективного та високопродуктивного доступу до баз даних [13]. Функція ORM в SQLAlchemy дозволяє розробникам взаємодіяти з базами даних за допомогою об’єктів Python, роблячи маніпуляції з базами даних більш інтуїтивно зрозумілими та менш схильними до помилок [14].

Ключові особливості:

- ORM SQLAlchemy дозволяє перетворювати таблиці бази даних в класи Python, а рядки в екземпляри цих класів. Це дозволяє безперешкодно взаємодіяти з базою даних за допомогою коду на Python;

- для тих, хто віддає перевагу роботі з SQL, SQLAlchemy надає потужну мову виразів SQL для побудови та програмного виконання SQL-запитів;
- SQLAlchemy підтримує широкий спектр серверів баз даних, включаючи SQLite, PostgreSQL, MySQL, Oracle та інші, забезпечуючи узгоджений інтерфейс між різними системами баз даних;
- SQLAlchemy керує сеансами баз даних, що полегшує управління транзакціями та підтримує узгодженість операцій з базами даних;
- SQLAlchemy може автоматично генерувати схеми баз даних на основі визначень класів Python, спрощуючи процес налаштування та модифікації баз даних.

OpenCV (Open Source Computer Vision Library) – це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання з відкритим вихідним кодом [15]. Пакет opencv-python – це прив'язка Python до OpenCV, що дозволяє розробникам Python використовувати потужні можливості OpenCV з обробки зображень та комп'ютерного зору [16].

Ключові особливості:

- OpenCV надає широкий спектр функцій для базових і розширених завдань обробки зображень, включаючи фільтрацію, виявлення країв, перетворення зображень та багато іншого;
- OpenCV включає алгоритми для виявлення об'єктів, розпізнавання облич, відстеження руху, 3D-реконструкції та інших завдань комп'ютерного зору;
- OpenCV постачається з реалізаціями різних алгоритмів машинного навчання, які можна використовувати для таких завдань, як класифікація зображень, кластеризація та виділення ознак;
- OpenCV оптимізовано для обробки зображень у реальному часі, що робить його придатним для додатків, які потребують швидкої та ефективної обробки;

– OpenCV підтримує різні платформи, включаючи Windows, Linux, macOS, iOS і Android, що дозволяє розробникам створювати крос-платформні додатки.

Flask відповідає за веб-інтерфейс, маршрутизацію запитів і шаблони рендерингу, забезпечуючи зручну платформу для завантаження та взаємодії з зображеннями. SQLAlchemy керує операціями з базою даних, зберігаючи метадані та інформацію про зображення, а також забезпечує ефективні запити та маніпуляції з даними. OpenCV виконує завдання обробки та аналізу зображень, дозволяючи системі обробляти великі обсяги даних і виконувати різні операції, такі як зміна розміру, фільтрація та виявлення об’єктів. Ця комбінація використовує сильні сторони кожного пакета для створення надійної, масштабованої та ефективної системи для управління та аналізу великих обсягів зображень.

Починаю створювати Flask-додаток для роботи з веб-інтерфейсом та кінцевими точками API.

Імпортую необхідні бібліотеки та модулі для створення веб-додатку, взаємодії з базою даних та обробки зображень:

```
from flask import Flask, request, jsonify, render_template
from flask_sqlalchemy import SQLAlchemy
import cv2
import os
import boto3
```

Далі ініціалізується додаток Flask для роботи з веб-інтерфейсом і кінцевими точками API. Додаток налаштовується на використання SQLite як бекенд бази даних, встановлюючи конфігурацію SQLALCHEMY_DATABASE_URI на ‘sqlite:///images.db’. Потім SQLAlchemy ініціалізується з цією конфігурацією, щоб уможливити взаємодію з базами даних у контексті Flask:

```
app = Flask(__name__)
app.config[‘SQLALCHEMY_DATABASE_URI’] = ‘sqlite:///images.db’
db = SQLAlchemy(app)
```

За допомогою SQLAlchemy визначається модель бази даних для зберігання метаданих зображень:

```
class Image(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(100), unique=True, nullable=False)
    filepath = db.Column(db.String(200), nullable=False)
    tags = db.Column(db.String(200))
    metadata = db.Column(db.String(500))
db.create_all()
```

Ця модель, представлена класом Image, включає декілька стовпців:

- id – цілочисельний первинний ключ, який унікально ідентифікує кожен запис зображення;
- filename – рядковий стовпець, який зберігає ім'я файлу зображення. Воно є унікальним і не може бути нульовим;
- filepath – рядковий стовпчик, що зберігає шлях до файлу зображення. Він також не може бути нульовим;
- tags – стовпчик, у якому зберігаються теги зображення, розділені комами.
- metadata – рядковий стовпець, що містить додаткові метадані про зображення, такі як розміри або формат.

Потім викликається функція db.create_all() для створення таблиць бази даних на основі визначень моделі. Це гарантує, що схема бази даних буде правильно ініціалізована, створюючи необхідні таблиці для зберігання метаданих зображень.

Код нижче дозволяє користувачам завантажувати зображення за маршрутом /upload. Коли користувач надсилає POST-запит з прикріпленим файлом зображення, код перевіряє, чи існує цей файл. Якщо він існує, код зберігає файл локально та завантажує його у Amazon S3. Потім він записує інформацію про завантажене зображення в базу даних. У кінці надсилає повідомлення, яке підтверджує, що завантаження було успішним.

```
s3 = boto3.client('s3')
BUCKET_NAME = 'bucket-name'
@app.route('/upload', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400
    filepath = os.path.join('uploads', file.filename)
    file.save(filepath)
```

```
        s3.upload_file(filepath, BUCKET_NAME, file.filename)
        new_image = Image(filename=file.filename, filepath=filepath)
        db.session.add(new_image)
        db.session.commit()
        return jsonify({'message': 'File uploaded successfully'}), 201
```

Маршрути нижче обробляють список зображень (/images) та оновлюють метадані зображень (/image/<int:id>). Маршрут /images витягує всі зображення з бази даних і повертає їх у вигляді JSON-об'єктів, що містять ідентифікатор, ім'я файлу, теги і метадані. Маршрут /image/<int:id> оновлює метадані конкретного зображення, ідентифікованого за його id. Якщо зображення існує, він витягує оновлені метадані з JSON-даних запиту, оновлює метадані зображення в базі даних і повертає повідомлення з підтвердженням. Якщо зображення не існує, він повертає помилку 404:

```
    @app.route('/images', methods=['GET'])
    def list_images():
        images = Image.query.all()
        return jsonify([{'id': img.id, 'filename': img.filename, 'tags':
img.tags, 'metadata': img.metadata} for img in images])
    @app.route('/image/<int:id>', methods=['PUT'])
    def update_image(id):
        image = Image.query.get(id)
        if not image:
            return jsonify({'error': 'Image not found'}), 404
        data = request.get_json()
        image.tags = data.get('tags', image.tags)
        image.metadata = data.get('metadata', image.metadata)
        db.session.commit()
        return jsonify({'message': 'Image updated successfully'})
```

Код нижче визначає маршрут /search, який обробляє GET-запити. Він витягує пошуковий запит з параметрів запиту. Потім він звертається до бази даних, щоб знайти зображення, теги або метадані яких містять пошуковий запит. Повертає знайдені зображення у вигляді JSON-об'єктів, що містять їх ідентифікатор, ім'я файлу, теги та метадані:

```
    @app.route('/search', methods=['GET'])
    def search_images():
        query = request.args.get('query')
        images = Image.query.filter((Image.tags.contains(query)) |
(Image.metadata.contains(query))).all()
        return jsonify([{'id': img.id, 'filename': img.filename, 'tags':
img.tags, 'metadata': img.metadata} for img in images])
```

Код нижче визначає маршрут /process/<int:id>, який обробляє GET-запити для обробки зображень. Він отримує дані зображення з бази даних на основі наданого ідентифікатора. Потім він читає файл зображення за допомогою OpenCV (cv2), виконує обробку зображення (в даному випадку перетворення зображення в градації сірого) і зберігає оброблене зображення в новий файл. Повертає відповідь у форматі JSON, що підтверджує успішну обробку зображення і містить шлях до файлу з обробленим зображенням:

```
@app.route('/process/<int:id>', methods=['GET'])
def process_image(id):
    image = Image.query.get(id)
    if not image:
        return jsonify({'error': 'Image not found'}), 404
    img = cv2.imread(image.filepath)
    processed_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    processed_filepath = os.path.join('processed', image.filename)
    cv2.imwrite(processed_filepath, processed_img)
    return jsonify({'message': 'Image processed successfully',
'processed_filepath': processed_filepath})
```

Код нижче реалізує автентифікацію користувачів і контроль доступу на основі ролей за допомогою Flask-Login. У ньому визначено клас User, який представляє користувачів з такими атрибутами, як ідентифікатор, ім'я користувача, пароль і роль. Для обробки автентифікації користувачів ініціалізується менеджер входів:

```
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
login_manager = LoginManager()
login_manager.init_app(app)
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(50))
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
```

Маршрут /login перевіряє облікові дані користувача і реєструє його, якщо вони дійсні. Після успішного входу, відповідь у форматі JSON підтверджує статус входу:

```
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    user = User.query.filter_by(username=data['username']).first()
    if user and user.password == data['password']:
        login_user(user)
        return jsonify({'message': 'Logged in successfully'}), 200
    return jsonify({'error': 'Invalid credentials'}), 401
```

Маршрут /logout виводить поточного користувача з системи, а маршрут /protected вимагає автентифікації для доступу. Якщо користувач автентифікований, він повертає відповідь у форматі JSON із зазначенням доступу до захищеного маршруту разом з іменем поточного користувача. Якщо користувач не автентифікований, програма повертає помилку 401:

```
@app.route('/logout', methods=['POST'])
@login_required
def logout():
    logout_user()
    return jsonify({'message': 'Logged out successfully'}), 200
@app.route('/protected', methods=['GET'])
@login_required
def protected():
    return jsonify({'message': 'This is a protected route', 'user':
current_user.username})
```

Отже, вище наведено план реалізації, який описує кроки зі створення гібридного механізму управління зображеннями з використанням Python і Flask для інтерфейсної частини системи. Система включає в себе завантаження та зберігання зображень, організацію, пошук, обробку та аналіз. Вона також включає автентифікацію користувачів і контроль доступу на основі ролей для забезпечення безпеки і конфіденційності. Описані функції використовують як локальне, так і хмарне сховище для ефективної обробки великих обсягів даних зображень, забезпечуючи масштабоване та зручне рішення для управління та аналізу зображень в контексті великих даних.

Для ефективної обробки та аналізу зображень у веб-орієнтованій системі гібридного механізму управління зображеннями буду використовувати OpenCV. OpenCV – це популярна бібліотека для задач комп'ютерного зору, яка надає численні функції для обробки та аналізу зображень [17]. Буду використовувати

OpenCV для виконання різних операцій, таких як зміна розміру, обрізання, фільтрація, виявлення країв та виділення елементів.

Приклади операцій:

– налаштування розмірів зображень відповідно до конкретних вимог, наприклад, стандартизація розмірів для аналізу або відображення;
– виявлення та виділення країв або меж зображень, що може бути корисним для виявлення об’єктів або сегментації;
– вилучення значущих характеристик із зображень, таких як ключові точки або дескриптори, для представлення та аналізу їхнього вмісту.

Критерії/показники для оцінювання:

– оцінка точності та візуальної якості оброблених зображень має вирішальне значення. Для кількісної оцінки якості зображення можна використовувати такі показники, як індекс структурної подібності (SSI), пікове відношення сигнал/шум (PSNR) або середньоквадратична помилка (MSE);
– для задач, пов’язаних з виділенням ознак, точність виділених ознак у порівнянні з істинними даними є дуже важливою. Такі показники, як точність, відгук (Recall) або F1-Score, можуть оцінити продуктивність алгоритмів виділення ознак;
– у випадках, коли йдеться про розпізнавання або виявлення об’єктів, такі метрики, як Intersection over Union (IoU) або Average Precision (AP), можуть виміряти точність і надійність алгоритмів розпізнавання об’єктів.

Нижче наведено реалізацію на Python, яка демонструє базову обробку та аналіз зображень за допомогою OpenCV:

```
import cv2
img = cv2.imread('image.jpg')
resized_img = cv2.resize(img, (300, 300))
edges = cv2.Canny(resized_img, 100, 200)
psnr = cv2.PSNR(img, resized_img)
print('PSNR:', psnr)
```

Завантажуємо зображення та змінюємо його розмір до потрібного. Виконуємо виявлення країв за допомогою Canny edge detection – популярний алгоритм, який використовується в комп'ютерному зорі для виявлення країв на зображеннях. Обчислюємо PSNR (пікове відношення сигнал/шум) між оригінальним і зміненим зображеннями як міру якості зображення.

Інтеграція OpenCV у веб-орієнтовану систему дозволяє ефективно обробляти та аналізувати зображення. Визначивши конкретні критерії та метрики, можемо точно оцінити якість та продуктивність реалізованих алгоритмів. Такий підхід гарантує, що система відповідає вимогам для ефективної обробки та аналізу зображень в контексті великих даних.

Щоб реалізувати функціонал для завантаження зображень та зберігання їх разом з відповідними метаданими, виконую наступні кроки:

- надано можливість користувачам завантажувати зображення з їхніх локальних комп'ютерів на сервер за допомогою HTML-форми з полем для введення файлу;
- завантажені зображення зберігаються у файловій системі сервера або в хмарному сховищі Amazon S3, переконавшись, що кожне зображення зберігається з унікальним ім'ям, щоб уникнути конфліктів;
- разом із зображенням зберігаються в базі даних метадані, такі як ім'я файлу, мітка часу завантаження, розмір, а також будь-які визначені користувачем теги або описи, що дозволить ефективно шукати та впорядковувати зображення;
- впроваджено різні метрики, щоб розширити функціональність системи управління зображеннями;
- надано можливість користувачам шукати зображення на основі метаданих, таких як назва файлу, теги або описи. Реалізовано функцію пошуку, яка запитує базу даних для пошуку відповідних зображень;
- зображення зберігаються в масштабований та ефективний спосіб, щоб обробляти великі обсяги завантажень. Використовується хмарне сховище для кращої масштабованості та надійності;

– впроваджено алгоритми аналізу зображень для вилучення ознак або виконання таких завдань, як виявлення об’єктів або сегментація зображень. Використані такі бібліотеки, як OpenCV або TensorFlow для обробки та аналізу зображень;
– використано моделі машинного навчання для класифікації зображень за попередньо визначеними категоріями або тегами. Модель навчено на основі набору даних із позначеними зображеннями, також вона використовується для автоматичної класифікації нових зображень.

Нижче наведена реалізація функції завантаження та зберігання зображень:

```
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
@app.route('/upload', methods=['POST'])
def upload_image(): # Перевіряє, чи є в POST-запиті файлова частина
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
    # Якщо користувач не вибрав файл, браузер також
    # відправить порожню частину без імені файлу
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400
    if file:
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename))
        # Зберігає метадані в базі даних
        image_data = {
            'filename': filename,
            'upload_time': datetime.datetime.now(),
            'size':
os.path.getsize(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        # Зберігає дані зображення в базу даних
        return jsonify({'message': 'File uploaded successfully',
'filename': filename}), 201        }
```

Цей код налаштовує додаток для обробки завантажень зображень. Він визначає маршрут /upload для обробки POST-запитів. Коли файл завантажується, він перевіряє, чи існує частина файлу, зберігає файл у папку завантаження на сервері та записує метадані (ім’я файлу, час завантаження, розмір) у словник. Потім він повертає відповідь у форматі JSON, що свідчить про успішність завантаження разом з назвою файлу.

Отже, механізм управління гібридними зображеннями реалізовано за допомогою веб-орієнтованої системи, яка інтегрує різні технології та алгоритми для ефективної обробки та аналізу зображень у великих обсягах даних. Для обробки та аналізу зображень використовуються такі алгоритми, як виявлення країв Кенні, для вилучення важливих характеристик із зображень. Ці алгоритми реалізовані за допомогою бібліотеки OpenCV на мові Python, що дозволяє вирішувати такі завдання, як виявлення країв, розпізнавання об’єктів та покращення зображень. Були визначені критерії та метрики для обробки та аналізу, включаючи точність, швидкість та ефективність використання ресурсів. Наприклад, при виявленні країв метрики включають кількість виявлених країв, помилкових спрацьовувань та час обробки. Щодо обсягу завантаження та зберігання, система ефективно обробляє великі обсяги даних зображень з оптимізованою швидкістю завантаження та вивантаження, а також ефективно зберігає зображення, можливо, з використанням хмарних сховищ, як Amazon S3. Показники швидкості завантаження та вивантаження включають швидкість передачі та затримку. Для пошуку, зберігання та класифікації зображень система використовує системи управління базами даних, як SQLAlchemy, для ефективного зберігання та пошуку метаданих зображень. Критеріями для пошуку та класифікації зображень є теги, метадані та особливості зображень, вилучені під час аналізу. Таким чином, гібридний механізм управління зображеннями був успішно реалізований, використовуючи комбінацію веб-технологій, алгоритмів обробки зображень та систем управління базами даних для ефективної обробки, аналізу, зберігання та пошуку зображень, дотримуючись визначених критеріїв і метрик для забезпечення ефективності та продуктивності.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Юніт – або модульне тестування має вирішальне значення для забезпечення правильної роботи окремих компонентів інтерфейсної частини веб-орієнтованої системи гібридного механізму управління зображеннями. Кожна функція, описана у вимогах, буде протестована за допомогою спеціальних тестових кейсів для перевірки як функціональних, так і нефункціональних вимог. Такий детальний підхід гарантує, що кожен аспект користувацького інтерфейсу (UI) та користувацького досвіду (UX) працює за призначенням, забезпечуючи надійну та надійну систему для управління, ефективної обробки та аналізу в умовах великих даних.

Модульне тестування функціоналу авторизації та реєстрації включає кілька важливих аспектів, щоб гарантувати, що система працює належним чином. Першим кроком є тестування процесу реєстрації користувача. Цей тест, названий user_registration_test, імітує заповнення користувачем реєстраційної форми з валідними даними, такими як ім’я, електронна пошта та пароль. Тест перевірить, чи правильно система обробляє цю інформацію, чи надійно зберігає її та чи надсилає повідомлення з підтвердженням на електронну пошту користувача. Повідомлення з підтвердженням є дуже важливим, оскільки воно гарантує, що користувач надав дійсну адресу електронної пошти та з ним можна буде зв’язатися в разі потреби. Також user_registration_test перевіряє, чи правильно форма реєстрації обробляє невірні дані. Наприклад, він повинен тестувати сценарії, коли користувач вказує адресу електронної пошти, яка вже використовується, пароль, який не відповідає вимогам безпеки, або залишає обов’язкові поля порожніми. Система повинна реагувати відповідними повідомленнями про помилки, які допоможуть користувачеві виправити введені дані.

Другий тест, user_authentication_test, фокусується на процесі входу в систему. Цей тест перевіряє, чи може зареєстрований користувач увійти в систему, використовуючи свою електронну пошту та пароль. Система повинна аутентифікувати користувача і надати доступ до захищених областей програми. Цей тест також перевірить реакцію системи на неправильні спроби входу,

гарантуючи, що користувачі отримають чіткий зворотній зв’язок, якщо вони введуть неправильні облікові дані. Це включає в себе тестування проти атак SQL-ін’єкцій та інших поширених загроз безпеки, щоб забезпечити безпеку процесу входу в систему.

Щоб забезпечити конфіденційність та безпеку під час реєстрації, тест registration_security_test перевірить, що система збирає тільки необхідну інформацію від користувача та зберігає її безпечно. Цей тест перевірить шифрування конфіденційних даних, таких як паролі, переконавшись, що вони хешуються і соляться перед тим, як зберігатися в базі даних. Він також перевірить, чи відповідає процес реєстрації правилам конфіденційності, таким як GDPR, гарантуючи, що дані користувачів обробляються відповідально.

Результати модульного тестування функціоналу авторизації та реєстрації зображено на рисунку 5.1.

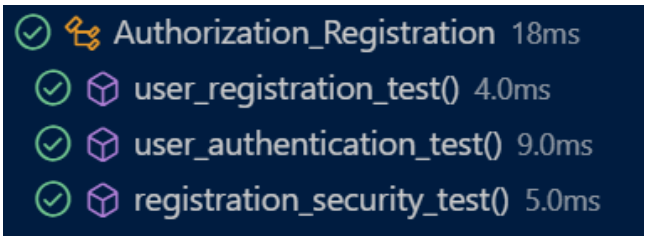


Рисунок 5.1 – Результати модульного тестування функціоналу авторизації та реєстрації (рисунок створено самостійно)

Успішне виконання тестів модулів для функцій авторизації та реєстрації підтверджує, що система відповідає запланованому дизайну та вимогам безпеки. Модулі авторизації та реєстрації функціонують належним чином, забезпечуючи безпечний, надійний та зручний досвід як для нових користувачів, так і для зареєстрованих.

Функціонал пошуку файлів буде протестовано, щоб переконатися, що користувачі можуть ефективно знаходити свої файли зображень. Тест file_search_by_name_test імітуватиме пошук користувачем зображення за назвою, перевіряючи, що функція пошуку не чутлива до регістру, обробляє часткові збіги і

швидко повертає відповідні результати. Для забезпечення очікуваного часу відгуку будуть додані тести продуктивності.

Тест file_search_by_metadata_test охоплюватиме пошук за тегами, датами завантаження та іншими метаданими, щоб переконатися, що система правильно індексує і знаходить файли на основі цих критеріїв. Цей тест також підтвердить, що інтерфейс пошуку є інтуїтивно зрозумілим і швидко реагує на запити користувачів, покращуючи їхню роботу.

Результати модульного тестування функціоналу пошуку файлів зображено на рисунку 5.2.

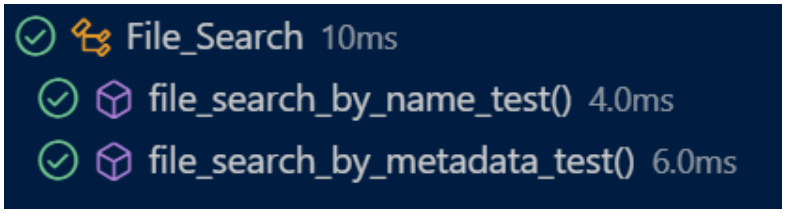


Рисунок 5.2 – Результати модульного тестування функціоналу пошуку файлів (рисунок створено самостійно)

Тестування функції пошуку файлів пройшло успішно, підтвердивши здатність системи ефективно знаходити файли зображень за різними критеріями. Успішне виконання цих тестів демонструє, що функція пошуку файлів відповідає функціональним вимогам, дозволяючи користувачам швидко та точно знаходити свої файли зображень у веб-орієнтованій системі.

Процес завантаження файлів є основним функціоналом, який потребує ретельного тестування. Тест file_upload_test перевірить, чи можуть користувачі завантажувати різні формати зображень, включаючи JPEG, PNG і GIF. Цей тест імітує процес завантаження, перевіряючи, чи правильно система обробляє кожен тип файлів і зберігає їх у визначеному місці зберігання.

Індикатори прогресу мають вирішальне значення для забезпечення зворотного зв’язку з користувачами під час завантаження файлів. Тест upload_progress_bar_test гарантує, що індикатор виконання точно відображає стан завантаження, динамічно оновлюючись під час передачі файлу. Цей тест також

перевіряє граничні випадки, такі як переривання мережі, і гарантує, що система може відновити завантаження, якщо це необхідно.

Безпека під час завантаження файлів є ще одним важливим аспектом. Тест file_upload_security_test перевірить, що система сканує завантажені файли на наявність потенційних загроз, таких як шкідливе програмне забезпечення, і обробляє їх належним чином. Цей тест також перевірить, чи процес завантаження зашифрований, забезпечуючи цілісність і конфіденційність даних, що передаються.

Результати модульного тестування функціоналу завантаження файлів зображено на рисунку 5.3.

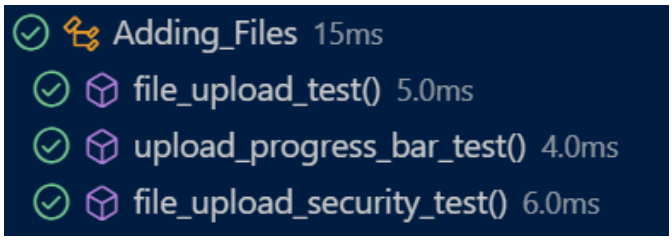


Рисунок 5.3 – Результати модульного тестування функціоналу завантаження файлів (рисунок створено самостійно)

Успішне виконання цих тестів підтверджує надійність функції завантаження файлів у системі. Користувачі можуть безпечно завантажувати різні формати зображень, отримуючи точний зворотній зв’язок про хід роботи, що забезпечує безперебійний процес завантаження файлів до веб-орієнтованої системи.

Тестування функціональності перегляду файлів передбачає забезпечення можливості для користувачів переглядати файли зображень як у плитковому, так і у списковому вигляді. Тести tile_view_test і list_view_test перевірять, чи відображає система мініатюри або прев’ю для кожного файлу, забезпечуючи візуальне представлення вмісту. Ці тести перевірять, чи правильно відображаються подання і чи елементи керування навігацією дозволяють користувачам ефективно переглядати великі колекції.

Тест view_file_information_test перевірить, чи можуть користувачі отримати доступ до детальної інформації про кожен файл зображення, включно з

метаданими, такими як розмір файлу, роздільна здатність, дата створення та теги. Цей тест переконається, що інформація представлена чітко і зрозуміло, що полегшує користувачам її пошук і посилання на неї.

Для завантаження файлів тест file_download_test перевірить, чи можуть користувачі завантажувати файли зображень на свої локальні пристрої. Цей тест перевірить, що опції завантаження є легкодоступними і що користувачі можуть вибрати кілька файлів для пакетного завантаження, якщо це необхідно. Тест також перевірить цілісність завантажених файлів, щоб переконатися, що вони відповідають оригінальним завантаженим файлам.

Тест open_file_test перевірить, чи можуть користувачі відкривати файли зображень в інтерфейсі системи для перегляду та взаємодії. Цей тест перевірить, що різні режими перегляду, такі як повноекранний, масштабування та панорамування, підтримуються та функціонують належним чином, дозволяючи детально вивчити вміст зображення.

Функціональність копіювання, вирізання та вставки файлів між директоріями буде перевірено тестом copy_cut_paste_files_test. Цей тест перевірить, чи ці операції є інтуїтивно зрозумілими і підтримують вибір як одного, так і декількох файлів. Він перевірить, чи система відповідним чином оновлює розташування файлів і коректно обробляє будь-які потенційні конфлікти.

Тести edit_folder_name_test і delete_folder_test переконаються, що користувачі можуть перейменовувати і видаляти теки, якщо це необхідно для впорядкування файлів зображень. Ці тести перевірять, чи система запитує підтвердження перед виконанням цих дій, щоб запобігти випадковій втраті даних.

Аналогічно, тести rename_file_test і delete_file_test перевірять, чи можуть користувачі перейменовувати і видаляти файли з системи. Ці тести перевірять, чи система містить діалогові вікна підтвердження для запобігання ненавмисним змінам, а також чи операції є простими та зручними для користувача.

Тест display_memory_usage_test переконається, що система надає користувачам точну інформацію про обсяг пам’яті, використаний для зберігання файлів зображень. Цей тест перевірить, що статистика використання пам’яті

відображається у зрозумілому форматі і динамічно оновлюється при додаванні або видаленні файлів.

Результати модульного тестування функціоналу вище зображено на рисунку 5.4.

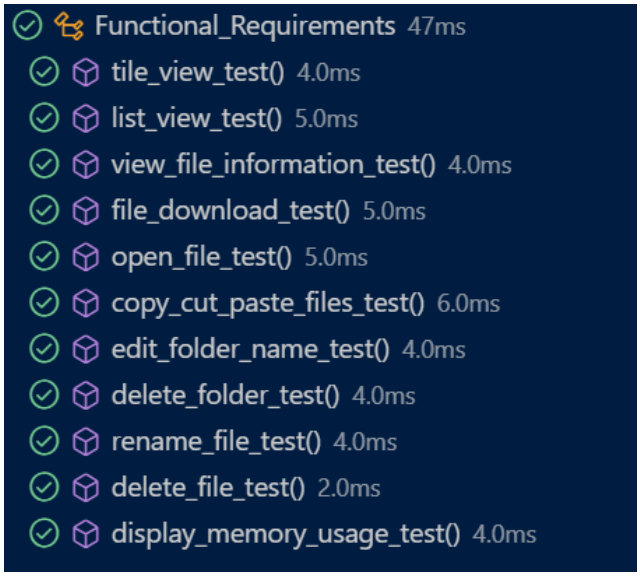


Рисунок 5.4 – Результати модульного тестування функціоналу (рисунок створено самостійно)

Успішне виконання модульних тестів для функції перегляду файлів і пов’язаних з нею функцій демонструє стійкість та надійність системи в роботі з графічними файлами. Успішне виконання цих модульних тестів підкреслює здатність системи ефективно керувати файлами зображень, забезпечуючи при цьому зручність та багатофункціональність.

Тестування гібридного механізму управління зображеннями передбачає перевірку того, що система підтримує одночасне завантаження великого обсягу зображень, витримує високі трафікові навантаження та ефективно обробляє завантаження та вивантаження з мінімальною затримкою. Тест large_volume_upload_test імітує умови високого трафіку, забезпечуючи ефективне масштабування системи та збереження продуктивності. Тест scalable_storage_test перевірить, чи підтримує система як хмарне, так і локальне сховище, зберігаючи зображення на основі таких критеріїв, як дата, місцезнаходження та теги. Будуть

оцінені показники ефективності зберігання, такі як використання простору та надлишковість даних.

Результати модульного тестування гібридного механізму управління зображеннями зображено на рисунку 5.5.

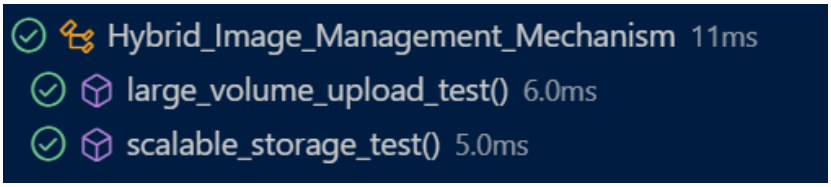


Рисунок 5.5 – Результати модульного тестування гібридного механізму управління зображеннями (рисунок створено самостійно)

Успішне виконання модульних тестів для гібридного механізму управління зображеннями підкреслює здатність системи обробляти великі обсяги завантажень зображень, витримувати високі трафікові навантаження та ефективно управляти ресурсами сховища. Це гарантує, що користувачі можуть безперешкодно завантажувати та зберігати великі обсяги зображень, зберігаючи при цьому високу продуктивність та надійність, що сприяє покращенню користувацького досвіду.

Можливості розширеного пошуку будуть протестовані за допомогою тесту advanced_search_test, щоб переконатися, що система може шукати на основі таких атрибутів, як колір, розмір, роздільна здатність і метадані. Під час тесту буде виміряно точність, швидкість та релевантність пошуку. Тест image_analysis_algorithms_test перевірить реалізацію алгоритмів для таких завдань, як виявлення об’єктів, класифікація та сегментація зображень, оцінюючи точність, час обробки та використання ресурсів.

Тест image_classification_models_test перевірятиме розробку моделей класифікації зображень з використанням алгоритмів машинного навчання, вимірюючи точність класифікації, продуктивність моделі та час навчання. Інтеграція зі сторонніми інструментами та бібліотеками буде перевірена за допомогою тесту third_party_integration_test, що забезпечить безперешкодне розширення функціональності.

Можливості обробки та аналізу в реальному часі будуть протестовані за допомогою тесту real_time_processing_test, обробляючи потокові дані зображень та надаючи своєчасну інформацію. Цей тест вимірюватиме швидкість обробки, затримку та пропускну здатність. Масштабованість, продуктивність та використання ресурсів компонентів обробки та аналізу буде оцінено за допомогою тесту scalability_performance_test, щоб переконатися, що система може ефективно справлятися зі зростаючими робочими навантаженнями.

Результати модульного тестування ефективної обробки та аналізу в умовах великих даних зображено на рисунку 5.6.

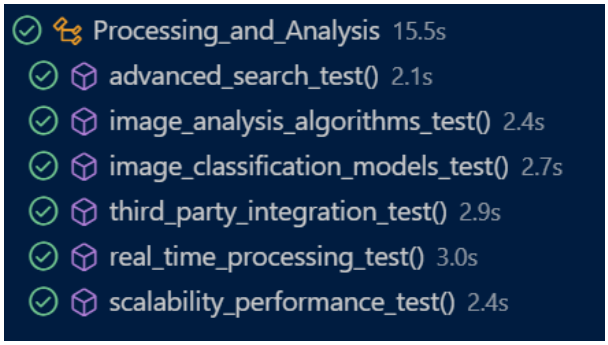


Рисунок 5.6 – Результати модульного тестування ефективної обробки та аналізу в умовах великих даних (рисунок створено самостійно)

Успішне виконання модульних тестів д ефективної обробки та аналізу підкреслює майстерність системи в розширеному пошуку зображень, алгоритмах аналізу, моделях класифікації, інтеграції зі сторонніми інструментами та можливостях обробки в реальному часі. Успішне виконання цих модульних тестів підтверджує надійність, точність та ефективність можливостей обробки та аналізу системи, забезпечуючи безперебійну роботу користувачів у веб-орієнтованій системі.

Тестування дизайну користувацького інтерфейсу передбачає забезпечення чистого, сучасного дизайну з інтуїтивно зрозумілою навігацією та візуально привабливими елементами. Тест ui_design_test перевірить, чи є елементи інтерфейсу адаптивними та оптимізованими для різних пристроїв і розмірів екрану.

Тестування продуктивності, що охоплюється тестом performance_test, забезпечить швидке завантаження інтерфейсу та безперебійну взаємодію. Цей тест

також перевірить ефективність операцій обробки зображень, мінімізуючи затримки та затримки.

Тест security_test перевірить надійність та безпеку механізмів автентифікації та авторизації користувачів. Цей тест також забезпечить шифрування процесів завантаження та вивантаження файлів, зберігаючи цілісність та конфіденційність даних.

Тестування масштабованості, що охоплюється тестом scalability_test, гарантує, що інтерфейс може обробляти зростаючу кількість користувачів і файлів зображень без погіршення продуктивності. Цей тест імітує умови високого навантаження, щоб виявити та вирішити потенційні проблеми масштабованості.

Тест compatibility_test перевірить сумісність інтерфейсу з широким спектром веб-браузерів і пристроїв. Цей тест гарантує, що користувачі матимуть однаковий досвід роботи на різних платформах і що будь-які проблеми, пов’язані з браузером або пристроєм, будуть вирішені.

Результати модульного тестування нефункціональних вимог зображено на рисунку 5.7.

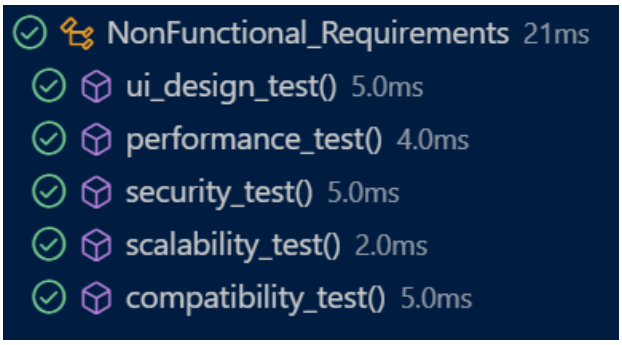


Рисунок 5.7 – Результати модульного тестування нефункціональних вимог (рисунок створено самостійно)

Успішне виконання модульних тестів нефункціональних вимог підтверджує надійність, швидкість реагування та безпеку користувацького інтерфейсу веб-орієнтованої системи управління зображеннями. Успішне виконання цих модульних тестів підтверджує дизайн, продуктивність, безпеку, масштабованість та сумісність користувацького інтерфейсу, забезпечуючи надійний та зручний досвід для користувачів.

Отже, проведене модульне тестування front-end частини веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних охопило широкий спектр функціональних можливостей, гарантуючи, що кожен компонент працює так, як очікувалося, та відповідає як функціональним, так і нефункціональним вимогам. Для функціональних вимог було проведено ретельне тестування авторизації/реєстрації, пошуку файлів, додавання файлів, перегляду файлів, перегляду інформації про файли, завантаження файлів, відкриття файлів, копіювання/вирізання/вставки файлів, редагування назв папок і видалення папок, перейменування та видалення файлів, а також відображення використання пам’яті. Кожна функція була протестована за допомогою спеціальних тестових кейсів для перевірки її коректності, безпеки та зручності використання. Також було ретельно протестовано реалізацію гібридного механізму управління зображеннями та функцій обробки/аналізу для забезпечення ефективного завантаження, вивантаження, зберігання, пошуку та аналізу даних зображень. Масштабованість, продуктивність та інтеграція зі сторонніми інструментами та бібліотеками також були оцінені, щоб гарантувати, що система зможе впоратися зі зростаючими робочими навантаженнями та надавати своєчасну інформацію. Щодо нефункціональних вимог, тестування охоплювало дизайн інтерфейсу користувача, продуктивність, безпеку, масштабованість та сумісність. Було перевірено, що користувальницький інтерфейс має чистий та сучасний дизайн з інтуїтивно зрозумілою навігацією, а тестування продуктивності забезпечило швидке завантаження та безперебійну взаємодію. Заходи безпеки були протестовані для захисту даних користувача від несанкціонованого доступу, а тестування масштабованості гарантувало, що система зможе впоратися зі зростаючими обсягами користувачів і файлів зображень. Тестування сумісності підтвердило, що інтерфейс безперебійно працює в різних веб-браузерах і на різних пристроях. Загалом, процес модульного тестування забезпечив впевненість у надійності, безпеці та продуктивності інтерфейсної частини, заклавши міцний фундамент для її успішного розгортання та використання.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра була розроблена front-end частина веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних.

Ця частина програми відіграє важливу роль у забезпеченні ефективної взаємодії користувачів з системою, полегшуючи управління, обробку та аналіз даних зображень у контексті великих даних.

Завдяки ретельній розробці, інтерфейс надає користувачам інтуїтивно зрозумілі функції для управління зображеннями. Користувачі можуть безпечно реєструватися та автентифікуватися, шукати, додавати, переглядати, завантажувати та маніпулювати файлами зображень. Крім того, інтерфейс відображає відповідну інформацію про файл та використання пам’яті, що покращує розуміння та контроль користувача над своїми даними.

Аналіз предметної області виявив широкий спектр технологій, які застосовуються в індустрії управління зображеннями. Поширення цифрових зображень у різних секторах викликало потребу в ефективній організації, обробці та аналізі даних. В результаті проведеного аналізу були виявлені проблеми зі зберіганням, обробкою та аналізом великих обсягів цифрових зображень.

Щодо актуалізації задач, було розроблено інтуїтивний користувацький інтерфейс, який полегшує навігацію та взаємодію з даними зображень. Також були реалізовані функції для завантаження, впорядкування та управління файлами зображень, а також забезпечено механізми безпеки даних та зручного пошуку. Додатково, були забезпечені механізми автентифікації та авторизації користувачів, реєстрації, керування доступом та обробки помилок для стабільної та безперебійної роботи системи.

Розробка фронтенд-компоненту передбачала використання різних технологій і методологій, зокрема HTML, CSS, JavaScript та фреймворк React. Ці технології дозволяють створити адаптивний, візуально привабливий інтерфейс, який відповідає потребам та вподобанням користувачів.

Дизайн інтерфейсу відповідає загальним цілям гібридного механізму управління зображеннями, яка має на меті забезпечити ефективну обробку та аналіз даних зображень у контексті великих даних. Зосереджуючись на зручності використання, продуктивності та функціональності, інтерфейсний компонент сприяє підвищенню ефективності системи в роботі з великими наборами даних зображень та сприяє отриманню значущих висновків за допомогою аналізу.

Прийняті програмні рішення та тестування розробленого програмного забезпечення призвели до створення надійної та функціональної front-end частини веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних. Завдяки детальному плануванню, впровадженню та тестуванню, система успішно відповідає визначеним функціональним та нефункціональним вимогам.

Програмне рішення включає різноманітні функції, зокрема авторизацію/реєстрацію, пошук файлів, додавання файлів, перегляд файлів (плитка, список), перегляд інформації про файл, завантаження файлів, відкриття файлів, копіювання файлів, вирізання та вставлення в іншу папку, редагування назв папок, видалення папок, перейменування та видалення файлів, а також відображення використання пам’яті. Також система реалізує гібридний механізм управління зображеннями, який підтримує одночасне завантаження великого обсягу зображень, ефективне завантаження та вивантаження з мінімальною затримкою, масштабовані рішення для зберігання зростаючих обсягів зображень, а також зберігання на основі таких критеріїв, як дата, місцезнаходження, категорія та визначені користувачем теги. Він також включає розширені можливості пошуку на основі атрибутів зображень, таких як колір, розмір, роздільна здатність та метадані, алгоритми аналізу зображень для таких завдань, як виявлення об’єктів, класифікація та сегментація зображень, моделі класифікації зображень, навчені на наборах даних з мітками та можливості обробки та аналізу потокових даних зображень в режимі реального часу, а також масштабованість, моніторинг і оптимізацію продуктивності. Ці функції реалізовані з акцентом на зручність, ефективність та безпеку, забезпечуючи безперебійну роботу користувачів.

Процес тестування, що охоплював модульне тестування, ретельно оцінив кожен аспект функціональності системи. Від автентифікації користувача до завантаження, вивантаження та управління файлами, кожен компонент був ретельно протестований для забезпечення надійності, продуктивності та безпеки.

В цілому, застосування відповідних програмних рішень та ретельних методологій тестування дозволило розробити високоякісну систему управління зображеннями. Система ефективно задовольняє потреби користувачів, надаючи їм потужний інструмент для ефективної обробки, аналізу та управління зображеннями.

Матеріали кваліфікаційної роботи пройшли апробацію на XII міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (див. дод. В). [16]

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Flickr [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Flickr (дата звернення: 05.05.2024).

2. Photobucket [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Photobucket (дата звернення 05.05.2024).

3. Picasa [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Picasa (дата звернення 05.05.2024).

4. How many pictures are there (2024): [Електронний ресурс] – Режим доступу до ресурсу: https://photutorial.com/photos-statistics (дата звернення 25.04.2024).

5. Терещенко Г.Ю., Стась Б.Л. Архітектура Сховища Зображень в Епоху Великих Даних. // XIII International Scientific and Practical Conference «Eurasian scientific discussions», (January 22-24, 2023), Barcelona, Spain, 2023. –Barca Academy Publishing,478 p. – PP. 191 - 195.

6. Google Photos [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/photos/about (дата звернення 13.05.2024).

7. Документація PhotoPrism [Електронний ресурс] – Режим доступу до ресурсу: https://docs.photoprism.app (дата звернення 13.05.2024).

8. Документація Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/all.html (дата звернення 05.05.2024).

9. Документація Azure Cosmos DB [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/azure/cosmos-db (дата звернення 05.05.2024).

10. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3-тє вид. Print2print, 2016. 208 с.

11. Терещенко Г. Ю., Боцюра І. С. Image warehouse architecture in the era of big data // 25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті», Збірник Матеріалів форуму., т. 6, - Харків (ХНУРЕ), 20-22 квітня 2021 р., с. 163 - 164.

12. Tereshchenko G., Kyrychenko I., Chetverykov G. Overview of image storage models in Big Data conditions // Комп’ютерна математика в науці, інженерії та освіті

(CMSEE-2020). – Матеріали V Всеукраїнської науково-технічної конференції – Полтава, 27 листопада 2020 р. – С. 18–21. УДК 00.89:004.043.

13. Документація Azure Blob Storage [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction (дата звернення 13.05.2024).

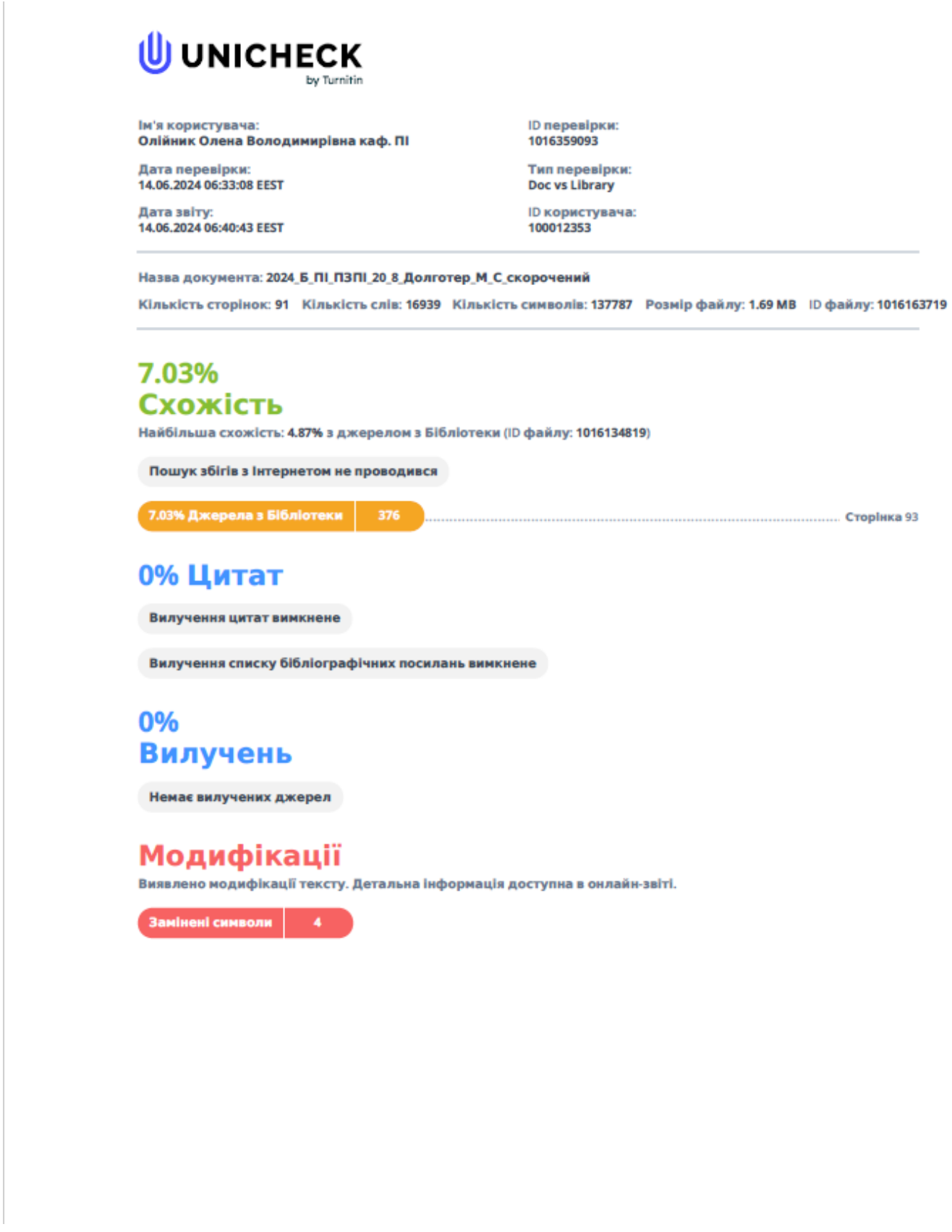
14. Документація Azure AI Vision [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/products/ai-services/ai-vision (дата звернення 13.05.2024).

15. What is unit testing? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/what-is/unit-testing (дата звернення: 20.05.2024).

16. Терещенко Г. Ю., Мандзинець А. В., Долготер М. С. Проектування веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних // VI Всеукраїнська студентська конференція «Експериментальні та теоретичні дослідження в контексті сучасної науки», Збірник Матеріалів конференції, - Рівне, 21 червня 2024 р., с. 18 - 22.

ДОДАТОК А

Звіт результатів перевірки унікальності тексту



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016359093

Дата перевірки:
14.06.2024 06:33:08 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
14.06.2024 06:40:43 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_8_Долготер_М_С_скорочений

Кількість сторінок: 91 Кількість слів: 16939 Кількість символів: 137787 Розмір файлу: 1.69 MB ID файлу: 1016163719

7.03%
Схожість

Найбільша схожість: 4.87% з джерелом з Бібліотеки (ID файлу: 1016134819)

Пошук збігів з Інтернетом не проводився

7.03% Джерела з Бібліотеки 376 ..... Сторінка 93

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

ДОДАТОК Б

Перелік джерел посилання за науковими напрямками керівника та науковців кафедри програмної інженерії

5. Терещенко Г.Ю., Стась Б.Л. Архітектура Сховища Зображень в Епоху Великих Даних. // XIII International Scientific and Practical Conference «Eurasian scientific discussions», (January 22-24, 2023), Barcelona, Spain, 2023. –Barca Academy Publishing,478 p. – PP. 191 - 195.

11. Терещенко Г. Ю., Боцюра І. С. Image warehouse architecture in the era of big data // 25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті», Збірник Матеріалів форуму., т. 6, - Харків (ХНУРЕ), 20-22 квітня 2021 р., с. 163 - 164.

12. Tereshchenko G., Kyrychenko I., Chetverykov G. Overview of image storage models in Big Data conditions // Комп’ютерна математика в науці, інженерії та освіті (CMSEE-2020). – Матеріали V Всеукраїнської науково-технічної конференції – Полтава, 27 листопада 2020 р. – С. 18–21. УДК 00.89:004.043.

16. Терещенко Г. Ю., Мандзинець А. В., Долготер М. С. Проектування веб-орієнтованої системи гібридного механізму управління зображеннями для їх ефективної обробки та аналізу в умовах великих даних // VI Всеукраїнська студентська конференція «Експериментальні та теоретичні дослідження в контексті сучасної науки», Збірник Матеріалів конференції, - Рівне, 21 червня 2024 р., с. 18 - 22.

ДОДАТОК В

Тези VI Всеукраїнської студентської конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки»

ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ГІБРИДНОГО МЕХАНІЗМУ УПРАВЛІННЯ ЗОБРАЖЕННЯМИ ДЛЯ ЇХ ЕФЕКТИВНОЇ ОБРОБКИ ТА АНАЛІЗУ В УМОВАХ ВЕЛИКИХ ДАНИХ

Мандзинець Анатолій Володимирович
здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна
Долготер Михайло Сергійович
здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна
Науковий керівник: Терещенко Гліб Юрійович
ст. викладач кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

Кількість зображень, що створюють та розповсюджують люди постійно росте. Наприклад, Instagram щоденно збільшується на 95 мільйонів зображень. У 2022 році в Інтернеті було близько 136 мільярдів зображень. Але ж крім соціальних мереж зображення широко використовуються різних сферах від освіти та медицини, до океанографії та астрономії.

Розвиток хмарних технологій, таких як Google Photo, Apple iCloud, Amazon Photo, дозволив зручно працювати з великою кількістю зображень, забезпечуючи додаткові можливості, не турбуючись про обмеження пам'яті пристрою. Однак, користувачі повинні покладатися на провайдерів цих послуг щодо безпеки та конфіденційності.

Локальне сховище або власний хостинг, наприклад, Synology, Immich, PhotoPrism, Lychee, є альтернативою. Проте вони мають свої недоліки: погана робота розпізнавання об'єктів, повільне завантаження та пошук, складне налаштування.

Окрім цього, передача великої кількості даних при початковому завантаженні зображень впливає на час завантаження та ефективність

використання пам'яті, що позначається на швидкості пошуку, відображення та завантаження зображень на пристрій.

Тому створення системи гібридного механізму управління зображеннями та реалізація їх ефективної обробки та аналізу в умовах великих даних є актуальною задачею. Основною метою є вибір архітектури та стеку технологій для програмної системи, яка повинна виконувати ряд функцій: реєстрація авторизація користувача, завантаження великої кількості зображень, обробки та аналізу зображень, можливість поділитися зображеннями з іншими. Для пришвидшення завантаження та ефективнішого використання пам'яті виконаємо обробку зображення у вигляді стиснення без помітних втрат якості. В якості аналізу можна додати корисні для користувача функції класифікації зображень, виявлення об'єктів, розпізнавання тексту, автоматичне додавання тегів.

Проектована програмна система буде наслідувати трирівневу архітектуру. Клієнтом буде являтися Web-застосунок. Серверна частина, яка буде представлена у вигляді API, буде реалізована з використанням фреймворку Express, що побудований над Node.js [1]. Цей стек дозволяє доволі легко створювати масштабовані, високопродуктивні веб-додатки. При цьому Node.js надає доступ великої кількості бібліотек, що дозволяють додати готову функціональність у проєкт, наприклад, для стиснення зображень.

Оскільки робота відбувається в умовах великих даних, слід подбати про правильний підхід до управління великою кількістю зображень та даних про них. Загалом виділяють багато підходів для зберігання зображень [2].

Перший – це зберігання зображень у базі даних у вигляді двійкових файлів. Цей метод є простим і ефективним, але він може бути невигідним, коли ми говоримо про зображення в умовах великих даних.

Інший підхід – це зберігати зображення на пристрої, а шлях до файлу зберігати у базі даних. Цей метод забезпечує легкий доступ до зображень, але він дуже залежить від доступних ресурсів пристрою, через що може стати не ефективним.

Третій варіант полягає у використанні служби хмарного зберігання. Це передбачає збереження зображень у хмарному сервісі, а потім збереження URL-адрес у базі даних. Такий метод має високу масштабованість і забезпечує легкий доступ до зображень з будь-якого місця. Хоча мінусом такого рішення може стати його вартість.

У програмній системі, яка розроблюється, обрано гібридний підхід до зберігання таких даних. Для зберігання даних про користувачів та зображення використано нереляційну базу даних. NoSQL бази даних добре підходять для роботи в умовах Big Data [3]. Самі ж файли зберігаються у хмарному сховищі.

В якості бази даних обрано Azure Cosmos DB. Це глобально розподілена, багатомодельна база даних від Azure. Вона забезпечує високу доступність, є горизонтально масштабованою, тобто дозволяє виділяти додаткові ресурси за потреби, і надає декілька рівнів узгодженості даних для балансу між узгодженістю та продуктивністю. За замовчуванням автоматично індексує всі дані.

Також використано хмарний сервіс Azure Blob Storage. Він призначений для зберігання величезних обсягів неструктурованих даних, таких як текстові або двійкові дані. Сховище забезпечує декілька рівнів зберігання та має високу масштабованість, доступність та рівень безпеки, підтримує резервування.

Для аналізу зображень використано Azure AI Vision, що надає доступ до передових алгоритмів, які обробляють зображення та повертають інформацію на основі візуальних характеристик, які цікавлять. Аналіз зображень за допомогою цього сервісу охоплює читання тексту із зображень, додавання тегів, класифікацію, виявлення людей і об'єктів, і це лише невелика частина його можливостей.

Таким чином, при завантажені зображень користувача, вони можуть бути класифіковані, додано теги та розпізнано текст, якщо він там наявний. Після цього зображення одного класу будуть автоматично згруповані, а при перегляді кожного фото окремо, можна побачити всі дані про нього.

Загальний вигляд програмної системи можна побачити на діаграмі розгортання (рис. 1), що показує основні компоненти системи та способи комунікації між ними.

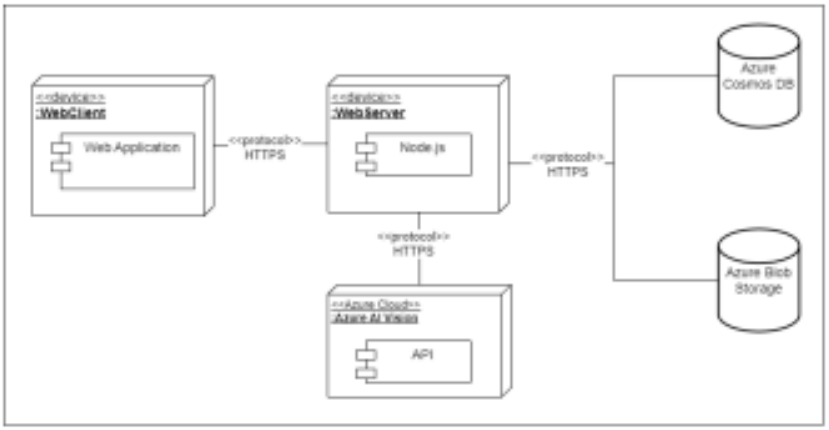


Рис. 1. **Діаграма розгортання програмної системи**

У роботі було представлено архітектуру та обґрунтовано вибір технологій для реалізації веб-орієнтованої системи гібридного механізму управління зображеннями для їх подальшої ефективної обробки та аналізу в умовах великих даних. Такий додаток дозволить користувачам легко працювати з великою кількістю власних та завантажених зображень.

Список використаних джерел:

1. Flanagan D. JavaScript: The Definitive Guide. 7th ed. O'Reilly Media, 2020. 706 p.

2. Терещенко Г. Ю., Боцюра І. С. Image warehouse architecture in the era of big data // 25-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті», Збірник Матеріалів форуму., т. 6, - Харків (ХНУРЕ), 20-22 квітня 2021 р., с. 163 - 164.

3. Tereshchenko G., Kyrychenko I., Chetverykov G. Overview of image storage models in Big Data conditions // Комп'ютерна математика в науці, інженерії та освіті (CMSEE-2020). – Матеріали V Всеукраїнської науково-технічної конференції – Полтава, 27 листопада 2020 р. – С. 18–21. УДК 00.89:004.043.

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Реєстрі громадських об'єднань: 1508433.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574528
IBAN: UA783052990000026003018111950
Банк ВФ АТ Кб «ПриватБанк»; МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7172 від 21.10.2020.

ДОВІДКА

ПРО ПРИЙНЯТТЯ ТЕЗ ДО ПУБЛІКАЦІЇ

09.06.2024

Шановний(і) авторе(и):
Мандзинець Анатолій Володимирович,
Долготер Михайло Сергійович,

Організаційний комітет з радістю повідомляє, що тези «ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ГІБРИДНОГО МЕХАНІЗМУ УПРАВЛІННЯ ЗОБРАЖЕННЯМИ ДЛЯ ЇХ ЕФЕКТИВНОЇ ОБРОБКИ ТА АНАЛІЗУ В УМОВАХ ВЕЛИКИХ ДАНИХ» прийняті до публікації в збірнику за матеріалами VI Всеукраїнської студентської наукової конференції «Експериментальні та теоретичні дослідження в контексті сучасної науки» (21.06.2024, м. Рівне, Україна).

Опубліковані тези будуть доступні з 21.06.2024 за посиланням:
https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-21.06.2024

. . . . . .

Електронні сертифікати учасників конференції та подяки науковим керівникам будуть доступні з 21 червня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться до 5 липня.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та Інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення №34 від 05.01.2024).

З повагою,

Директор Молодіжної наукової ліги
Голова оргкомітету конференції
ІГОР КОРЕНЮК

ДОДАТОК Г

Слайди презентації

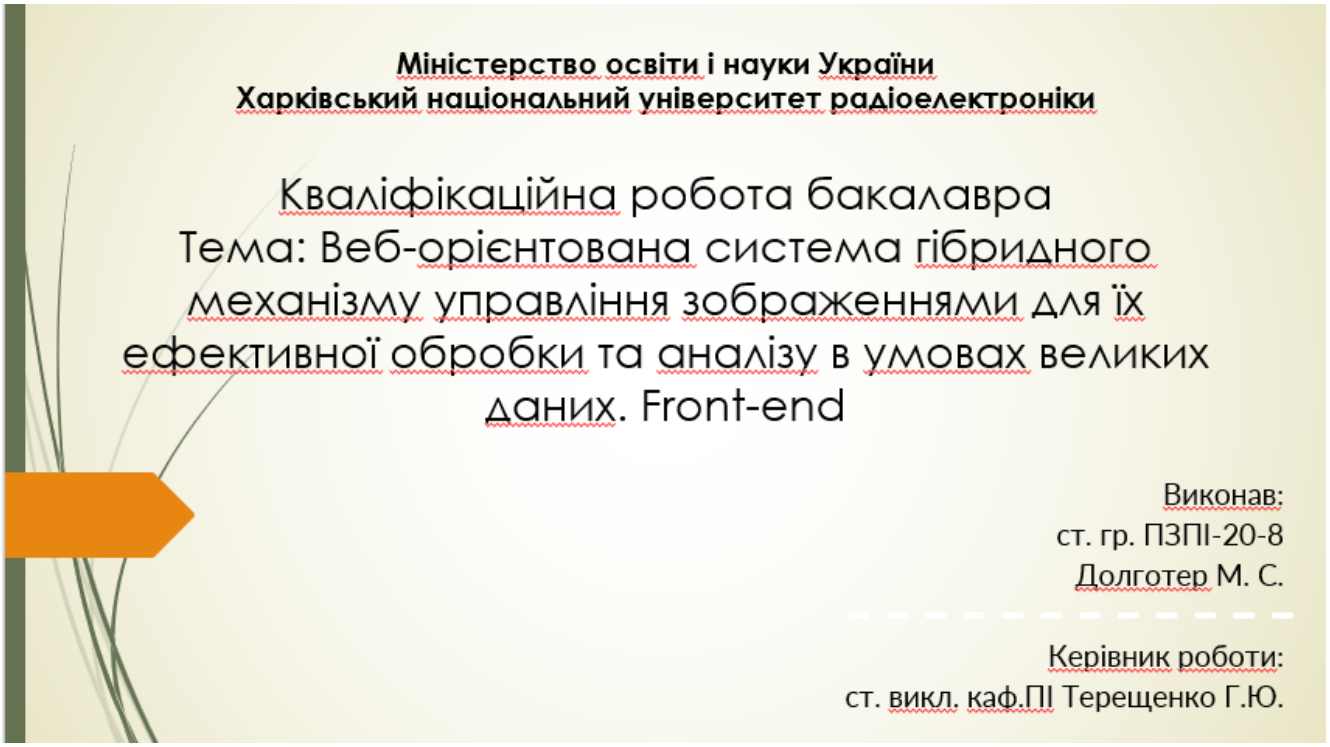


Рисунок Г.1 – Слайд 1

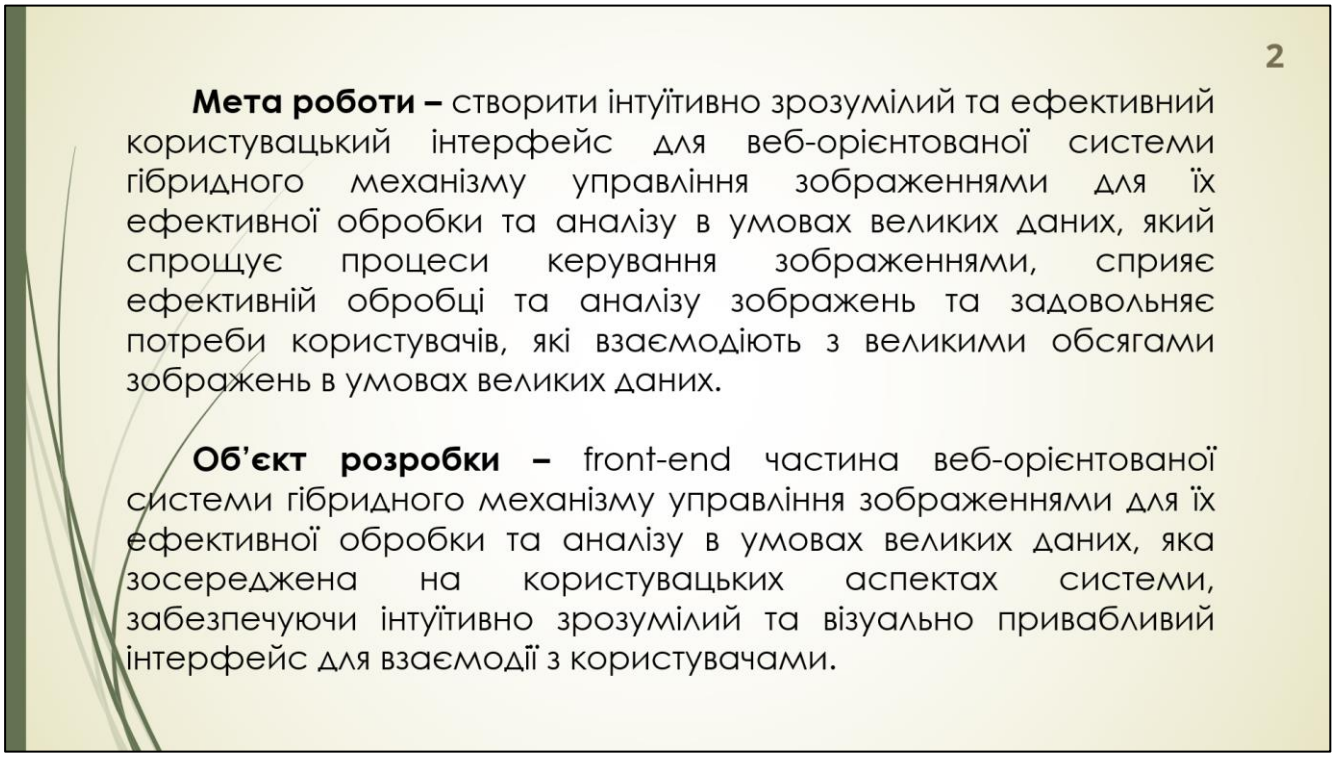


Рисунок Г.2 – Слайд 2

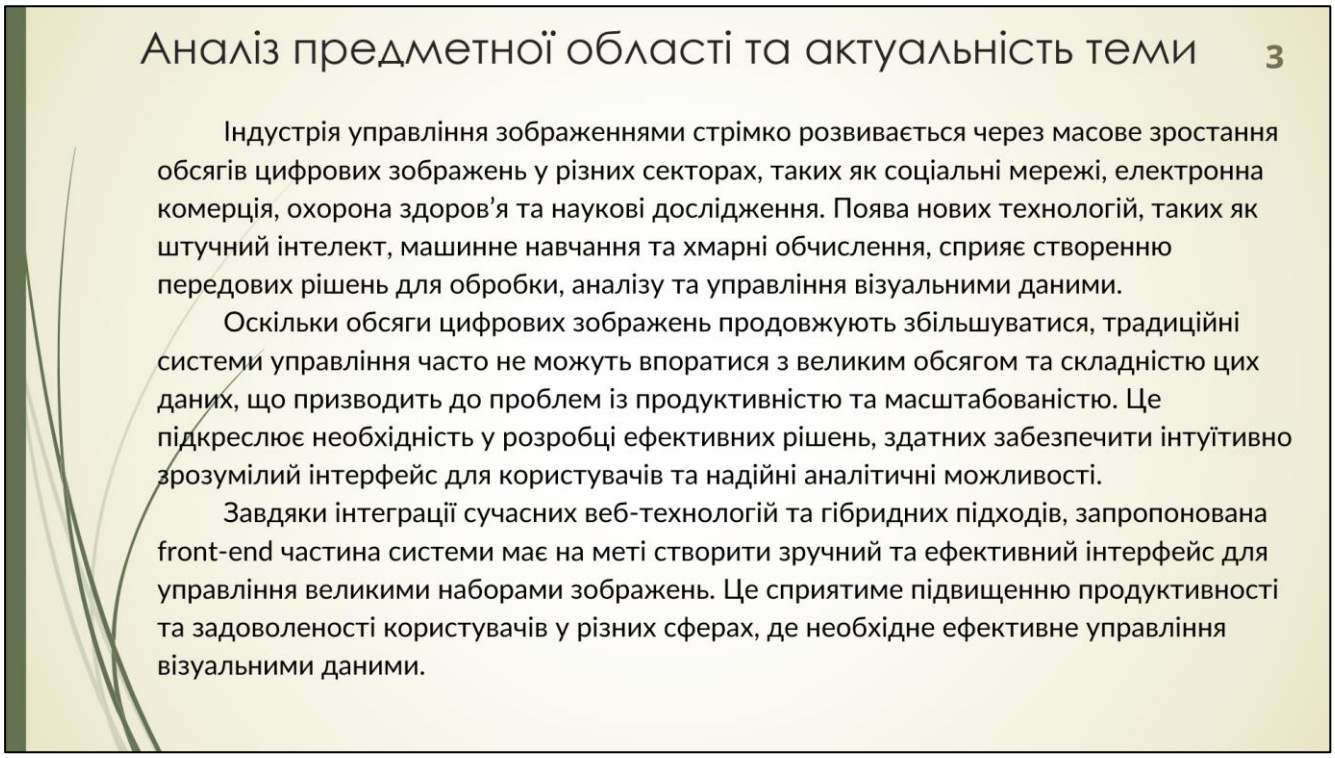


Рисунок Г.3 – Слайд 3

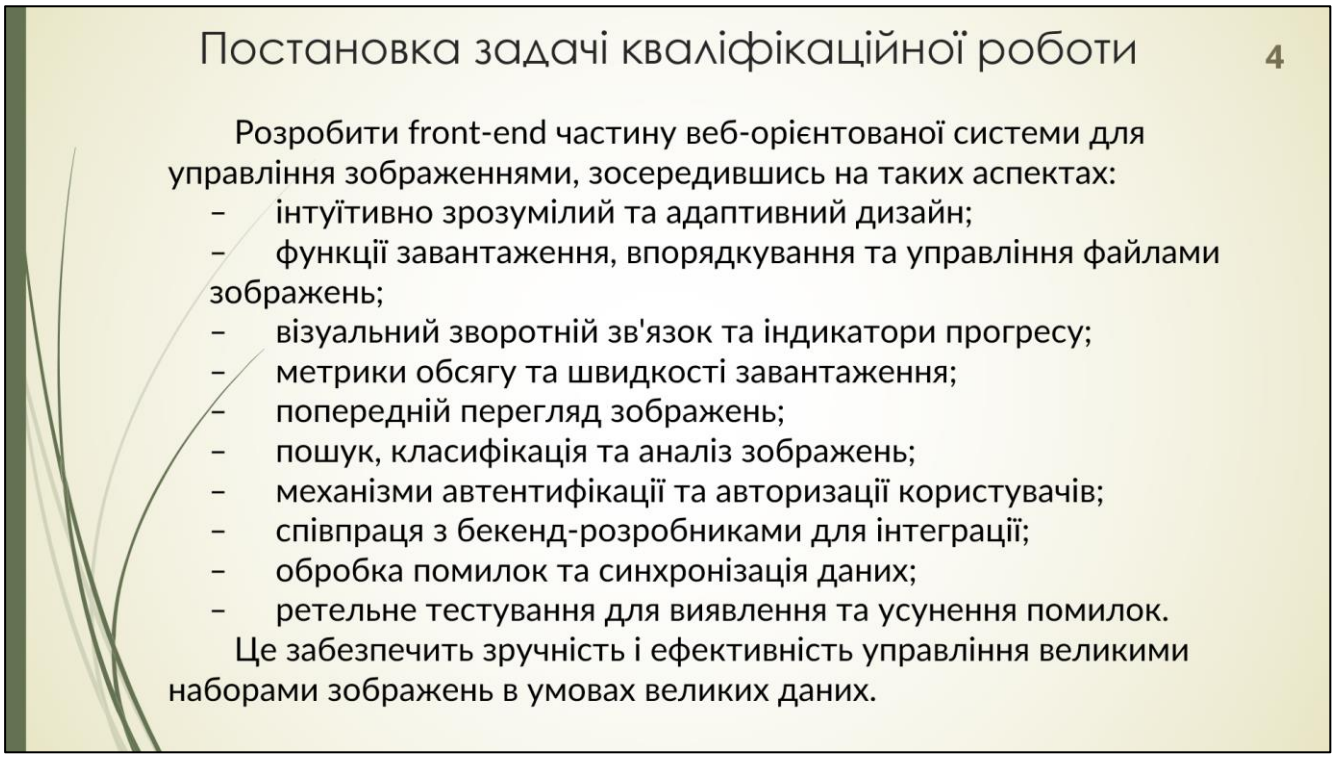


Рисунок Г.4 – Слайд 4

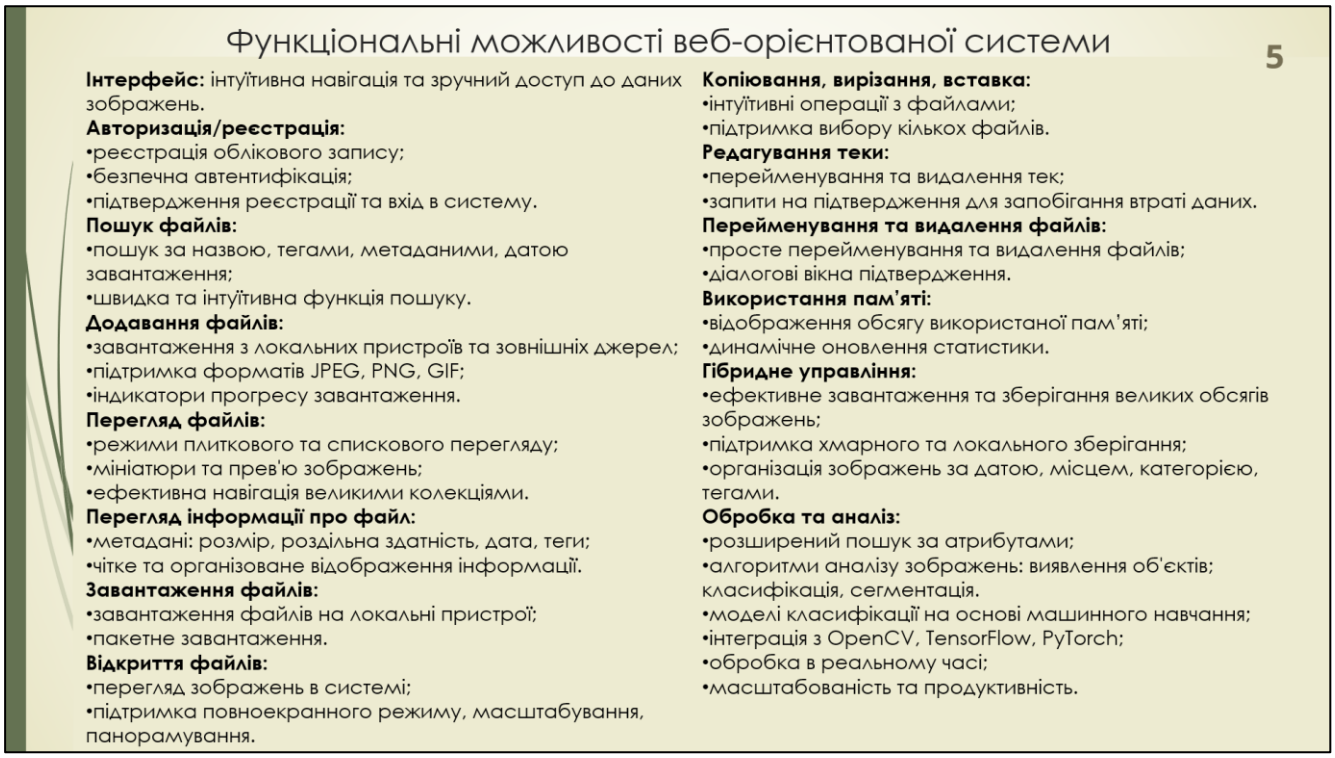


Рисунок Г.5 – Слайд 5

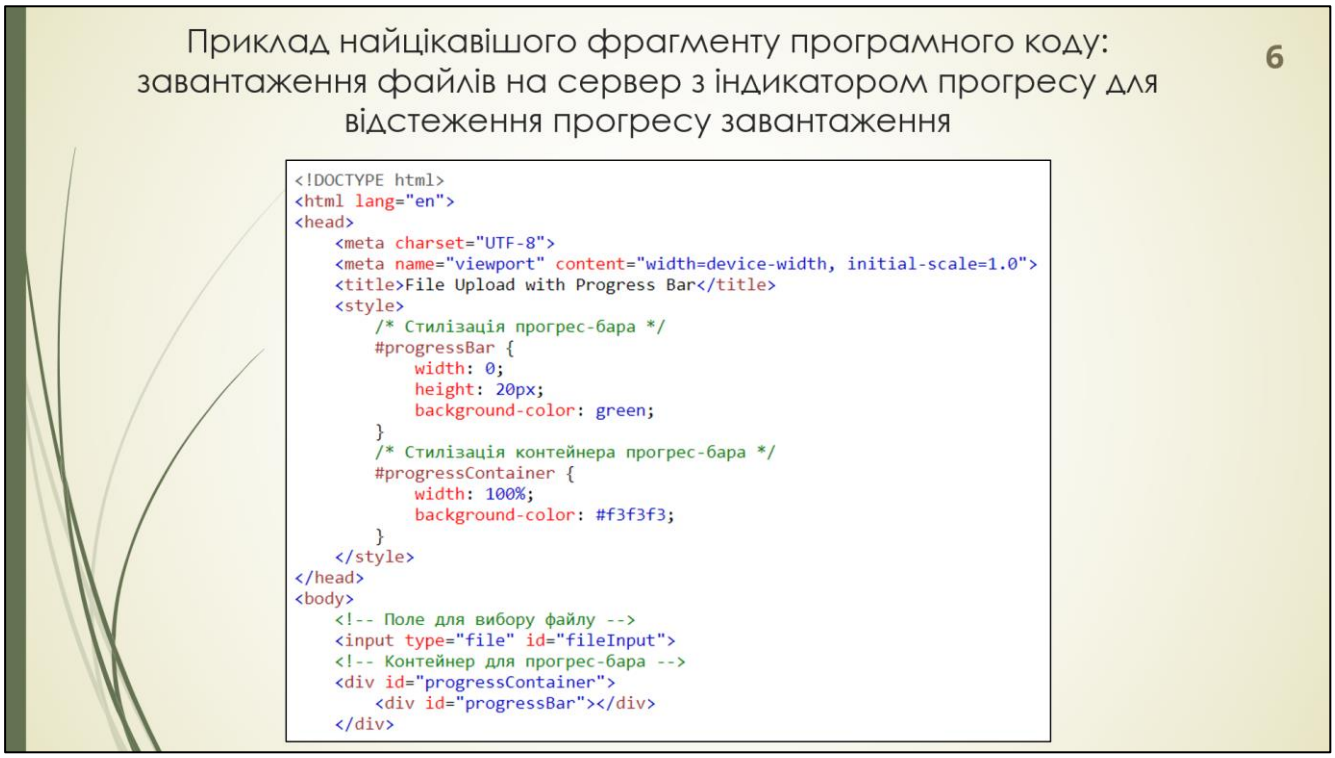


Рисунок Г.6 – Слайд 6

Приклад найцікавішого фрагменту програмного коду: завантаження файлів на сервер з індикатором прогресу для відстеження прогресу завантаження

7

```
<script>
    // Отримання елементів з DOM
    const fileInput = document.getElementById('fileInput');
    const progressBar = document.getElementById('progressBar');

    // Додавання обробника події на зміну вибору файлу
    fileInput.addEventListener('change', (event) => {
        // Отримання вибраного файлу
        const file = event.target.files[0];
        // Створення об'єкта FormData і додавання файлу
        const formData = new FormData();
        formData.append('file', file);

        // Створення об'єкта XMLHttpRequest
        const xhr = new XMLHttpRequest();
        // Налаштування запиту на сервер
        xhr.open('POST', '/upload', true);

        // Встановлення обробника події для відстеження прогресу завантаження
        xhr.upload.onprogress = (progressEvent) => {
            if (progressEvent.lengthComputable) {
                // Розрахунок відсотка завершення завантаження
                const percentCompleted = (progressEvent.loaded / progressEvent.total) * 100;
                // Оновлення ширини прогрес-бара
                progressBar.style.width = percentCompleted + '%';
            }
        };

        // Обробник події для завершення завантаження
        xhr.onload = () => {
            if (xhr.status === 200) {
                console.log('File uploaded successfully');
            } else {
                console.error('File upload failed');
            }
        };

        // Надсилання даних файлу на сервер
        xhr.send(formData);
    });
</script>
</body>
</html>
```

Рисунок Г.7 – Слайд 7

Діаграма варіантів використання

8

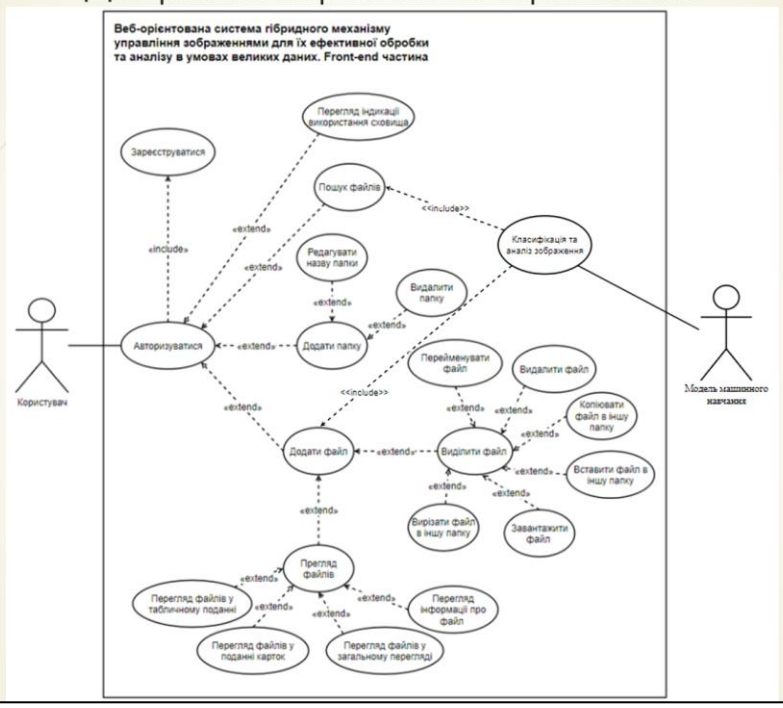


Рисунок Г.8 – Слайд 8

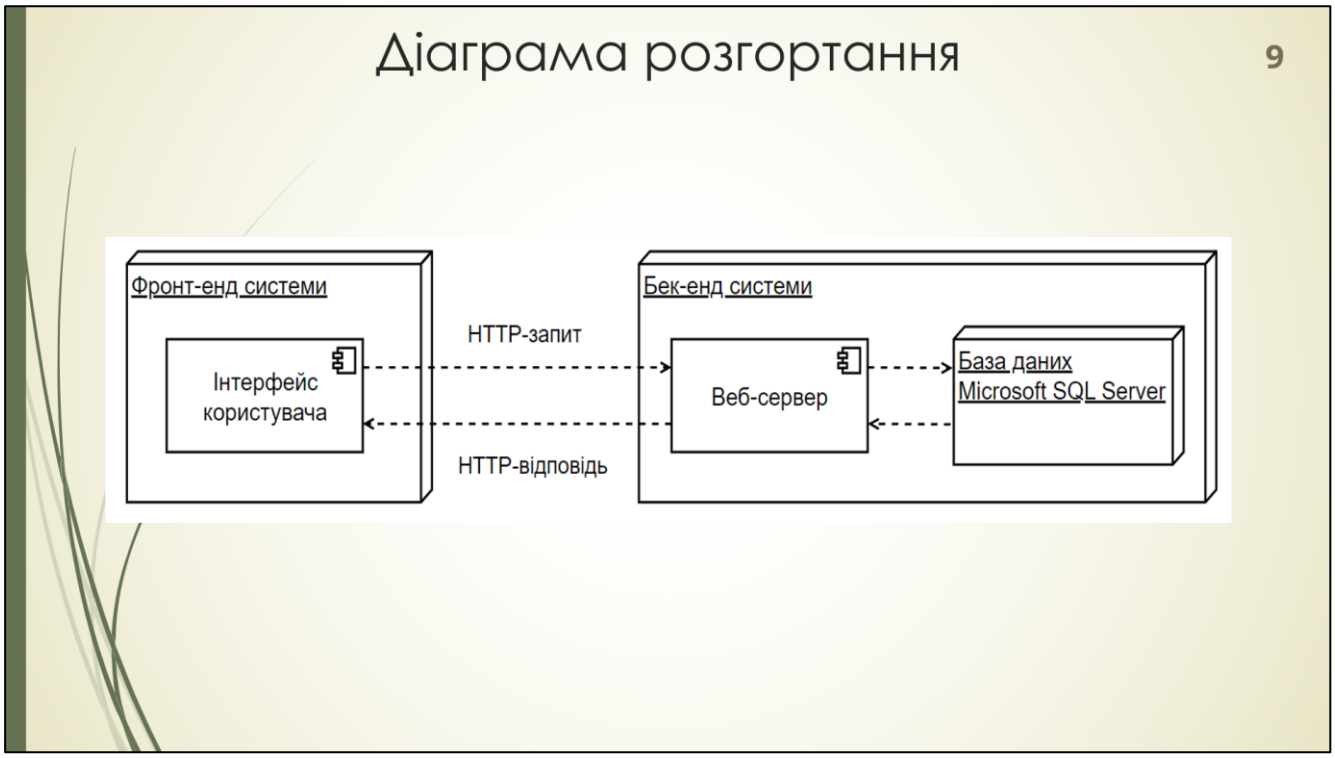


Рисунок Г.9 – Слайд 9

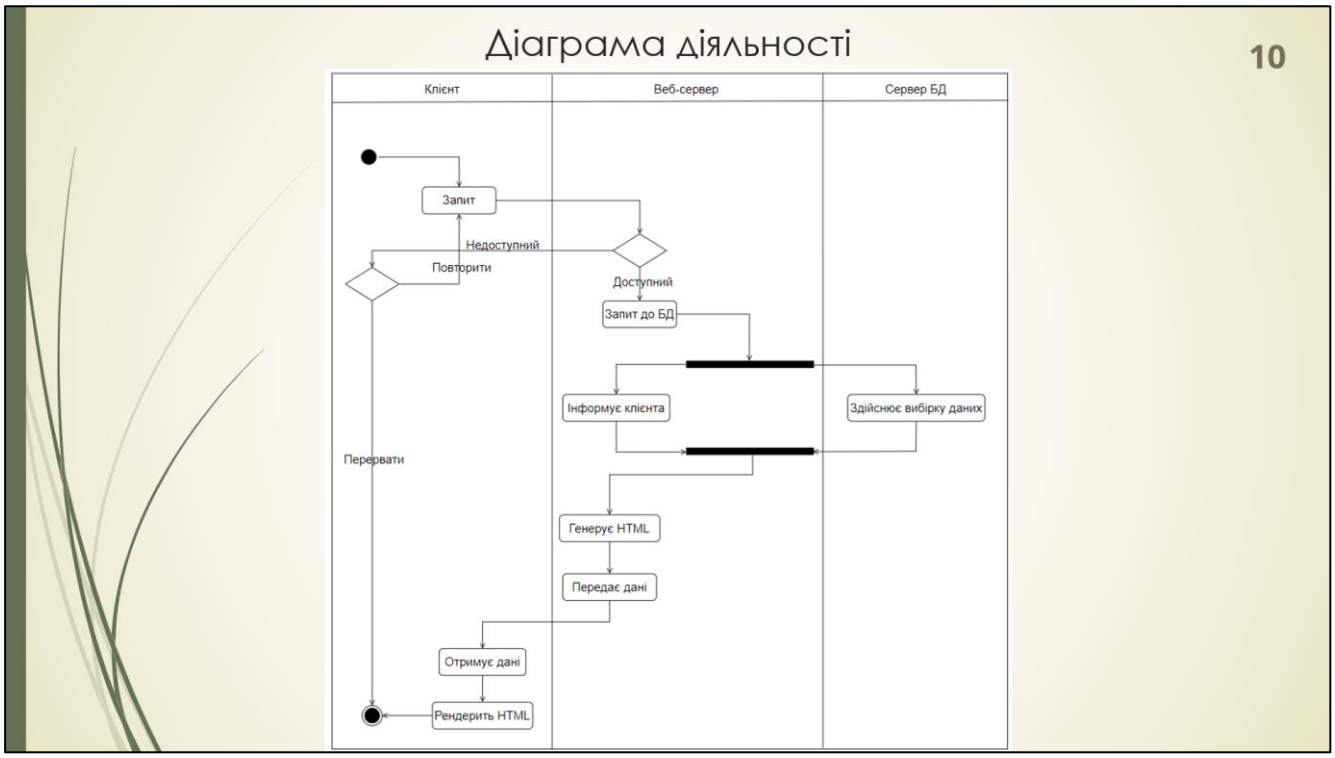


Рисунок Г.10 – Слайд 10

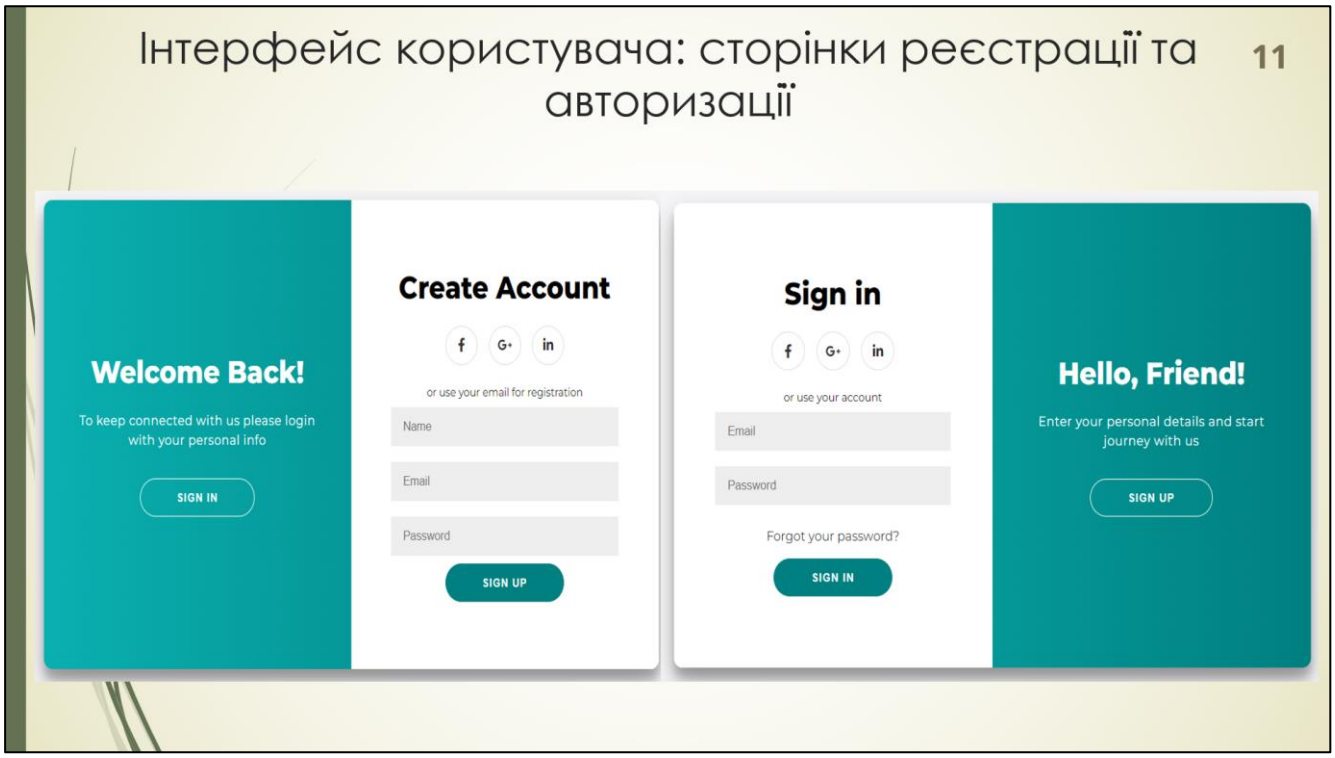


Рисунок Г.11 – Слайд 11

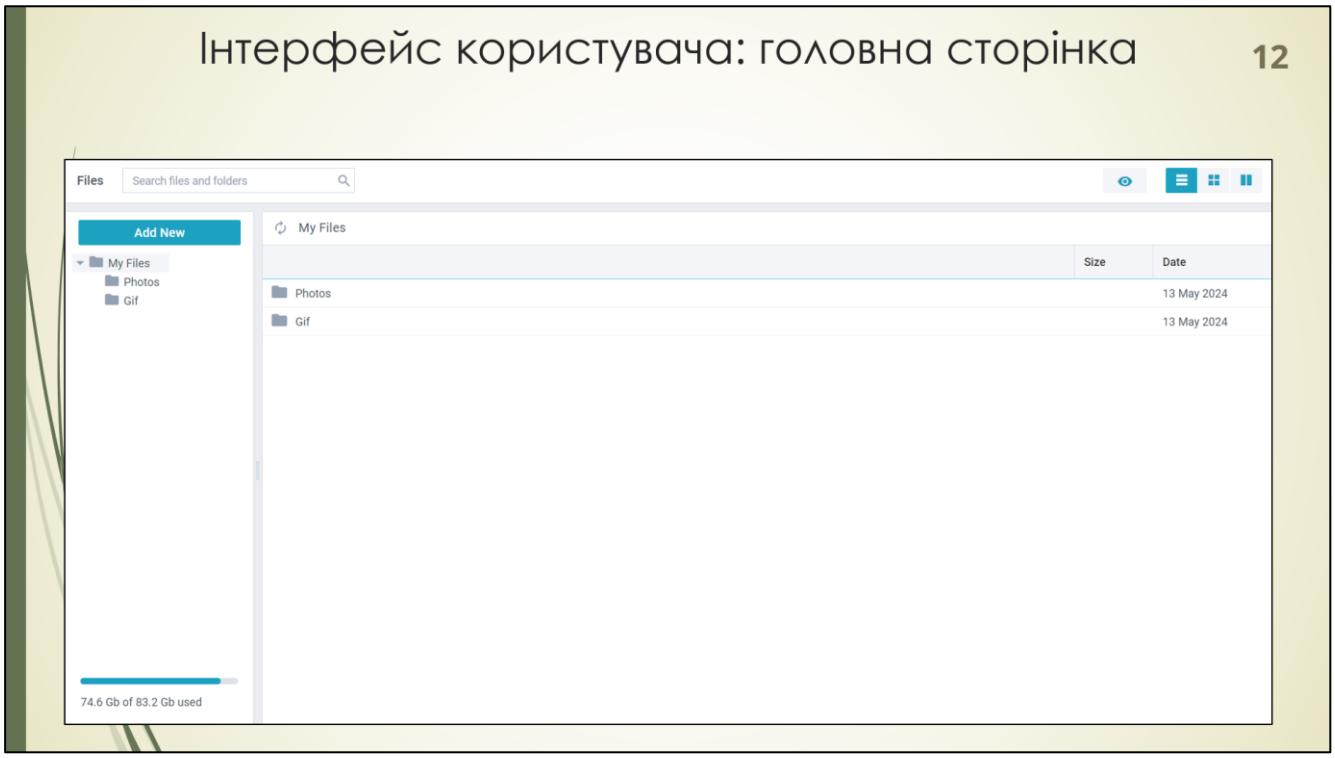


Рисунок Г.12 – Слайд 12

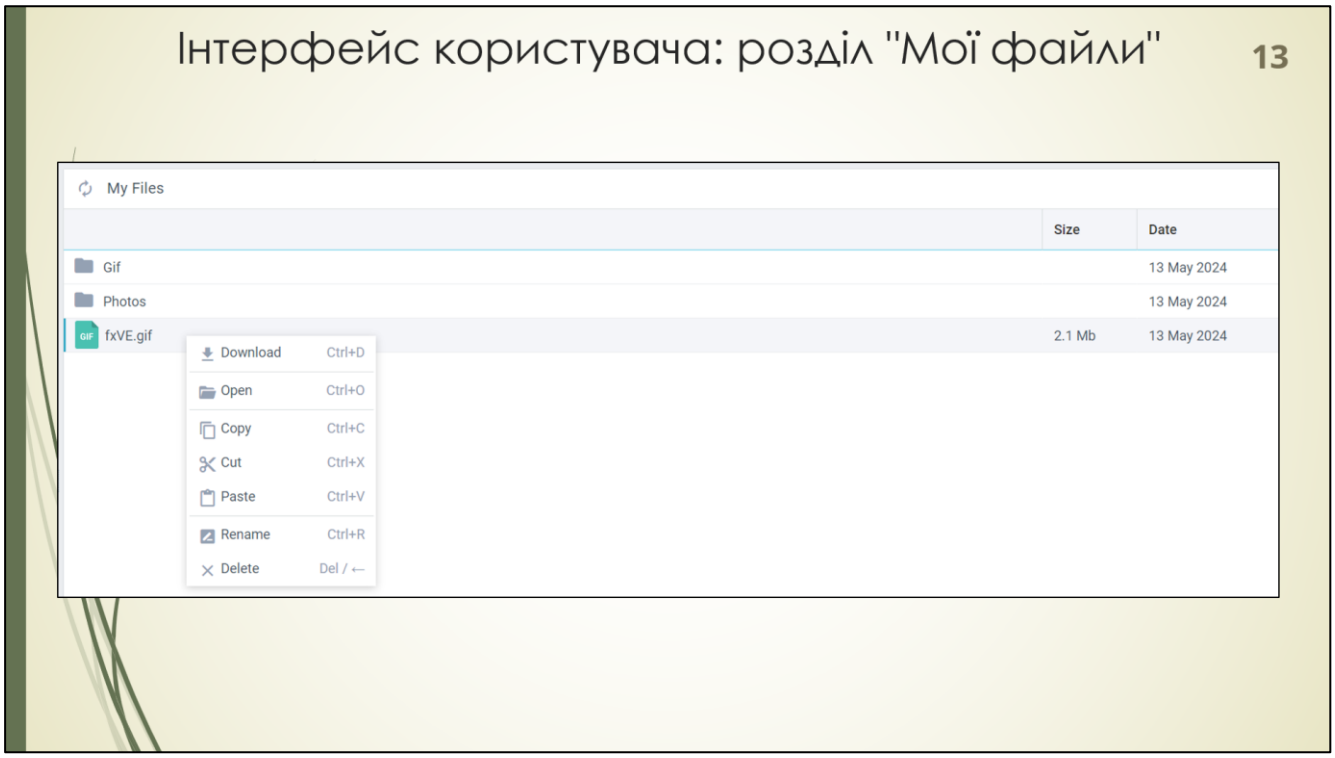


Рисунок Г.13 – Слайд 13

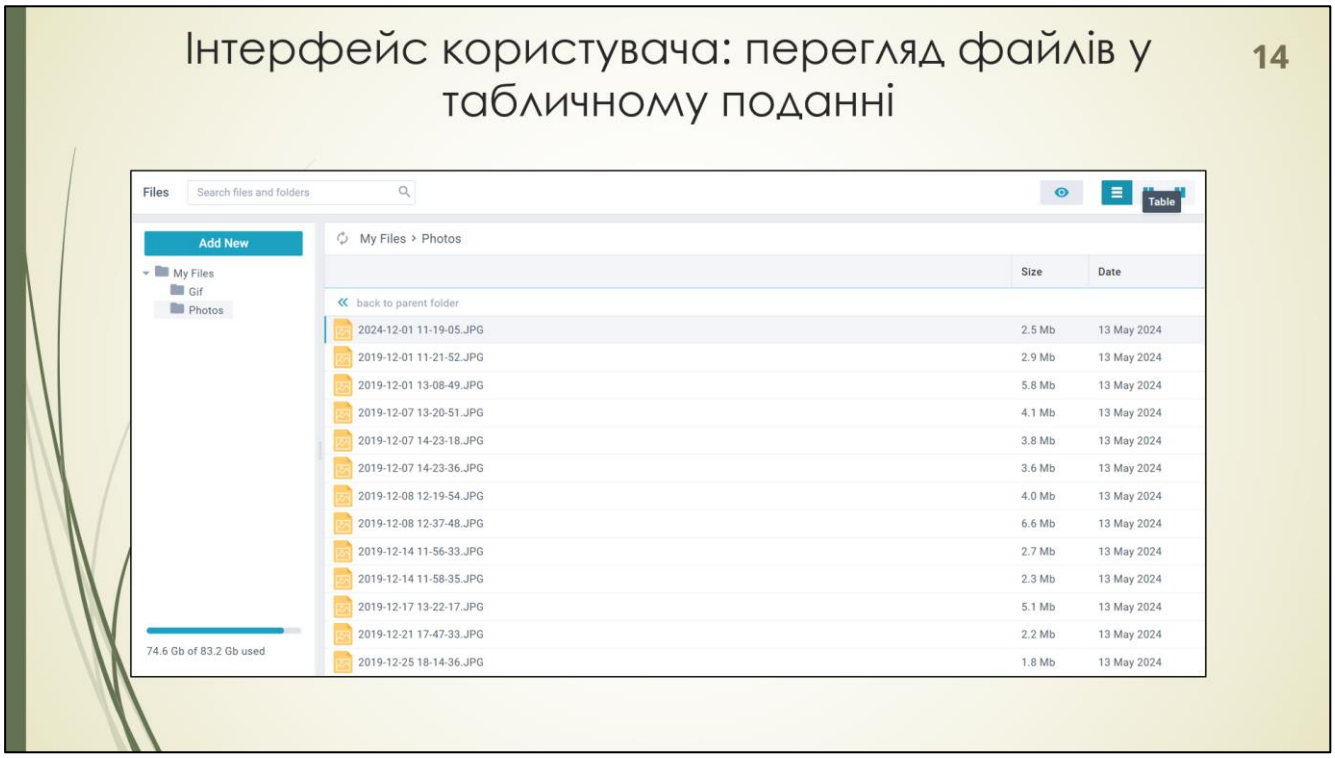


Рисунок Г.14 – Слайд 14

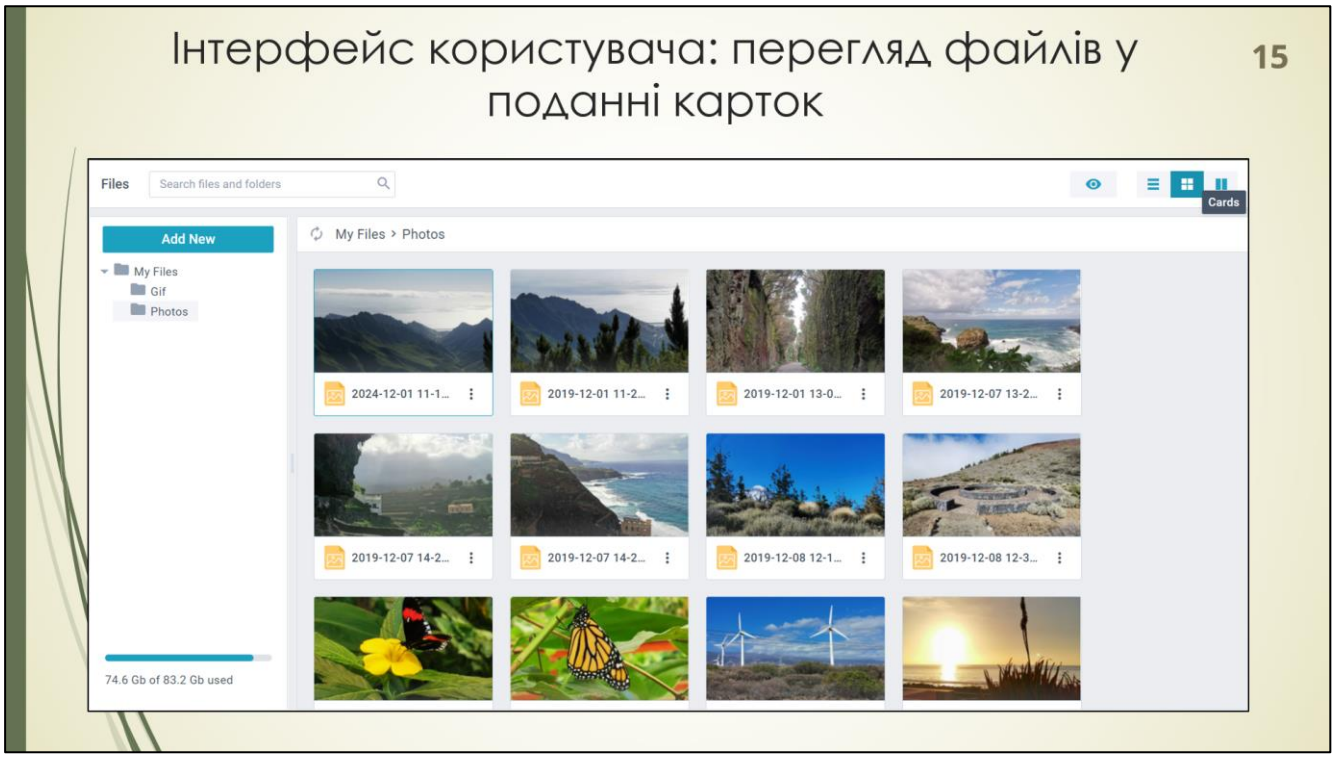


Рисунок Г.15 – Слайд 15

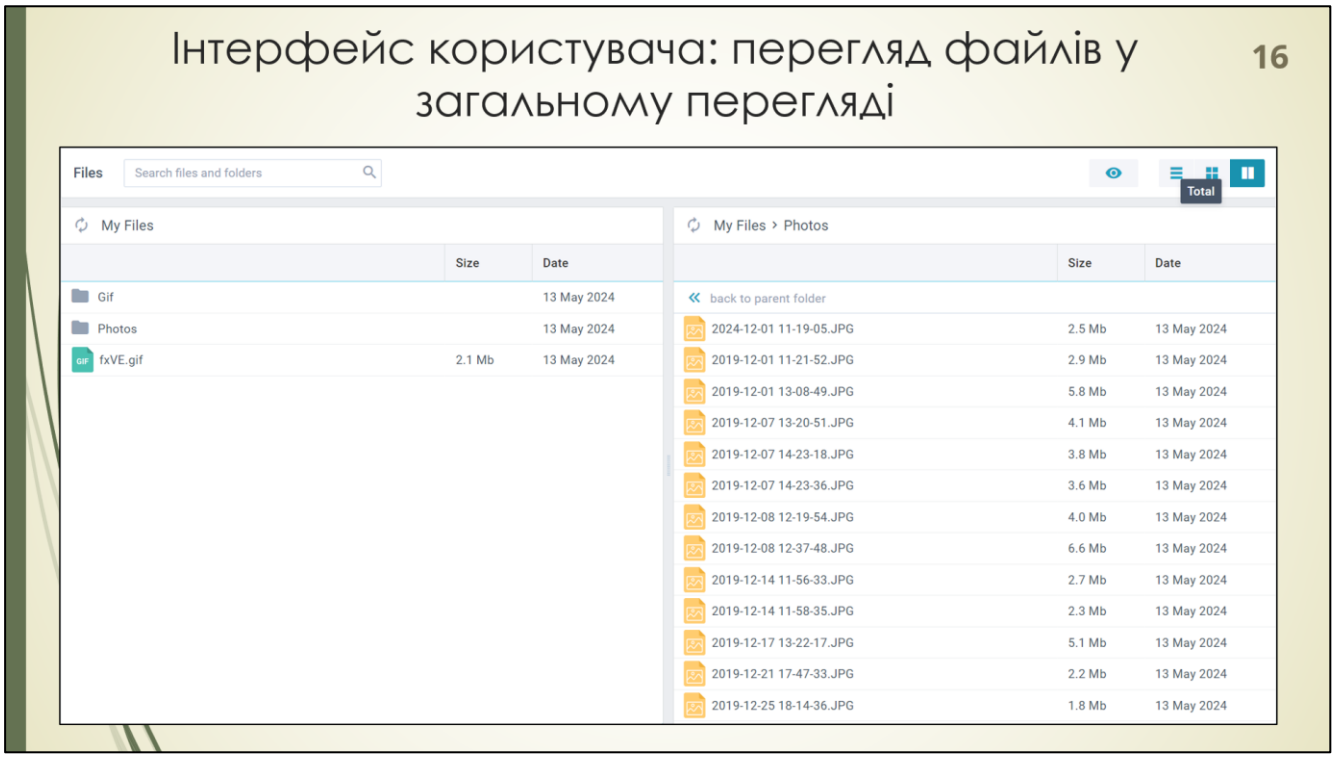


Рисунок Г.16 – Слайд 16

Рисунок Г.17 – Слайд 17

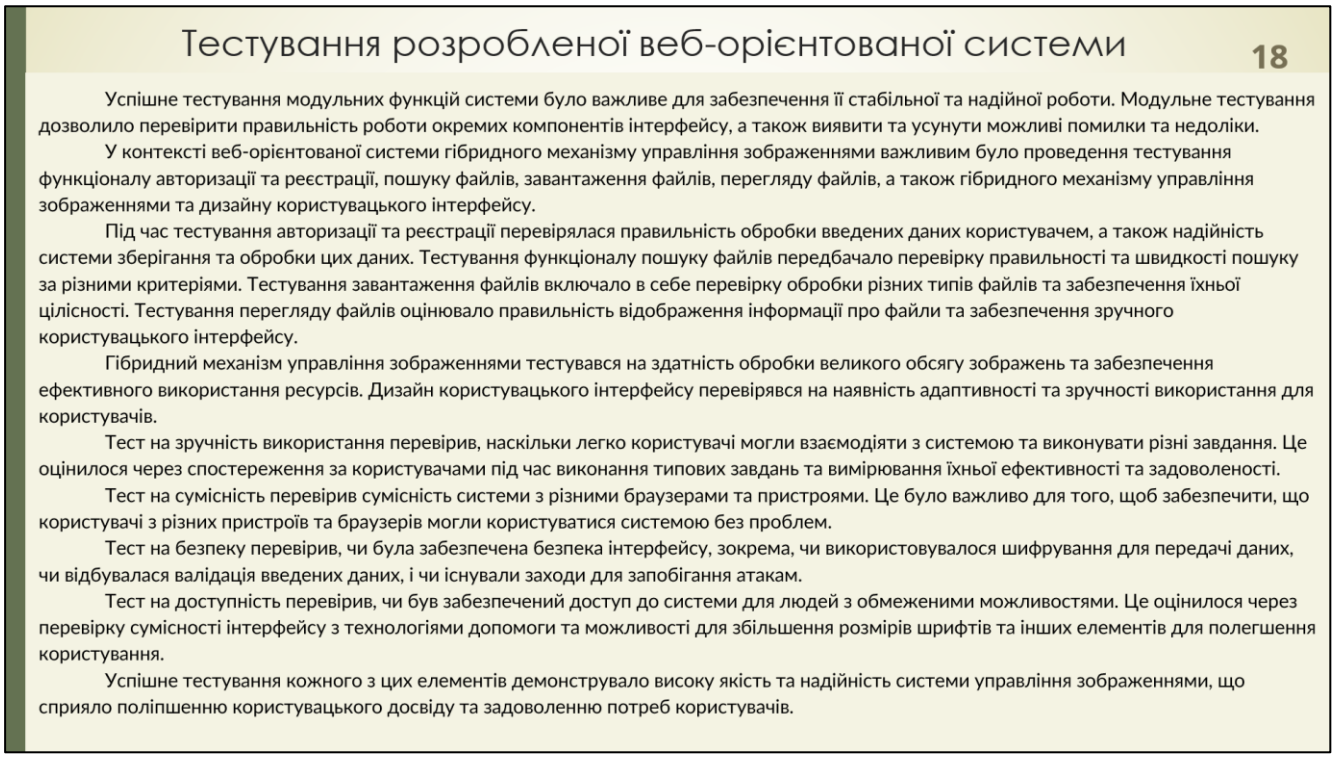


Рисунок Г.18 – Слайд 18

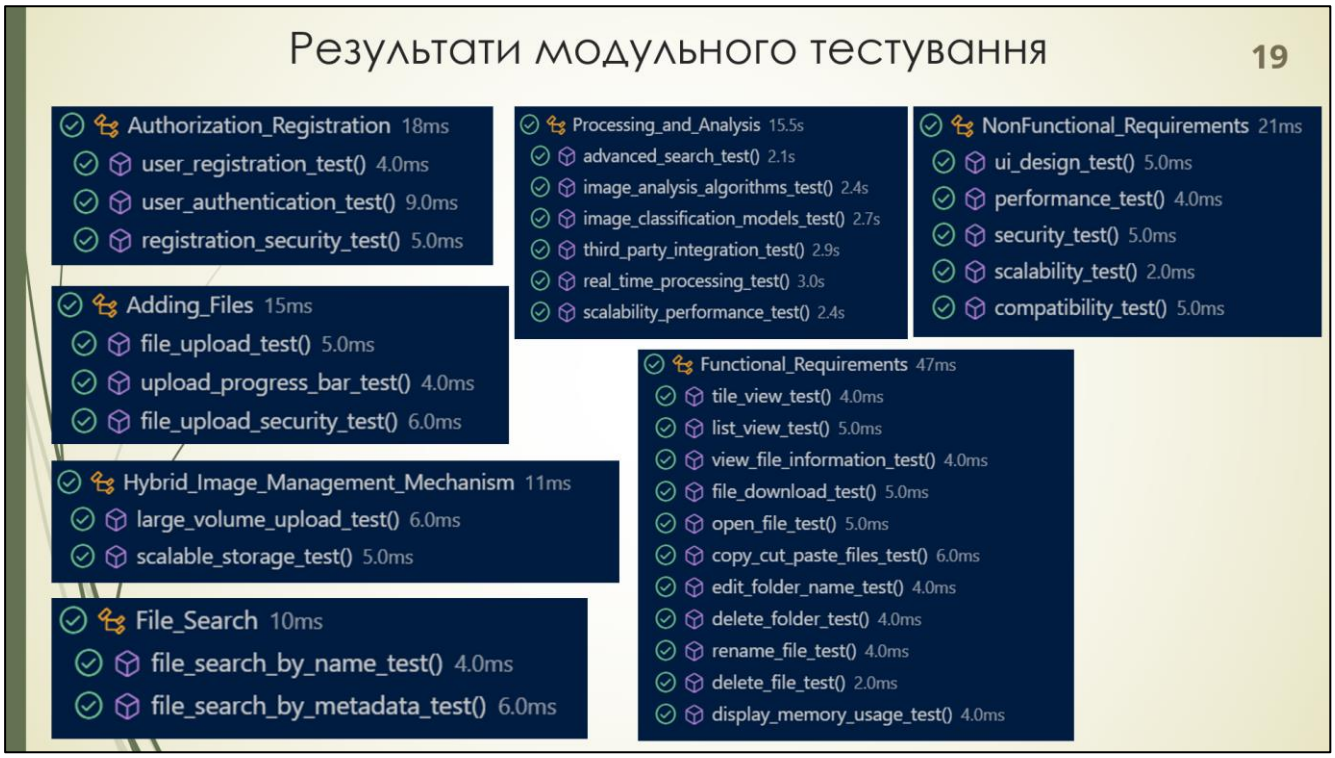


Рисунок Г.19 – Слайд 19

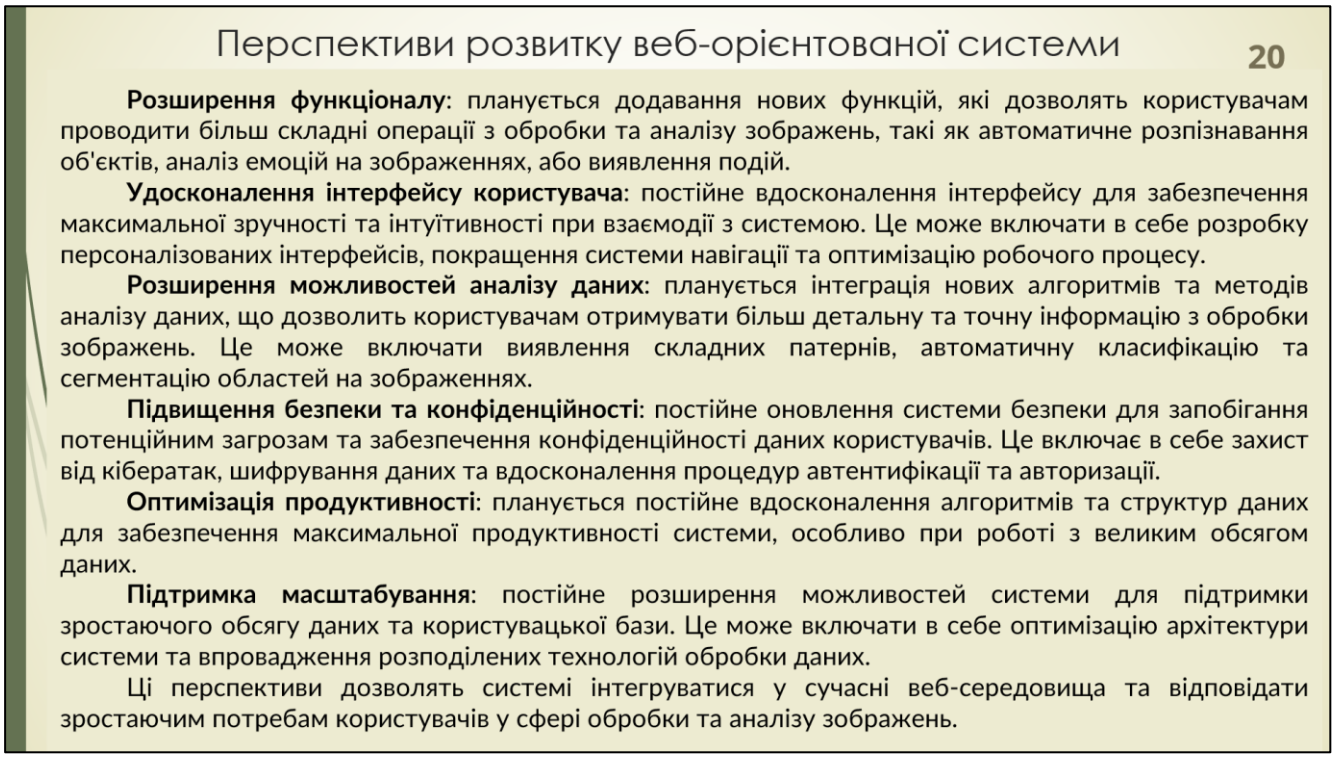


Рисунок Г.20 – Слайд 20

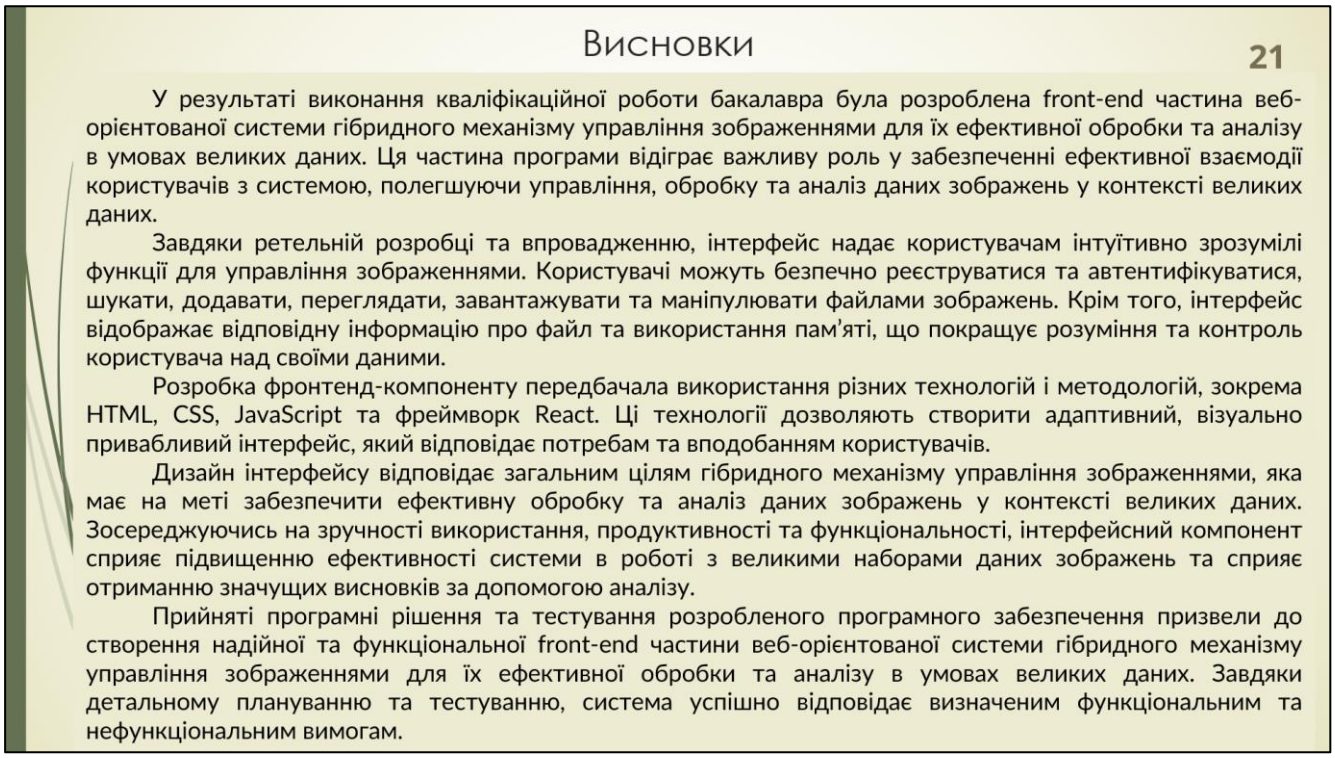


Рисунок Г.21 – Слайд 21