

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБСАЙТУ ДЛЯ КОМПРЕСІЇ ЗОБРАЖЕНЬ
DEVELOPMENT OF THE IMAGE COMPRESSION WEBSITE

Виконала: студентка 4 курсу, групи КНТ-111сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЖЕВЕДЬ Д.С.

(ПРИЗВИЩЕ та ініціали)

Керівник КАЗУРОВА А. Є.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОДКОВАЛІХІНА О.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

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

ЖЕВЕДЬ Діані Сергіївні

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебсайту для компресії зображень
Development of the Image Compression Website

керівник проєкту (роботи) к.т.н., доцент, КАЗУРОВА Аліна Євгенівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КАЗУРОВА А.Є., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання “15” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Діана ЖЕВЕДЬ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Аліна КАЗУРОВА
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
96 с., 31 рис., 3 табл., 3 дод., 15 джерел.

РЕАСТ, JAVASCRIPT, ВЕБЗАСТОСУНОК, ЗОБРАЖЕННЯ,
СТИСНЕННЯ ЗОБРАЖЕНЬ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для
компресії зображень.

Предмет дослідження – програмні засоби для обробки зображень.

Мета роботи – розробка програмного забезпечення для обробки даних
зображень.

Матеріали, методи та технічні засоби: мова програмування Javascript,
фреймворк React.

Результати. Розроблено програмне забезпечення вебсайту для компресії
зображень за допомогою мови програмування Javascript та фреймворку React.

Висновки. Виконано проєктування програмного забезпечення вебсайту
для для компресії зображень. Розроблено програмне забезпечення вебсайту
для для компресії зображень. Здійснено тестування розробленого програмного
забезпечення вебсайту для для компресії зображень.

Галузь використання – організації та користувачі, що займаються
обробкою зображень.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 96 pages, 31 figures, 3 tables, 3 appendixes, 15 sources.

REACT, JAVASCRIPT, WEB APPLICATION, IMAGE, IMAGE COMPRESSION, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of developing software for image compression.

The subject of research is image processing software.

The purpose of the work is to develop software for image data processing.

Materials, methods and technical means: Javascript programming language, React framework.

Results. Developed website software for image compression using Javascript programming language and React framework.

Conclusions. The design of website software for image compression has been completed. Developed website software for image compression. Testing of the developed website software for image compression was carried out.

Scope of use – organizations and users engaged in image processing.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Аналіз різних методів стиснення зображень	11
1.2 Аналіз програмного забезпечення для стиснення зображень	19
1.3 Висновки за розділом 1	30
2 Матеріали і методи	33
2.1 Вибір мови програмування	33
2.2 Вибір фреймворку для створення програмного забезпечення для стиснення зображень.....	35
2.3 Висновки за розділом 2	40
3 Опис програми	41
3.1 Структура програмного забезпечення для стиснення зображень	41
3.2 Функціонування програмного забезпечення для стиснення зображень	45
3.3 Розробка алгоритм стиснення зображень	47
3.4 Проєктування інтерфейсу програмного забезпечення для стиснення зображень.....	51
3.5 Висновки за розділом 3	52
4 Експлуатація, тестування та експериментальне дослідження програми.....	53
4.1 Призначення й умови застосування програми	53
4.2 Характеристики програми для стиснення зображень	54
4.3 Інструкція по експлуатації програми.....	54
4.3.1 Звернення до програми	54
4.3.2 Вхідні й вихідні дані	54
4.3.3 Повідомлення.....	55
4.4 Виконання програмного забезпечення для стиснення зображень.....	55
4.5 Тестування програмного забезпечення для стиснення зображень.....	58

4.6 Висновки за розділом 4	58
Висновки	60
Перелік джерел посилання	62
Додаток А Технічне завдання	64
Додаток Б Фрагмент тексту програми	68
Додаток В Слайди презентації	89

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JS – JavaScript;

MVC – Model-View-Controllе;

ОС – операційна система;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для компресії зображень визначається швидким розвитком інтернет-технологій та зростаючою важливістю швидкості завантаження веб-сторінок для користувачів. Зображення відіграють ключову роль у веб-дизайні, соціальних мережах, електронній комерції та інших онлайн-платформах, проте вони також становлять значну частину обсягу даних, що передається через мережу.

Зростання мобільного інтернету і швидкісного доступу до мережі ставить перед розробниками веб-сайтів вимогу забезпечити максимально ефективну оптимізацію зображень для забезпечення швидкого завантаження на різних пристроях і зв'язках. Повільне завантаження сторінок може призвести до втрати користувачів і зниження конверсії [1]-[5].

Таким чином, розробка програмного забезпечення для автоматичної компресії зображень для веб-сайтів стає актуальною задачею, спрямованою на поліпшення користувацького досвіду, збільшення швидкості завантаження та оптимізацію використання мережевого трафіку.

В сучасному цифровому світі, де зображення відіграють важливу роль у веб-дизайні, медіа і соціальних мережах, ефективна компресія зображень стає необхідною для забезпечення швидкої загрузки сторінок та оптимізації простору зберігання. Зображення займають значну частину обсягу даних веб-сайтів і, отже, впливають на їхню продуктивність та швидкість роботи.

Проте існують деякі потенційні недоліки вебсайтів, які спеціалізуються на компресії зображень, які важливо враховувати при розробці та модифікації існуючих вебсайтів, які надають послуги з обробки зображень, що дасть можливість покращити користувацький досвід та забезпечити ефективну роботу сайту. Деякі вебсайти, які надають сервіс з компресії зображень характеризуються поганою навігацією та незручним користувацьким інтерфейсом, що ускладнює процес роботи користувача з відповідним програмним забезпеченням [1]-[5].

Тому актуальною є розробка програмного забезпечення у вигляді вебсайту для компресії зображень. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для вебсайту для компресії зображень.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для компресії зображень;
- здійснити проєктування програмного забезпечення для вебсайту для компресії зображень;
- створити програмне забезпечення для вебсайту для компресії зображень;
- виконати тестування розробленого програмного забезпечення для вебсайту для компресії зображень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз різних методів стиснення зображень

Цифрове зображення — це двовимірний масив пікселів. Стиснення зображень зменшує обсяг пам'яті, необхідного для фотографій і відео, покращуючи таким чином продуктивність зберігання та передачі. Можливе стиснення зображень із втратами або без втрат. Стиснення без втрат передбачає стиснення даних таким чином, щоб їх можна було розпакувати до ідентичного відтворення оригіналу [1]-[3]. Однак у методах стиснення з втратами деякими дрібнішими деталями зображення можна пожертвувати, щоб заощадити трохи більше пропускну здатності або місця для зберігання.

Найпоширенішими процесами стиснення зображення є [1]-[3]:

- вказівка параметрів Rate (доступні біти) і Distortion (допустима помилка) для цільового зображення;
- класифікація візуальних даних відповідно до їх відповідності;
- розподіл доступного бітового бюджету між цими класами таким чином, щоб спотворення було мінімізоване;
- використовуючи інформацію про розподіл бітів, отриману на кроці 3, кількісно визначте кожен клас окремо;
- використовуючи ентропійний кодер, закодуйте кожен клас окремо та збережіть його у файлі;

Використовуючи двовимірний роздільний банк QMF, спектр вхідного сигналу розкладається на численні вузькосмугові зображення [1]-[3]. Щоб зменшити надлишкові біти в даних або зображеннях, Patel et al. (2016) представив техніку кодування Хаффмана, яка аналізує численні характеристики або специфікації, такі як «пікове співвідношення сигнал-шум (PSNR), середня квадратична помилка (MSE), біт на піксель (BPP) і коефіцієнт стиснення (CR)» [6].]. Кекре та ін. (2016) запропонували векторне квантування для стиснення зображення. VQ є багатовимірною версією скалярного квантування [7]. Чжан та ін. (2020) запропонували багатомасштабну

прогресивну статистичну модель на основі системи стиснення зображень без втрат. Запропонована статистична модель ефективно балансує попиксельну точність моделі та швидкість багатомасштабної моделі [1]-[3]. Мохаммед і Абу-Чаді (2011) досліджували методи стиснення зображення на основі кодування з усіканням блоків. Для порівняння було використано оригінальне кодування з усіканням блоку (BTC) і кодування з усіканням блоку в абсолютний момент (AMBTC) [1]-[3]. BTC розбиває оригінальне зображення на $n \times n$ підблоків, зменшуючи кількість рівнів сірого в кожному блоці. Саркар та ін. (2018) запропонував гібридну модель стиснення зображень із втратами, використовуючи кодування довжини серії та кодування Хаффмана, взявши приклад двох зображень, тобто `clock.tiff` і `man.tiff`. Інші параметри, такі як PSNR, MSE і структурна подібність, залишаються майже незмінними, а розмір накопичувача зменшується [1]-[3]. У таблиці 1 узагальнено численні підходи до стиснення зображень, які дослідники використовували в минулому.

Лі та ін. закодували тривимірний бінарний кубоїд за допомогою обрізаної згорткової мережі (тобто TCAE). (2018). Для представлення великого контексту, зберігаючи при цьому високу обчислювальну ефективність, проект повністю згорткової мережі може виграти від зменшення згортки та виконання ймовірного передбачення для всіх бітів за один прямий проход [1]-[3]. DeVore та ін. (1992) детально пояснив концепцію стиснення зображення через техніку кодування з перетворенням за допомогою математичних виразів [1]-[3]. Купер і Лоренц (2006) зосередилися на стисненні зображень за допомогою сингулярного розкладання (SVD). Це цінний пристрій для зменшення обсягу зберігання та транспортування даних. SVD — це матрична факторізація, яка дозволяє по-новому отримувати алгебраїчну та геометричну інформацію із зображення. Багато галузей прийняли SVD, включаючи стиснення даних, обробку сигналів і аналіз шаблонів [1]-[3]. З іншого боку, Telagarapu et al. (2011) стиснули три зображення за допомогою дискретного косинусного перетворення (DCT). DCT представляє вхідні дані як суму косинусних функцій зі змінними частотами та величинами. Найбільш

широко використовуваними ДКП є одновимірні та двовимірні [1]-[3]. Щоб вирішити проблему стиснення фокальної площини, Leon-Salas et al. (2007) розробили однокристальний підхід. У цьому дослідженні була застосована техніка прогнозного кодування в аналоговій області, а потім компактний ентропійний кодер на кристалі. Через можливість стиснення без втрат і простоту використання було обрано прогнозне кодування [1]-[3]. Чоудхурі та Хатун (2012) представили дискретне вейвлетне перетворення як новий підхід (DWT). DWT — це перетворення стиснення зображення з різною роздільною здатністю. Цей метод використовує кодування підсмуг для опису частоти-часу сигналу [16]. Sun та ін. (2008) запропонував алгоритм стиснення зображення майже без втрат, який поєднував категоризацію, приховування інформації та LZW. Подібні пікселі були отримані в тій же послідовності за допомогою класифікації та декомпозиції. Щоб зменшити обсяг даних, зображення маски було приховане в цих послідовностях. Потім для кодування послідовностей використовували LZW. Алгоритм був простим і мав вищий коефіцієнт стиснення, ніж попередні підходи, такі як LZW [1]-[3]. Стиснення фрактального зображення було визначено Liu et al. (2016). Техніка фрактального стиснення працює шляхом порівняння частин зображення з іншими частинами подібного зображення. Фрактальні алгоритми декодують зображення шляхом перетворення геометричних форм у математичні дані, відомі як «фрактальні коди» [1]-[3].

Існує два типи методів стиснення зображень: без втрат і стиснення з втратами. Декомпресія даних без втрат призводить до отримання зображення, ідентичного оригіналу. Ці процеси стирають дані, в результаті чого зображення не є ідентичним оригіналу.

Стиснення без втрат: реконструйоване зображення за допомогою техніки стиснення без втрат кількісно ідентичне оригінальному зображенню. За допомогою стиснення без втрат можна досягти лише обмеженого ступеня стиснення. Стиснення без втрат може зменшити розмір файлу вдвічі залежно від типу даних, що стискаються.

Як результат, стиснення без втрат є перевагою для доставки файлів через Інтернет, оскільки менші файли переміщуються швидше. Як правило, методи стиснення зображень без втрат розглядають зображення як набір пікселів, упорядкованих у порядку старшого рядка. Кожен піксель обробляється за допомогою двох різних процедур. Перший крок створює прогноз для числового значення наступного пікселя. Це часто поєднується з технікою визначення краю, яка намагається врахувати розриви інтенсивності. Різниця між очікуваними та фактичними значеннями пікселів на другій фазі кодується за допомогою ентропійного кодера та розподілу ймовірностей. На рисунку 1 показана блок-схема техніки стиснення без втрат. Існують різні типи методів стиснення без втрат, які докладно описані по черзі.

Як альтернатива стисненню з втратами, RLE використовує пару значень (довжина, значення) для заміни вихідних даних. Значення є унікальним, а довжина відображає кількість повторів. Ця базова техніка стиснення даних використовує цикли для зберігання стислих даних у менших фрагментах для полегшення пошуку пізніше. Прогони — це шаблони, у яких одне й те саме значення даних відображається в кількох елементах даних поспіль. Ці послідовності зберігаються як одне значення даних і підраховуються замість того, щоб зберегти початковий цикл. Розглянемо таку ілюстрацію: екран із суцільним білим текстом на основному чорному фоні. У вільній зоні буде багато довгих рядків білих пікселів і багато коротких рядків чорних пікселів.

Техніка статистичного кодування включає такі методи, як кодування Хаффмана, арифметичне кодування та кодування LZW [1]-[3]. (i) Кодування Хаффмана. У кодуванні Хаффмана використовується частота елемента даних (піксель у зображеннях). Мета полягає в тому, щоб кодувати дані частіше з меншою кількістю бітів. Коди зберігаються в книзі кодів для кожного зображення або колекції зображень. Щоб забезпечити можливість декодування, кодова книга та закодовані дані повинні передаватися за будь-яких обставин [6]. Щоб зашифрувати зображення, виконайте такі кроки:

спочатку розділіть зображення на блоки 8×8 , потім кожен блок є символом, який потрібно закодувати на другому кроці.

Обчисліть коди Хаффмана для набору блоків на наступному кроці та остаточно закодуйте блоки відповідно на останньому кроці. (ii) Арифметичне кодування Замість того, щоб кодувати кожен символ окремо, цей підхід кодує всю візуальну послідовність одним кодом. У результаті використовується асоціація сусідніх пікселів. Наступний принцип керує арифметичним кодуванням [1]-[3]. Символьний алфавіт є кінцевим, як і всі потенційні символьні послідовності заданої довжини; всі мислимі послідовності зліченні нескінченні; і ми можемо призначити унікальний підінтервал для кожного даного входу, оскільки діапазон $[0,1]$ включає необмежену кількість дійсних значень (послідовність символів). (iii) Кодування LZW Алгоритм LZW [1]-[3] базується на кількості послідовностей символів у закодованому рядку. Його теорія полягає в тому, щоб поступово розробляти словник шляхом заміни шаблонів індексним кодом. Для заповнення словника використовуються 256 значень таблиці ASCII. Для монохроматичних зображень (закодованих в 1 біт) кожен рядок порівнюється зі словником і додається, якщо його не знайдено. Тобто, якщо рядок ніколи не коротший за найбільше слово словника, він передається. Алгоритм перебудовує словник під час декодування; тому його не потрібно зберігати.

Іншим прикладом дослідження міжпіксельної надлишковості є техніка прогнозного кодування [1]-[3], яка використовує основне поняття кодування лише нової інформації в кожному пікселі. Різниця між фактичним і передбачуваним значенням пікселя часто використовується для визначення цієї додаткової інформації. Помилка передбачення обчислюється шляхом округлення результату предиктора до найближчого цілого числа та порівняння його з фактичним значенням пікселя. Для кодування цієї помилки (VLC) можна використовувати кодування змінної довжини. Парадигма, яка використовується для опису візуальних ефектів, є відмінною рисою методу. Зображення представлені у вигляді непричинних випадкових полів, у яких

інтенсивність кожного пікселя визначається інтенсивністю сайтів у всіх напрямках навколо нього.

Система стиснення з втратами може аналізувати дані про колір для діапазону пікселів і виявляти тонкі відхилення в значеннях кольору пікселя, які людське око/мозок не може по-іншому розрізнити. Комп'ютер може замінити інші пікселі меншого розміру, розбіжності в значеннях кольорів яких знаходяться в межах зору людини. Пікселі, які були точно оцінені, згодом відкидаються. Цей тип стиснення може призвести до значної економії розміру файлу, але складність алгоритму визначає перевагу зображення кінцевого продукту.

Він перетворює вісім піксельних значень у масив коефіцієнтів, розташованих ближче до верхнього лівого кута, який зазвичай містить основну частину інформації, необхідної для кількісного визначення та кодування зображення з невеликими спотвореннями сприйняття. Потім квантовані коефіцієнти використовуються для кодування бітового потоку зображення. Декодер використовує той самий процес, за винятком того, що етап «деквантування» лише наближає реальні значення коефіцієнта. Іншими словами, втрати квантователя на кроці кодера є незворотними.

Block Truncation Coding BTC — це техніка стиснення зображення у відтінках сірого [1]-[3]. Він використовує квантувач для зменшення кількості рівнів сірого всередині кожного блоку, зберігаючи середнє значення та стандартне відхилення. Хоча стиснення BTC використовувалося для стиснення кольору задовго до DXTC, воно є раннім предком популярної апаратної технології DirectX Texture Compression (DXTC). Якщо під час передачі або зберігання використовуються 8-бітні цілі значення, підблоки розміром 4×4 пікселя дозволяють стиснути близько 25%. Більші блоки дозволяють більше стискати, але через природу методу якість страждає зі збільшенням розміру блоку.

3.2.3. Кодування піддіапазону Він ділить діапазон частот сигналу на піддіапазони, а потім кодує кожен піддіапазон за допомогою кодера та швидкості передачі даних, які відповідають статистиці діапазону [1]-

[3]. Це пояснюється тим, що SBC допускає змінне призначення бітів між піддіапазонами та обмеження помилок кодування в піддіапазонах. Декодовані сигнали піддіапазону не дискретизуються та направляються через банк синтезуючих фільтрів перед відповідним підсумовуванням у декодері.

Методи векторного квантування (VQ) застосовують ідеї скалярного квантування до кількох вимірів. Цей метод передбачає створення векторного словника коду, який є набором векторів фіксованого розміру. Після цього зображення розбивається на вектори зображення, або частини, що не перекриваються [1]-[3]. Потім для кожного вектора зображення виявляється найближчий відповідний вектор словника, а його індекс у словнику використовується як кодування вихідного вектора зображення. Схеми кодування на основі VQ особливо добре підходять для мультимедійних програм завдяки їхнім можливостям пошуку на стороні декодера.

Техніка фрактального стиснення працює шляхом порівняння областей фотографій з іншими зображеннями. Алгоритми перетворюють ці фрагменти або геометричні фігури у «фрактальні коди», які можна використовувати для копіювання закодованого зображення. При перетворенні на фрактальне кодування зображення втрачає залежність від роздільної здатності. Зображення можна реконструювати відповідно до будь-якого розміру екрана без артефактів піксельного стиснення або втрати якості [1]-[3].

Розкладання сингулярних значень (SVD). Лінійна алгебра широко використовується для стиснення даних. У сучасному суспільстві необхідність зменшити кількість цифрових даних, що зберігаються та передаються, стає все більш важливою. Сингулярна декомпозиція [1]-[3] є цінною технікою для мінімізації зберігання та транспортування даних. SVD — це матрична факторизація, яка дозволяє по-новому отримувати алгебраїчну та геометричну інформацію із зображення. Багато галузей прийняли SVD, включаючи стиснення даних, обробку сигналів і аналіз шаблонів [1]-[3]. Мета SVD — знайти найкраще наближення початкових точок даних у найменшій кількості вимірів. Цього можна досягти, визначивши регіони з найвищим ступенем

мінливості. SVD використовується для скорочення великого, сильно змінюваного набору точок даних до простору нижчого виміру, який більш чітко відображає підструктуру вихідних даних і ранжує їх від найбільшої до найменшої дисперсії. Використовуючи підхід SVD, у найбільш мінливій області та зменшує її розмір. Іншими словами, SVD — це підхід до зменшення даних. Численні видатні математики вважають SVD важливою темою лінійної алгебри. Окрім зменшення зображення, SVD корисний у багатьох практичних і теоретичних контекстах. Перевагою SVD є те, що його можна використовувати для будь-якої дійсної (m,n) матриці. Він ділить A на три матриці U , S і V таким чином, що $A = USV^T$. S є діагональною матрицею, тоді як U і V є ортогональними матрицями.

Дискретне косинусне перетворення (DCT). DCT представляє точки вхідних даних як суму косинусних функцій з різними частотами та величинами. Найбільш поширені DCT є одновимірними та двовимірними. Joint Photographic Expert Group (JPEG) була заснована в 1992 році на DCT [14]. Це поширений метод стиснення. Це метод JPEG: початковий розмір блоку 8×8 . По-друге, DCT застосовується до кожного блоку в орієнтації зліва направо та зверху вниз. Потім, щоб обмежити обсяг даних у пам'яті, використовується квантування, і дані зберігаються в точний спосіб. 3.2.8. Дискретне вейвлетне перетворення (DWT) Підхід DWT — це метод перетворення з різною роздільною здатністю, який часто використовується для збільшення коефіцієнта стиснення зображень.

Цей метод використовує кодування підсмуг для представлення часових і частотних компонентів сигналу. У DWT картина представлена набором вейвлет-функцій (також званих вейвлетами) з різними розташуваннями та масштабом [1]-[3]. Коефіцієнти високої частоти (детальний) і низької частоти (приблизний) перетворення відображають дискретні дані. Спочатку розділено 3232 блоки. Він розділяє дані на коефіцієнти наближення та деталізації. У випрямлених матрицях коефіцієнти позначені. Потім LL перекладається на другий рівень. Ступінь стиснення отримується множенням коефіцієнтів на

заздалегідь визначений коефіцієнт масштабування. 4. Висновок: оскільки існує компроміс між ступенем стиснення та піковим SNR у стисненні зображення, розробка більш ефективної методики стиснення-декомпресії залишається складним завданням у цій галузі. Незважаючи на те, що в цій галузі було проведено значні дослідження, із постійно зростаючим попитом на методи стиснення з низькою швидкістю передачі даних, все ще є місце для нових підходів і розвитку більш ефективних алгоритмів у рамках існуючих методів. Огляд показує, що ця тема й надалі буде викликати інтерес науковців у наступні роки. У результаті ми виявлено, що стиснення з втратами забезпечує вищий коефіцієнт стиснення, ніж стиснення без втрат. Стиснення без втрати даних є оптимальним для стиснення тексту. Якщо бітова глибина зображення перевищує 0,5 bpp, усі алгоритми стиснення з втратами мають високий коефіцієнт стиснення. Крім того, стиснення зображення сильно залежить від якості зображення.

1.2 Аналіз програмного забезпечення для стиснення зображень

Проаналізуємо програмне забезпечення для компресії зображень. Онлайн-інструменти для оптимізації зображень допомагають стискати зображення або змінювати тип файлу без відчутної втрати якості.

Compress Now — чудовий інструмент для оптимізації зображень, який дозволяє стискати файли зображень трьох найпоширеніших типів: JPG, GIF і PNG [4]-[7] (рисунок 1.1).

Він має кілька інших корисних функцій, у тому числі той факт, що він дозволяє вибрати ступінь, до якого ви бажаєте стиснути завантажене зображення. Максимальний розмір файлу, який можна завантажити, становить 9 МБ = 9000 КБ.

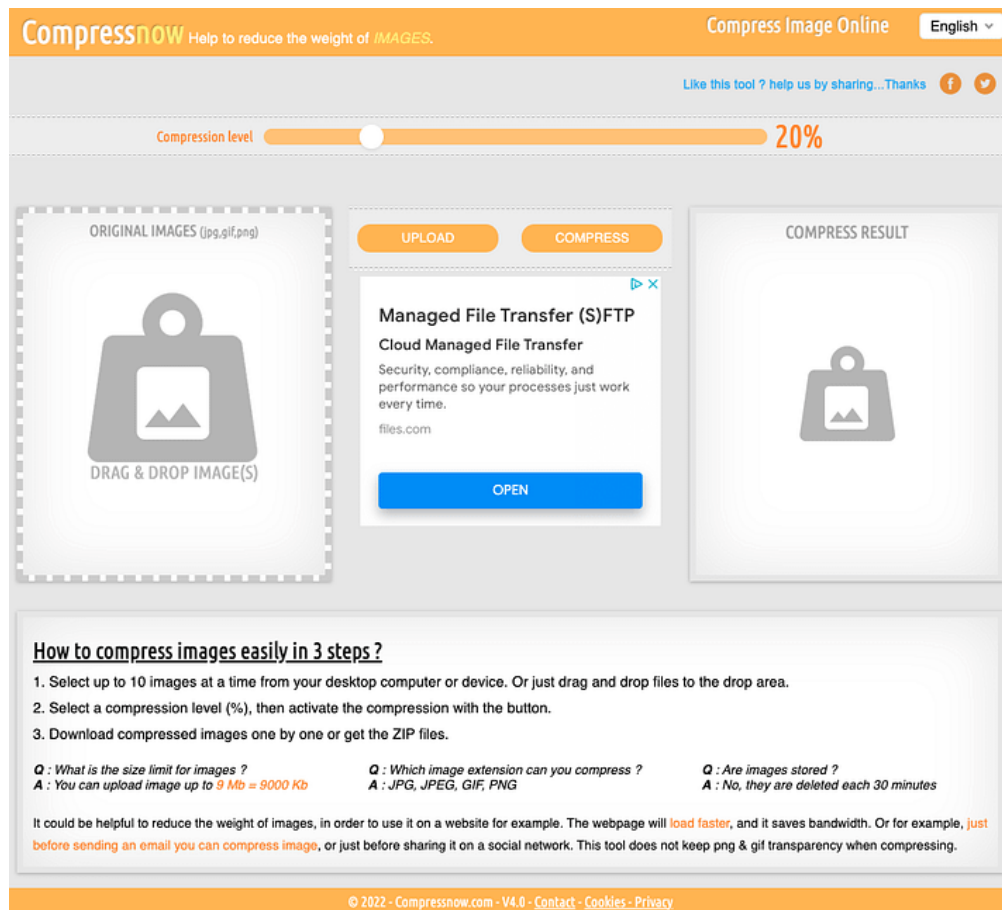


Рисунок 1.1 – Програмне забезпечення Compress Now

Під час тестування встановлена ступінь стиснення на 65%, і неозброєним людським можна відрізнити оригінал від набагато легшої стисненої версії. Стиснуте зображення було трохи менш блискучим.

Єдиним недоліком цього онлайн-оптимізатора зображень є те, що хоча можна завантажувати всі три типи файлів зображень (JPEG, PNG, GIF) в інструмент, він видасть лише JPEG на іншому кінці.

Компресія зображення в Compress Now:

- збереження файлу зображення JPEG: 99,97%;
- збереження файлу зображення PNG: 99,98% (вихідний файл – JPEG).

JPEG Optimizer (рисунок 1.2) є гарним рішенням, якщо потрібно, стиснути зображення JPEG, оскільки, як випливає з назви, немає підтримки для файлів PNG або GIF.

Більшості підійде стиснення за замовчуванням 65%, хоча веб-сайт може стискати зображення в діапазоні 0-99%. Використовуючи лише налаштування за замовчуванням, можна стиснути зображення майже вдвічі від початкового розміру, і якість помітно не зміниться.

Можна встановити максимальний розмір зображення (за замовчуванням 4 МБ) і змінити його розміри. Після того, як інструмент стисне зображення можна завантажити його та використовувати будь-де.

Компресія зображення в JPEG Optimizer – збереження файлів зображень JPEG – 95,39% [4]-[7].

Рисунок 1.2 – Програмне забезпечення JPEG Optimizer

Програмне забезпечення Tiny PNG (рисунок 1.3), з легкістю обробляє стиснення форматів зображень JPEG і PNG, дозволяючи масово завантажувати до 20 зображень за один раз. Розмір кожного файлу може досягати 5 МБ. Для більших файлів вам доведеться платити [4]-[7].

Після стиснення зображення можна завантажити на комп'ютер або зберегти в Dropbox. Tiny PNG також полегшує роботу для користувачів WordPress за допомогою безкоштовного плагіна. Щоб порівняти плагіни оптимізації зображень для WordPress.

Компресія зображення в Tiny PNG:

- збереження файлів зображень JPEG: 92%;
- збереження файлу зображення PNG: 83%.



Рисунок 1.3 – Програмне забезпечення Tiny PNG

Програмне забезпечення Optimizilla (рисунок 1.4) – ще один простий у використанні інструмент оптимізації зображень, який стискає формати JPEG, GIF і PNG. В Optimizilla можна завантажити до 20 зображень. Інструмент додає кожне зображення до черги та автоматично стискає його за допомогою оптимізації з втратами [4]-[7].

Приємно, що ви можна бачити всі стиснуті зображення у слайдері та перевіряти їхній статус під час процесу оптимізації. Можна побачити, на який відсоток було стиснуто кожне зображення, і видалити зображення, які можна зберігати.

Ще одна функція полягає в тому, що якщо не влаштовує рівень стиснення за замовчуванням, можна клацнути будь-яке зображення на

повзунку, а потім прокрутити вниз, щоб переглянути та змінити його якість вручну.

Компресія зображення в Optimizilla:

- збереження файлів зображень JPEG: 91%;
- збереження файлу зображення PNG: 79%.

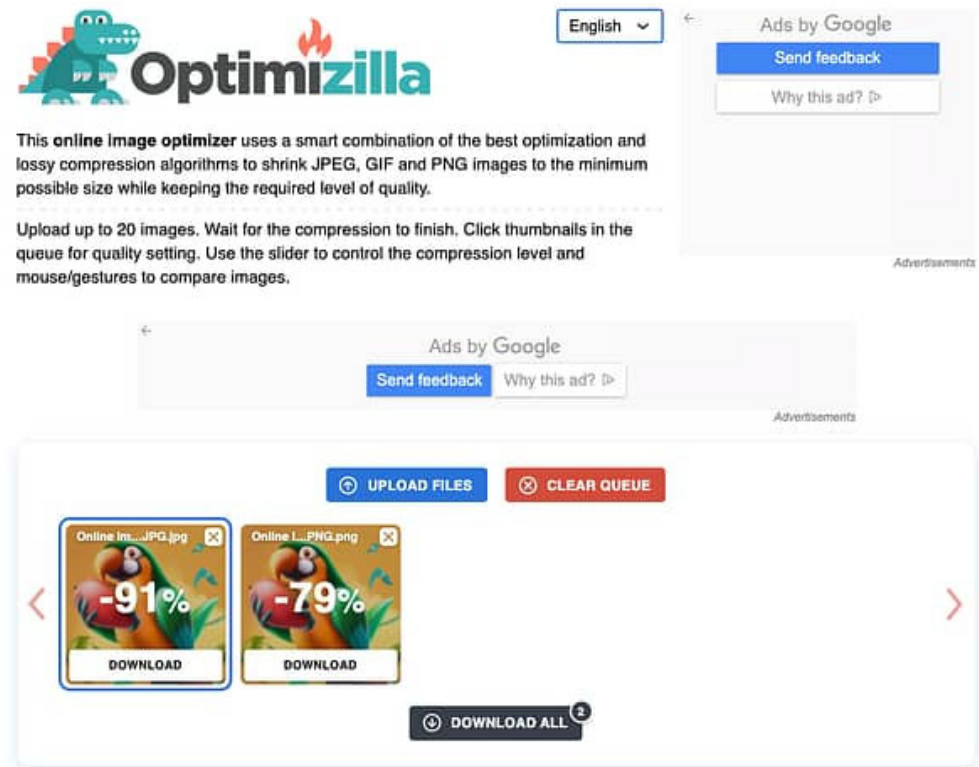


Рисунок 1.4 – Програмне забезпечення Optimizilla

Програмне забезпечення JPEG.io (рисунок 1.5) дозволяє конвертувати зображення будь-якого формату в JPEG. Він обробляє більше типів зображень, ніж більшість онлайн-інструментів для оптимізації зображень. Щоб використовувати його, ви перетягуєте та завантажуєте файли PNG, GIF, SVG або навіть PSD на їхній веб-сайт. Спочатку інструмент перетворює зображення у JPEG, а потім стискає їх, перш ніж повернути зображення [4]-[7].

Інструмент також може отримувати файли зображень із вашого Google Drive або Dropbox і повертати вам оптимізовані файли JPEG у архіві zip.

Компресія зображення в JPEG.io:

- збереження файлу зображення JPEG: 79,57%;
- збереження у файлі зображення PNG: 86,98% (вихід JPEG).

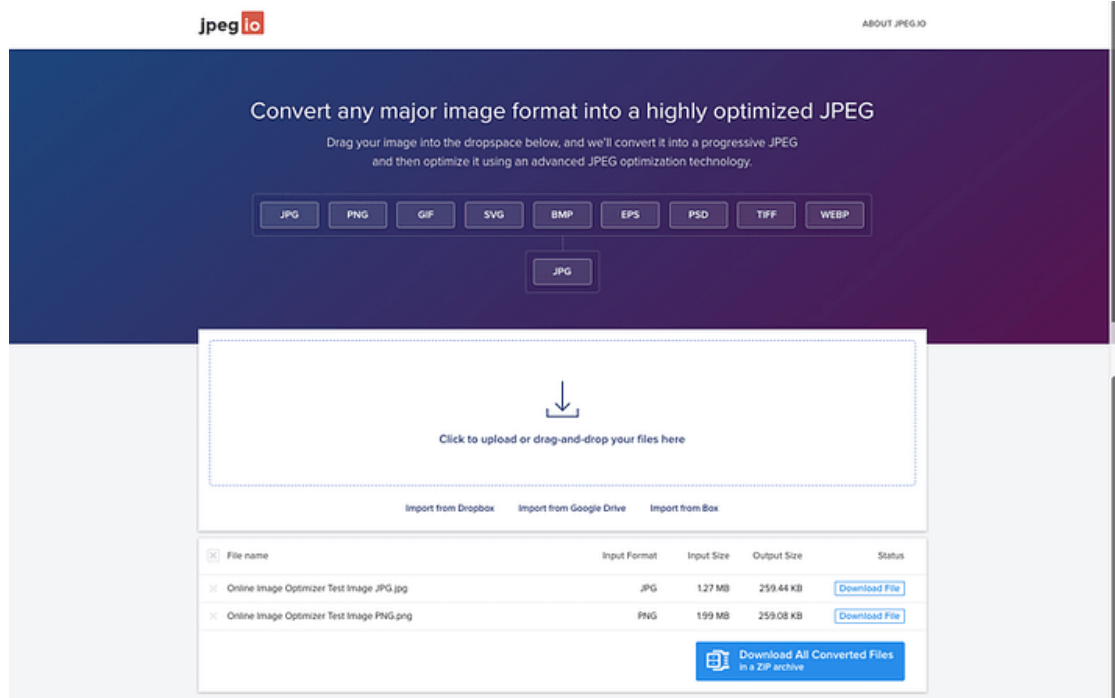


Рисунок 1.5 – Програмне забезпечення JPEG.io

Програмне забезпечення ImageRecycle (рисунок 1.6) стискати зображення різних форматів. Не лише PNG чи JPEG, цей інструмент підтримує навіть GIF та PDF. Можете перетягнути зображення на веб-сайт і завантажити оптимізовані зображення одним кліком [4]-[7].

Однією з корисних рис ImageRecycle є те, що він надає вам безкоштовний звіт про ваш веб-сайт із детальною інформацією про зображення, які потребують оптимізації. Просто введіть URL-адресу свого веб-сайту, щоб отримати цей звіт.

Щоб отримати доступ до таких функцій, як URL-адреса чи оптимізатор сторінок, вам доведеться вибрати їхні плани одноразової оплати. Ви можете вибрати з різних цінових категорій від 10 доларів США (10 000 зображень), 20 доларів США (30 000 зображень) і 50 доларів США (100 000 зображень).

Існує 15-денна безкоштовна пробна версія з обмеженням у 100 МБ, щоб допомогти вам прийняти рішення. Після оплати ви можете використовувати свою квоту протягом року на необмеженій кількості веб-сайтів із будь-якою кількістю субрахунків. Резервне копіювання та відновлення протягом місяця включено. Користувачі WordPress, Magento та Joomla можуть покладатися на розширення замість того, щоб щоразу відвідувати веб-сайт, а користувачі Shopify отримують ексклюзивний додаток.

Компресія зображення в ImageRecycle:

- збереження файлу зображення JPEG: 92,78%;
- збереження файлу зображення PNG: 80,41%.

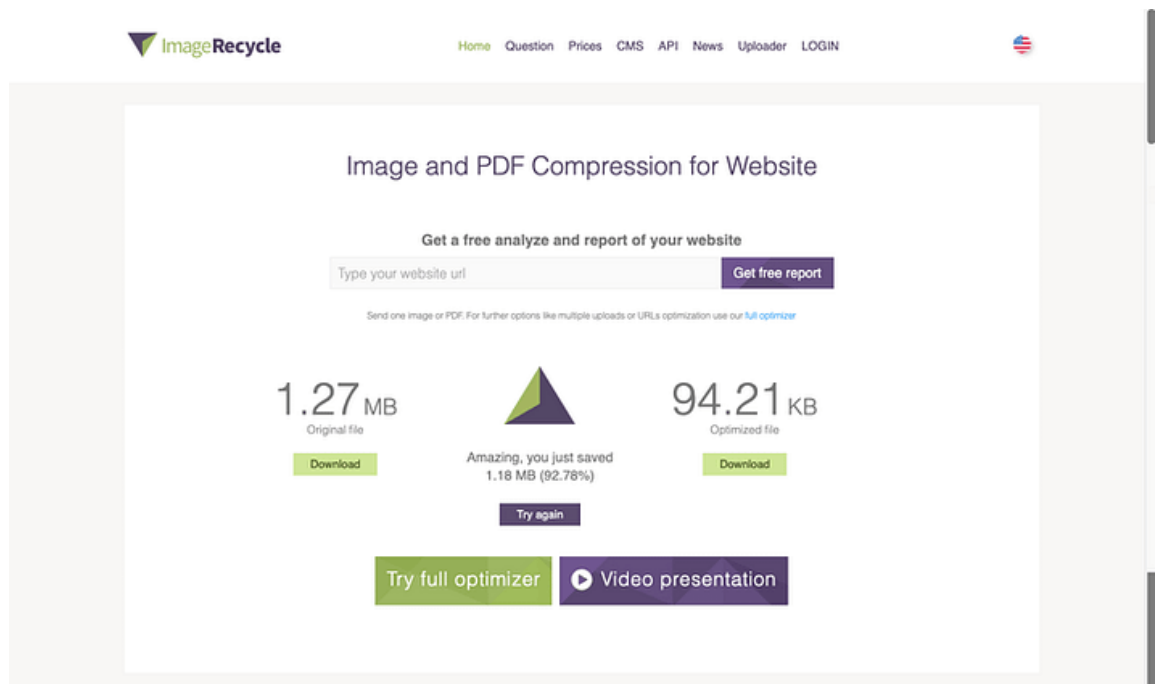


Рисунок 1.6 – Програмне забезпечення ImageRecycle

Програмне забезпечення Compressor.io (рисунок 1.7) надає високий рівень стиснення, який може заощадити сотні Кб. Compressor.io може зменшити розміри файлів майже на 90%. Він стискає зображення JPEG і PNG без втрати пікселів і стискає з втратами файли інших типів. Однак розмір файлу не може перевищувати 10 Мб [4]-[7].

Слайдер дозволяє порівнювати оригінальні та оптимізовані зображення поруч. Можете завантажити файл і зберегти його на Google Drive або Dropbox.

Компресія зображення в Compressor.io:

- збереження файлів зображень JPEG: 95%;
- збереження файлу зображення PNG: 78%.

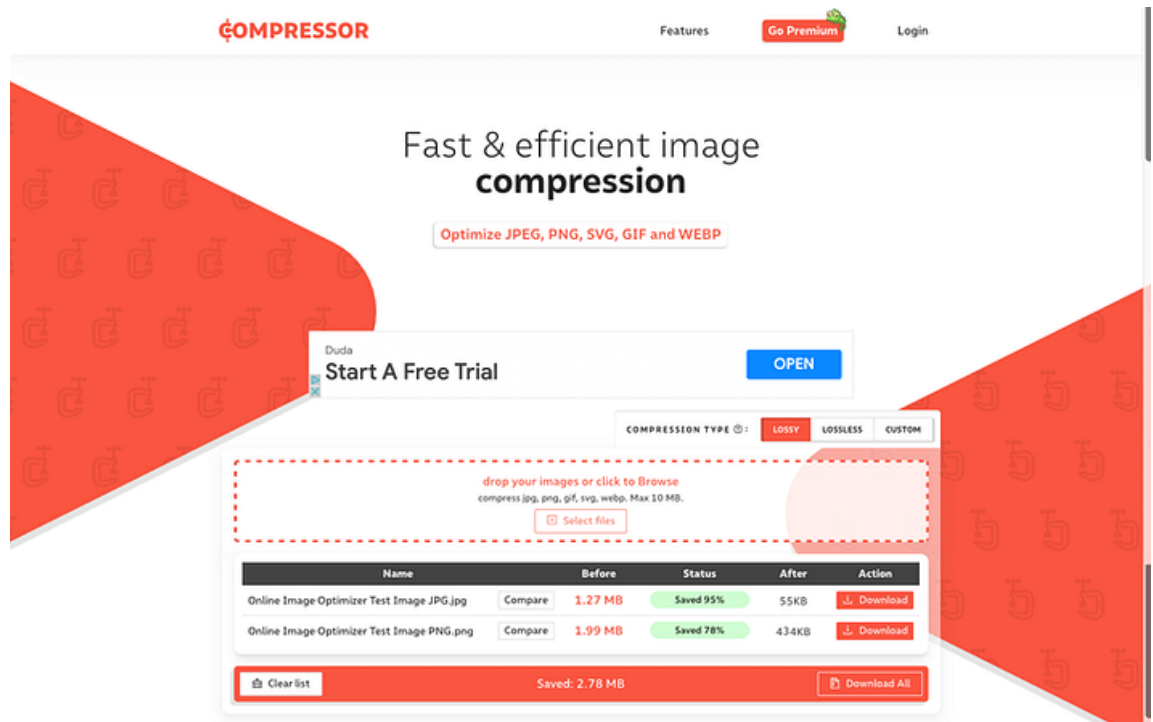


Рисунок 1.7 – Програмне забезпечення Compressor.io

Програмне забезпечення Ezgif (рисунок 1.8) – це універсальний онлайн-оптимізатор зображень. Його можна використовувати його для роботи з файлами GIF, PNG і JPG. У відділі GIF Ezgif піднімає планку на новий рівень. Це дозволяє багато працювати з GIF-файлами, зокрема створювати, змінювати розмір, обрізати чи оптимізувати їх. Можна створювати GIF-файли за допомогою JPEG, PNG та інших зображень і завантажувати до 2000 файлів, хоча загальний розмір обмежений у 100 Мб [4]-[7].

Крім того, оптимізація PNG і JPG також досить ефективна. Перед оптимізацією можна внести деякі прості зміни у зображення, наприклад

повернути, обрізати, додати накладення тощо. Що стосується самої оптимізації, то вона непогана.

Компресія зображення в Ezgif :

- збереження файлів зображень JPEG: 92,7%;
- збереження файлу зображення PNG: 24,5.

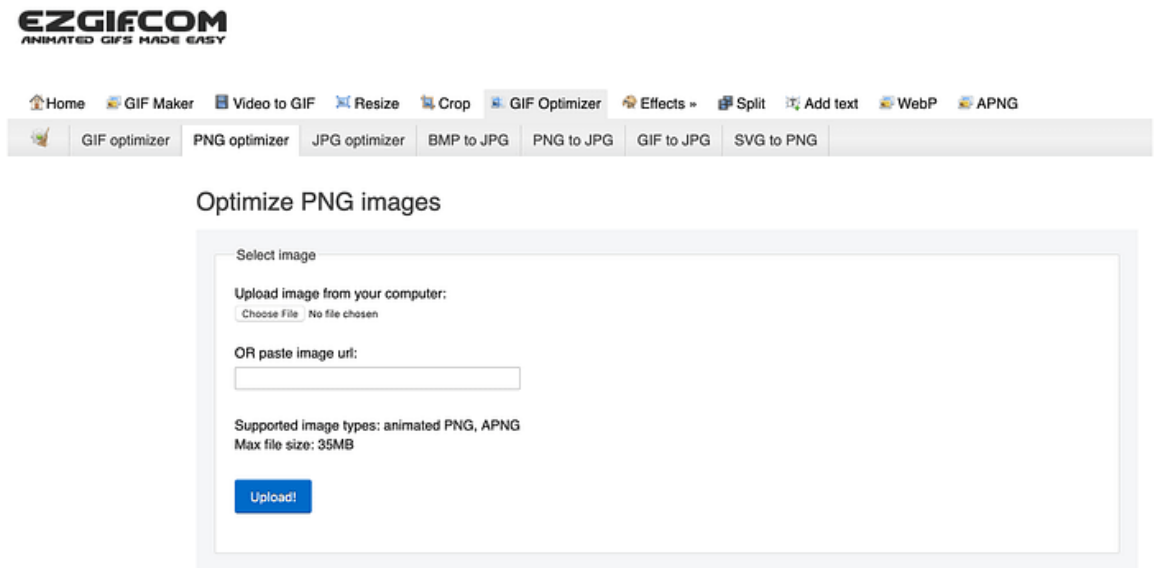


Рисунок 1.8 – Програмне забезпечення Ezgif

Програмне забезпечення Dynamic Drive Optimizer (рисунок 1.9) має можливість контролювати вихідний тип файлу. Іншими словами, можна почати з JPG і закінчити стисненим PNG або будь-якою іншою комбінацією входу і виходу [4]-[7].

Інша унікальна особливість Dynamic Drive полягає в тому, що коли він надає результати, він надає не лише один результат. Він генерує серію з дев'яти зображень різного рівня якості, щоб можна було швидко прокручувати та вибирати версію, яка найбільше підходить.

Під час сканування результатів JPEG єдиним, що помітно погіршувало якість, був останній варіант.

З файлами PNG найперша версія після оригіналу зберегла 256 кольорів зі зменшенням розміру файлу на 7%.

Компресія зображення в Dynamic Drive Optimizer:

- збереження файлу зображення JPEG: 93% (за рівня якості 60), максимум 97% (на рівні якості 10);
- збереження файлу зображення PNG: 77% (при 64 кольорах), максимум 92% (при 4 кольорах, але це майже не можна було використовувати).

Online Image Optimizer- GIF, JPG, and PNG

Image Optimizer lets you easily optimize your gifs, animated gifs, jpgs, and pngs, so they load as fast as possible on your site. Furthermore, you can easily convert from one image type to another. **Upload Size limit: 2.86 MB**

Enter the url of an image:

or upload one from your computer: No file chosen

File must be gif, jpg, or png. Max file size: 2.86 MB

convert to: ☐ show all results

Рисунок 1.9 – Програмне забезпечення Dynamic Drive Optimizer

Порівняльну характеристику програмного забезпечення для стиснення зображень наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для стиснення зображень

Критерій порівняння	Compress Now	JPEG Optimizer	Tiny PNG	Optimizilla	JPEG.io	ImageRecycle]	Compressor.io	Ezgif	Dynamic Drive
Зручність використання	+—	+	+—	+—	+	+—	+—	+	+—
Відсоток зменшення у файлі jpg	99,9%	95,3%	92%	91%	79,5%	92,7%	95%	92,7%	93
Відсоткове зменшення у файлі png	99,9%	-	83%	79%	86,9%	80,4%	78%	24,5%	77%
Можливість завантаження різних типів файлів для стиснення	+	+	+—	+	+	+	+	+	+—
Можливість отримання файлів різних типів в якості результату	-	+	—	+	+	+	+	+	+—
Різноманітність налаштувань перед початком стиснення	+—	+	+	+—	+	+	+—	+	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для стиснення зображень. Проте деякі програмні засоби для стиснення зображень можуть бути менш гнучкими у відношенні до специфічних потреб конкретного користувача. Деякі з розглянутих сервісів показали здатністю стиснути зображення без втрат якості, забезпечуючи високу рівень деталізації. Інші сервіси можуть підтримувати різні формати файлів, включаючи JPEG, PNG, GIF та інші, що важливо в залежності від вашої потреби. Також, важливим є зручність інтерфейсу та можливість використання сервісу як онлайн, так і через API для автоматизованих завдань.

Деякі сервіси можуть надавати додаткові можливості, такі як оптимізація для веб-сайтів, пакетне стиснення та інші.

Важливим чинником вибору програмного забезпечення для стиснення зображень є вартість сервісу та можливі обмеження, такі як обмеження щодо розміру файлу або кількості зображень на операцію.

Тому актуальною є розробка програмного забезпечення для компресії зображень. При розробці програмного забезпечення для компресії зображень необхідно забезпечити такі функціональні вимоги:

- стиснення зображень без втрат якості;
- підтримка різних форматів файлів;
- зручний інтерфейс користувача;
- можливість обробляти одночасно кілька зображень;
- можливість аналізувати та порівнювати якість зображення до та після стиснення;
- можливість автоматично оптимізувати зображення для веб-сайтів.

1.3 Висновки за розділом 1

Аналіз різних методів стиснення зображень показав, що існують різні підходи до оптимізації розміру файлів зображень. Методи можуть варіюватися від безвтратних до втратних, з різними алгоритмами стиснення та рівнем збереження деталей. Оцінка ефективності методів варіюється в залежності від типу зображення, потреб користувача та вимог до якості. Завдяки ретельному аналізу різних методів стиснення зображень, стало очевидним, що кожен метод має свої переваги та обмеження. Наприклад, безвтратні методи, які забезпечують високу якість зображення, можуть бути більш ефективними для фотографій, тоді як втратні методи можуть бути більш підходящими для картинок або іконок.

Аналіз програмного забезпечення для стиснення зображень виявив, що існує різноманітність програм, що пропонують різні функції для оптимізації файлів зображень. Визначені функції включають стиснення без втрат, підтримку різних форматів файлів, зручний інтерфейс користувача, пакетну обробку, можливість інтеграції та інші. Важливо враховувати вимоги та потреби користувача при виборі програмного забезпечення для оптимізації зображень. Також було виявлено, що багато програм пропонують різні рівні автоматизації та налаштування, що дозволяє користувачам підлаштувати процес оптимізації під свої потреби. Важливо, також, звернути увагу на швидкість та ефективність роботи програмного забезпечення для стиснення зображень, оскільки це може вплинути на продуктивність користувача та час, потрібний для завершення завдань.

Одним із ключових факторів успішної реалізації оптимізації зображень є здатність програмного забезпечення ефективно працювати з різними типами файлів та форматів, забезпечуючи високу якість стиснення для кожного з них.

Під час аналізу було виявлено, що інтеграція з іншими інструментами та сервісами, такими як веб-сайти чи платформи для розробників, може бути ключовою функцією для програмного забезпечення для стиснення зображень.

Також було виявлено, що підтримка різних мов програмування та інтерфейсів додатків, може значно полегшити інтеграцію та роботу з програмним забезпеченням для стиснення зображень.

Отже, аналіз даної предметної області підкреслив різноманітність методів та програмного забезпечення для стиснення зображень, а також важливість вибору оптимального рішення. На основі цього можна зробити висновок, що вибір програмного забезпечення для стиснення зображень повинен бути зроблений на основі конкретних потреб та вимог користувача, враховуючи різноманітність функцій та можливостей кожного інструменту.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для стиснення зображень. Проте деякі програмні засоби для стиснення зображень є менш гнучкими у відношенні до

специфічних потреб конкретного користувача. З метою специфічних користувацьких потреб важливо вибирати надійне програмне забезпечення або розробляти власне програмне забезпечення. Тому актуальною є розробка програмного забезпечення для стиснення зображень.

Сформульовано функціональні вимоги до програмного забезпечення для стиснення зображень.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для стиснення зображень було використано мову програмування JavaScript [8]-[10].

Мова програмування JavaScript, яку часто називають JS, є фундаментальною мовою програмування, яка змінила світ веб-розробки.

Мова програмування JavaScript була створений Бренданом Айхом, коли він працював у Netscape Communications Corporation у 1995 році. Спочатку вона називався «Mocha», а пізніше «LiveScript», але з часом стала відома як JavaScript. Її основною метою було надати інтерактивність веб-сторінкам, дозволяючи їм реагувати на дії користувача.

Ключові особливості JavaScript:

- мова високого рівня: JavaScript — це інтерпретована мова програмування високого рівня, що означає, що розробники можуть писати код, наближений до людської мови;

- універсальність: JavaScript не обмежується веб-розробкою. Його можна використовувати для сценаріїв на стороні сервера (Node.js), настільних програм (Electron) і навіть розробки ігор (Phaser);

- динамічний тип: JavaScript використовує динамічний тип, що дозволяє змінним змінювати свій тип даних під час виконання;

- керований подіями: керований подіями характер JavaScript дозволяє йому реагувати на дії користувача та маніпулювати об'єктною моделлю документа (DOM) веб-сторінок;

- кросплатформенність: вона працює на всіх основних браузерах і платформах, що робить її однією з найдоступніших мов програмування.

JavaScript відіграє вирішальну роль у сучасній веб-розробці, насамперед у таких ключових сферах:

- сценарії на стороні клієнта: JavaScript забезпечує динамічний вміст, інтерактивні форми та оновлення в реальному часі у веб-додатках. Такі

бібліотеки, як jQuery, і такі фреймворки, як React і Angular, спрощують розробку складних інтерфейсних програм;

- маніпуляції DOM: JavaScript використовується для маніпулювання DOM, дозволяючи розробникам створювати адаптивні та інтерактивні веб-сторінки;

- AJAX (асинхронний JavaScript і XML): AJAX дозволяє веб-сторінкам надсилати й отримувати дані з сервера без оновлення всієї сторінки, забезпечуючи зручнішу взаємодію з користувачем;

- інтеграція API: JavaScript може взаємодіяти з різними API, дозволяючи розробникам отримувати доступ до зовнішніх служб і даних;

- односторінкові додатки (SPA): фреймворки, такі як React, Angular і Vue.js, полегшують розробку SPA, які надають користувачам більше схожий на додаток досвід [8]-[10].

Незважаючи на те, що JavaScript пропонує величезну потужність, він також створює проблеми, такі як проблеми сумісності браузера, проблеми з безпекою та оптимізація продуктивності. Щоб подолати ці труднощі, розробники повинні дотримуватися передових практик:

- модулярність: організація коду у багаторазові модулі для зручності обслуговування;

- сумісність із різними браузерами: тестування та адаптація коду для узгодженої роботи в різних браузерах;

- безпека;

- оптимізація продуктивності: оптимізація коду для збільшення швидкості, зменшивши непотрібні операції та використовуючи кешування браузера.

JavaScript є основою сучасної веб-розробки, що дозволяє розробникам створювати динамічні, інтерактивні та зручні веб-додатки. Його універсальність і широкі можливості застосування роблять його важливою компетенцією для розробників у всьому світі. Оскільки веб-технології

продовжують розвиватися, JavaScript залишається життєво важливим інструментом для формування цифрового ландшафту [8]-[10].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для стиснення зображень наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для стиснення зображень

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	Ruby
Зручність кодування	+	+—	+
Багатозадачність та паралельність	+	+	+—
Динамічна типізація	+	+	—
Зручність для створення ПЗ для стиснення зображень	+	—	—
Поширеність та спільнота розробників	+	+	+—

Отже, для реалізації програмного забезпечення для стиснення зображень обрано мову програмування JavaScript, яка є мовою загального призначення високого рівня та характеризується своєю зрозумілістю і простотою синтаксису.

2.2 Вибір фреймворку для створення програмного забезпечення для стиснення зображень

В якості фреймворку для розробки програмного забезпечення для стиснення зображень було обрано середовище React [11]-[13].

React JS — це бібліотека з відкритим вихідним кодом для JavaScript, яка була офіційно випущена в 2013 році. Джордан Волке, який створив React.JS, також випустив її прототип під назвою FaxJS, випущений у 2011 році.

Бібліотека React.JS широко використовується при розробці інтерфейсів користувача та односторінкових додатків. Більше 40% розробників використовують React.JS і вважають його найкращою бібліотекою для розробки веб-додатків [11]-[13].

Ось список переваг, пов'язаних з використанням React.JS:

- віртуальний DOM відповідає за забезпечення швидкості та продуктивності програм React.JS. Завдяки віртуальному DOM розробники можуть створювати швидкі програми за сучасними стандартами. DOM — це деревовидна структура документів XHTML, XML або HTML. Для React.JS потрібно більше, ніж DOM. З DOM має бути щось інше, щоб він працював швидше та краще. Використання ідеї віртуальної версії DOM було найкращим для React. За допомогою віртуального DOM розробники можуть навіть оновлювати будь-які компоненти програми, і ці зміни не впливають на роботу програми. Virtual DOM — це віртуальна версія звичайного DOM. Перевага використання віртуального DOM полягає в тому, що для оновлення потрібно менше часу, ніж для стандартного DOM;

- компоненти можна вкладати в інші компоненти, що дозволяє створювати складні програми з простих будівельних блоків. React.JS використовує механізм на основі віртуальної DOM для заповнення даними в HTML DOM. Віртуальний DOM працює швидко, змінюючи лише окремі елементи DOM замість перезавантаження повного DOM;

- спільнота. Оскільки React.JS є відкритим кодом, він має потужну підтримку спільноти. Якщо ви зіткнетеся з будь-якою помилкою чи проблемою, спільнота готова надати вам бажану допомогу, коли це буде потрібно. Багато посібників, відео та практичних статей доступні безкоштовно для всіх. Близько 89 000 розробників використовують React.JS;

– SEO-зручність. SEO є таким же важливим, як і контент, і його слід враховувати під час розробки веб-сайту. React.JS — одна з найкращих бібліотек JavaScript для створення веб-додатків на основі SEO. Пошукові системи ніколи не дають рейтинг веб-сайтам, які завантажуються вічно. У той час веб-сайти були причиною їх тривалого завантаження. За допомогою експертів React.JS можна мінімізувати довший час завантаження та збільшити швидкість відтворення веб-додатків. Це істотно вплине на продуктивність вашого сайту. Пошукові системи, такі як React.JS, індексують програми на основі React за пріоритетом;

– легше вчитися. Багато організацій використовують React.JS для створення своїх програм. Єдина причина цього полягає в тому, що React.JS легше вивчити та реалізувати, ніж будь-який інший фреймворк. Люди все більше схиляються до React.JS, оскільки він створює ефективні веб-програми;

– інструменти розробника React. React.JS є одним із найкращих інструментів для створення користувацьких інтерфейсів, а також підтримує інструменти React Developer, які можна встановити як розширення браузера. Розширення доступне як для Mozilla Firefox, так і для Google Chrome;

– односторонній потік даних. Більшість фреймворків мають двосторонній потік даних, тобто весь додаток потрібно змінювати щоразу, коли ви вносите будь-які зміни. Але з React.JS це односторонній потік даних, який гарантує, що код залишається стабільним. Якщо експерт змінить певний розділ, це не вплине на структуру програми. Наприклад, якщо ви працюєте над дочірніми елементами, це не вплине на батьківські дані;

– React Native — це одна з бібліотек розробки інтерфейсу користувача, яка широко використовується для створення багатьох мобільних додатків. React Native схожий на звичайну бібліотеку React, особливо для мобільних телефонів. Як і React JS, React Native працює з елементами, які колись були розроблені для створення цілих інтерфейсів користувача. Такі платформи, як Android, iOS тощо, можуть повторно використовувати коди. Розроблені

програми призначені для певних версій веб-програм. Наприклад, Android 5.0 або IOS 11.0.

Недоліки React JS:

- відсутність належної документації. Технологія React змінюється настільки швидко, що належну документацію неможливо завершити. Щоб подолати проблему документації, розробники пишуть інструкції для нових інструментів і випусків у своїх проектах;

- швидкість розвитку. Темпи розвитку мають свої недоліки та переваги. До недоліків можна віднести те, що технологія, пов'язана з React.JS, змінюється настільки швидко, що розробникам іноді здається, що їм потрібно вивчати нові способи виконання завдань. Їм потрібно оновити нові способи навчання та навички волі;

- JSX як перешкода. React.JS використовує JSX, розширення синтаксису, яке дозволяє HTML працювати з JavaScript. Цей підхід має переваги, але багато розробників вважають JSX перешкодою та скаржаться на його ускладнення під час навчання [11]-[13].

Існує багато спеціальних ніш для React.JS, де видно, що він ефективніший у використанні, ніж будь-який інший фреймворк. Ось список деяких із них:

- панелі та інструменти візуалізації даних. Якщо ви прагнете створити інтерактивні інформаційні панелі, орієнтовані на дані, із практичними інструментами візуалізації для спрощення розуміння даних, React.JS — ідеальний вибір. Крім того, ви можете отримати доступ до багатьох готових до використання шаблонів інформаційної панелі, і розробники можуть ефективно інтегрувати їх у свої проекти;

- веб-сайти електронної комерції. React.JS — ідеальний вибір, якщо у вас є веб-сайт електронної комерції, щоб покращити його досвід і продуктивність. React.JS використовує багаторазові елементи інтерфейсу користувача на різних сторінках, що допомагає заощадити час і створює

налаштовані та добре розроблені інструменти навігації для плавної роботи користувача;

– програми для соціальних мереж. Однією з переваг використання React.JS є те, що він може розробляти динамічні SPA, які можуть допомогти у вибіркового рендерингу замість оновлення всіх елементів веб-програми. Цей підхід покращує продуктивність програми та зберігає запити клієнт-сервер, і ці функції є обов’язковими для програм соціальних мереж [11]-[13].

Обґрунтування вибору фреймворку для розробки програмного забезпечення для стиснення зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору фреймворку для стиснення зображень

Критерій порівняння фреймворків	Фреймворки		
	React	Vue	Angular
Зручність використання	+	—	—
Швидкість виконання запитів	+	+	+—
Широкий вибір вбудованих та сторонніх розширень	+	+—	+—
Підтримка REST-протоколу	+	+—	+—
Використання великою спільнотою	+	+—	+

Таким чином, для створення програмного забезпечення для стиснення зображень обрано фреймворк React, що є високо функціональним вебфреймворком, розробленим на мові програмування JavaScript для швидкого створення ефективних та чітко структурованих вебзастосунків.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для стиснення зображень обрано мову програмування JavaScript, яка є мовою загального призначення високого рівня та характеризується своєю зрозумілістю і простотою синтаксису.

Для створення програмного забезпечення для стиснення зображень обрано фреймворк React, що є високо функціональним вебфреймворком, розробленим на мові програмування JavaScript для швидкого створення ефективних та чітко структурованих вебзастосунків.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для стиснення зображень

До розробки окремих модулів та компонентів програмного забезпечення для стиснення зображень, було визначено базову архітектуру. Хороший вибір архітектури може запобігти розробникам приймати неправильні рішення та витратити занадто багато часу на налаштування завдань у системі, тобто бізнес-логіки, тобто логіки програми. Детальне дослідження завдання на розробку обґрунтувало доцільність використання клієнт-серверної архітектури, оскільки, згідно завдання, веб-сайт має надавати сервіс зі стиснення зображень великій кількості користувачів [14].

Структуру програмного забезпечення для стиснення зображень наведено на рис. 3.1.

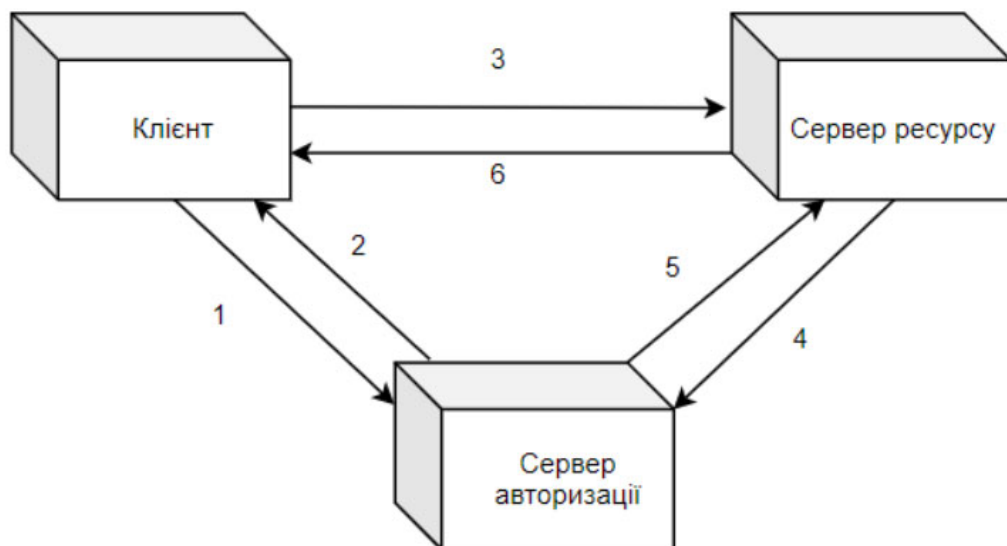


Рисунок 3.1 – Структура програмного забезпечення для стиснення зображень

Як видно з рис. 3.1, як базис для створення програмного забезпечення для стиснення зображень обрано клієнт-серверну архітектуру. Опис структури програмного забезпечення для стиснення зображень.

Крок 1. Клієнт надсилає запит на сервер авторизації для отримання веб-токена JSON.

Крок 2. Сервер авторизації перевіряє ідентифікатор клієнта та секретний ключ і повертає маркер.

Крок 3. Клієнт надсилає запит на сервер для отримання необхідної інформації, додаючи маркер до заголовка запиту.

Крок 4. Потім сервер ресурсів надсилає запит на сервер авторизації для перевірки маркера.

Крок 5. Сервер авторизації надсилає відповідь, чи маркер дійсний чи недійсний.

Крок 6. Після отримання правильної інформації про маркер Сервер ресурсів забезпечує доступ до клієнтів.

Програмне забезпечення для стиснення зображень являє собою вебзастосунок. Це дозволяє користувачам не завантажувати програми на пристрій, на якому вони працюють (наприклад, мобільних або настільних комп'ютерів) і дозволяє розробникам працювати на різних типах пристроїв (ноутбуки, планшети, смартфони тощо). Фреймворк, що забезпечує виконання, можна використовувати для створення адаптивного інтерфейсу користувача (тобто інтерфейсу, який відповідає пристрою, на якому працює користувач).

Розробка компонентів програмного забезпечення для стиснення зображень, таких як клієнтська та серверна частини, виконується окремо. Це дозволяє додатково тестувати та модифікувати ці компоненти незалежно один від одного, на відміну від монолітних систем, які залежать від різних інструментів розробки.

Серверна частина програмного забезпечення для стиснення зображень відповідає за обробку запитів, які надсилає клієнт, обробку даних, які отримує

від сервера для зберігання в системі, і взаємодію з загальною базою даних (БД). Згідно з загальноприйнятою практикою розробки, ці системні компоненти повинні використовувати архітектуру Model-View-Controller (MVC). Ці шаблони проектування дозволяють розбити серверну частину програмного забезпечення для стиснення зображень на менші модулі, сприяючи розробці та тестуванню незалежно і уникненню проблем з координацією системи.

Технологія MVC (Model, View і Controller). MVC є популярним способом організації коду. Основна ідея MVC полягає в тому, що кожна частина коду має певну мету, і ці цілі різні. Частина коду містить дані програми, частина коду забезпечує роботу програми, а частина коду контролює роботу програми.

В об'єктно-орієнтованому програмуванні контролер представлення моделі (MVC) — це ім'я, дане методології або шаблону проектування, які дозволяють інтерфейсу користувача успішно та ефективно спілкуватися з 34 базовими моделями даних. Шаблон MVC широко використовується при розробці додатків з такими мовами програмування, як Java, Smalltalk, C і C++. Багато розробників відзначали шаблон MVC як корисний шаблон для повторного використання об'єктного коду та шаблон, який може значно скоротити час, необхідний для розробки програм з інтерфейсом користувача. Модель контролера подання моделі представляє три основні компоненти або об'єкти, які будуть використовуватися в розробці програмного забезпечення: - модель, яка представляє базову логічну структуру даних у програмному додатку та пов'язаному з ним класі високого рівня. Ця об'єктна модель не містить інформації про інтерфейс користувача. - дисплей, який є набором класів, які представляють елементи в інтерфейсі користувача (усі речі, які користувач може бачити та реагувати на екрані, такі як кнопки, вікна дисплея тощо). - контролер, який представляє класи, які з'єднують модель і представлення, і використовується для зв'язку між класами в моделі і представленні. У контексті програмування на JavaScript модель складається з

простих класів JavaScript, представлення надає дані, а контролер складається з сервлетів. У результаті цього поділу запити користувачів обробляються наступним чином. Браузер клієнта надсилає запит сторінки до контролера на сервері. Потім контролер виконує дію виклику моделі, тим самим отримуючи необхідні дані у відповідь на запит. Потім контролер передає отримані дані в представлення. Візуалізація відтворюється та надсилається назад клієнту для відтворення в браузері. Архітектура MVC пропонує багато переваг програмісту при розробці програм:

- кілька розробників можуть одночасно працювати з трьома рівнями (Модель, Представлення та Контролер);
- пропонує покращену масштабованість, яка збільшує здатність програми розвиватися;
- оскільки компоненти мало залежать один від одного, їх легко обслуговувати;
- модель можна повторно використовувати з кількома представленнями, що забезпечує повторне використання коду;
- прийняття MVC робить програму більш виразною та легкою для розуміння;
- розширювати та тестувати додаток стає легко.

Структуру серверної частини програмного забезпечення для стиснення зображень, побудованої на основі архітектури MVC, наведено на рис. 3.2.

В архітектурі MVC модель включає в себе всю бізнес-логіку програмного забезпечення для стиснення зображень, таку як робота з зображеннями. Програмні модулі, пов'язані з представленням, відповідають за подання користувачам програмного забезпечення для стиснення зображень. Частина контролер з'єднує модель і представлення, оновлюючи представлення для користувача через інтерфейс моделі [15].



Рисунок 3.2 – Структура серверної частини програмного забезпечення для стиснення зображень на основі патерна MVC

3.2 Функціонування програмного забезпечення для стиснення зображень

На етапі проектування було вирішено, що програмне забезпечення для стиснення зображень складається із двох окремих модулів: клієнтського та серверного. Взаємодія між цими компонентами та обмін даними реалізуються за допомогою протоколу передачі даних HTTP, призначеного для відправлення запитів від клієнтів і серверів та обробки цих запитів.

Функціонування програмного забезпечення для стиснення зображень подамо за допомогою схеми, зображеної на рис. 3.3.

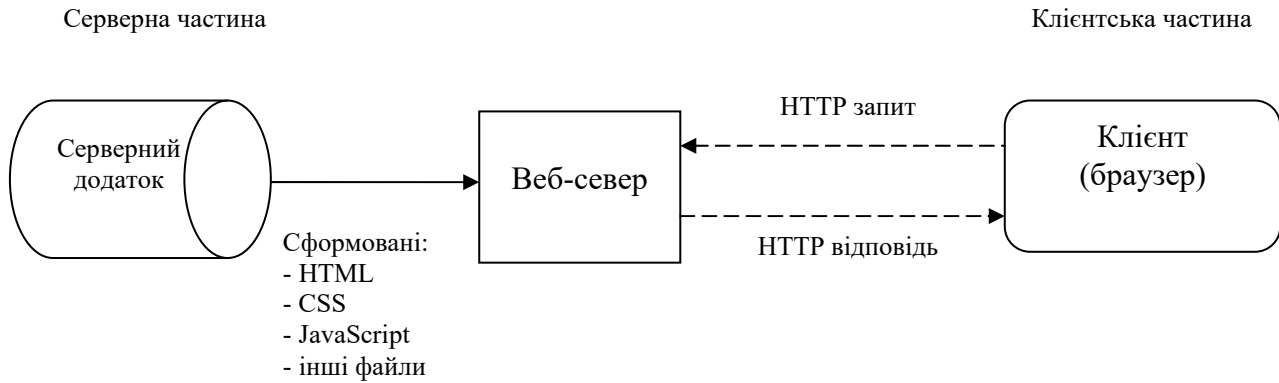


Рисунок 3.3 – Схема функціонування програмного забезпечення для стиснення зображень

Коли користувач хоче перейти на сторінку, браузер надсилає HTTP-запит GET із URL-адресою його HTML-сторінки. Сервер отримує запитаний документ зі своєї файлової системи та повертає відповідь HTTP, що містить документ і код статусу відповіді HTTP 200 OK (успішно). Сервер може повернути інший код стану, наприклад «404 Не знайдено», якщо файлу немає на сервері, або «301 Переміщено назавжди», якщо файл існує, але його переміщено в інше місце.

Сервер для веб-сайту обробляє лише запити GET, оскільки сервер не зберігає жодних змінених даних. Він також не змінює свої відповіді на основі даних запиту HTTP (таких як параметри URL-адреси або файли cookie).

Для забезпечення аутентифікації та встановлення дозволів на доступ до ресурсів використовуються відповідні маркери. Клієнти, які передають реєстраційні дані для аутентифікації, отримують ключ для подальшого доступу до сервера.

Алгоритм реалізації сервера визначається стандартом, створеним для опису взаємодії вебзастосунків, написаних на мові програмування JavaScript. Коли запит надходить до сервера програмного забезпечення для стиснення зображень, спочатку його обробляє проміжне програмне забезпечення. Цей рівень змінює запит до того, як він досягне відповідного контролера

програмного забезпечення для стиснення зображень, і повертає відповідь після обробки. Контролер, який буде виконувати запит, вибирається на основі адреси, на яку надійшов запит від програмного забезпечення для стиснення зображень.

При зверненні до клієнтської частини програмного забезпечення для стиснення зображень алгоритм отримання даних користувачем подібний до серверної частини.

3.3 Розробка алгоритм стиснення зображень

Використання запропонованого алгоритму обробки зображень обов'язково вимагає наявності елементів кодування та декодування (відтворення) зображення. Кодер забезпечує перетворення вхідного (нестисненого зображення) в байтовий масив, який кодує вхідне зображення. Крім того, якщо розмір вихідного масиву менший за розмір вихідного зображення, тоді можна запросити стиснення вхідної інформації, інакше мова йде про розширення. Важливим моментом також є можливість відтворення вихідного зображення без спотворень за допомогою стисненої стрічки в результаті групового кодування. На вхід кодера надходить початкове значення кольору зображення у формі матриці градацій сірого. Крім того, ми припустимо, що кожному кольору виділено один байт. Тобто діапазон можливих варіацій кольору від 0...255. Якщо необхідно стиснути кольорові зображення, наприклад у форматі RGB, то перед запуском кодера необхідно вибрати таблицю окремих компонентів червоного, зеленого і синього тонів. У цьому випадку кодер забезпечить кодування кожної компонентної таблиці формату RGB, що може призвести до ще більшого коефіцієнта стиснення вхідного зображення в цілому. Першим етапом кодувальника зображень є процес виділення областей зображення, які містять групові повторення одного кольору. Крім того, вибираються лише області, які відповідають певному розміру. Результатом роботи цього процесу є масив дескрипторів порядку

груп, що містить номер сегмента, починаючи з 501, кількість стовпців і рядків і номер кольору зображення. Значення 501 вибрано для маркування, яке не перекриває діапазон кольорів пікселів зображення. На зображенні значення кольорів сегментів, що відповідають елементам групових послідовностей, замінені значеннями групових послідовностей. Для подальшої роботи кодера використовується масив описів сегментів і модифіковане зображення з позначеними сегментами групових послідовностей. Наступним етапом роботи кодера є попиксельний аналіз модифікованого зображення, результатом якого є кодування зображення.

Тобто, якщо поточний піксель показує певний маркер групового порядку, тоді відбувається повторне кодування групового порядку, і елементи модифікованого зображення позначаються деяким маркером (наприклад, максимально можливим цілим числом), щоб ці пікселі не розглядалися знову ж таки, тому що вони вже закодовані за допомогою кодувальника. В іншому випадку, якщо значення кольору пікселя не представляє маркер групового порядку, це вказує на відсутність повтору, і, отже, відбувається кодування без повторів. Результатом цього процесу є вихідна стрічка як масив закодованих байтів зображень. На останньому кроці додаткові байти додаються до закодованої послідовності в кінці, що представляє розмір вхідного зображення. Наприкінці кодера створюється стрічка з одного набору байтів, яку можна передавати по каналах зв'язку або використовувати для подальшої обробки зображення. Схему роботи алгоритму стиснення зображення можна представити такою послідовністю операцій (рисунк 3.4).

Одним із процесів, які виконує кодер, є ідентифікація областей зображення з однаковими послідовностями кольорів (повторення груп). Схема цієї процедури наведена на рисунок 3.5.

Основна ідея процесу вибору групової послідовності полягає у виборі областей зображення, які представляють 4 пов'язані послідовності пікселів однакового кольору. Чотири підключених сусідів означає піксельне з'єднання в горизонтальному або вертикальному напрямку.

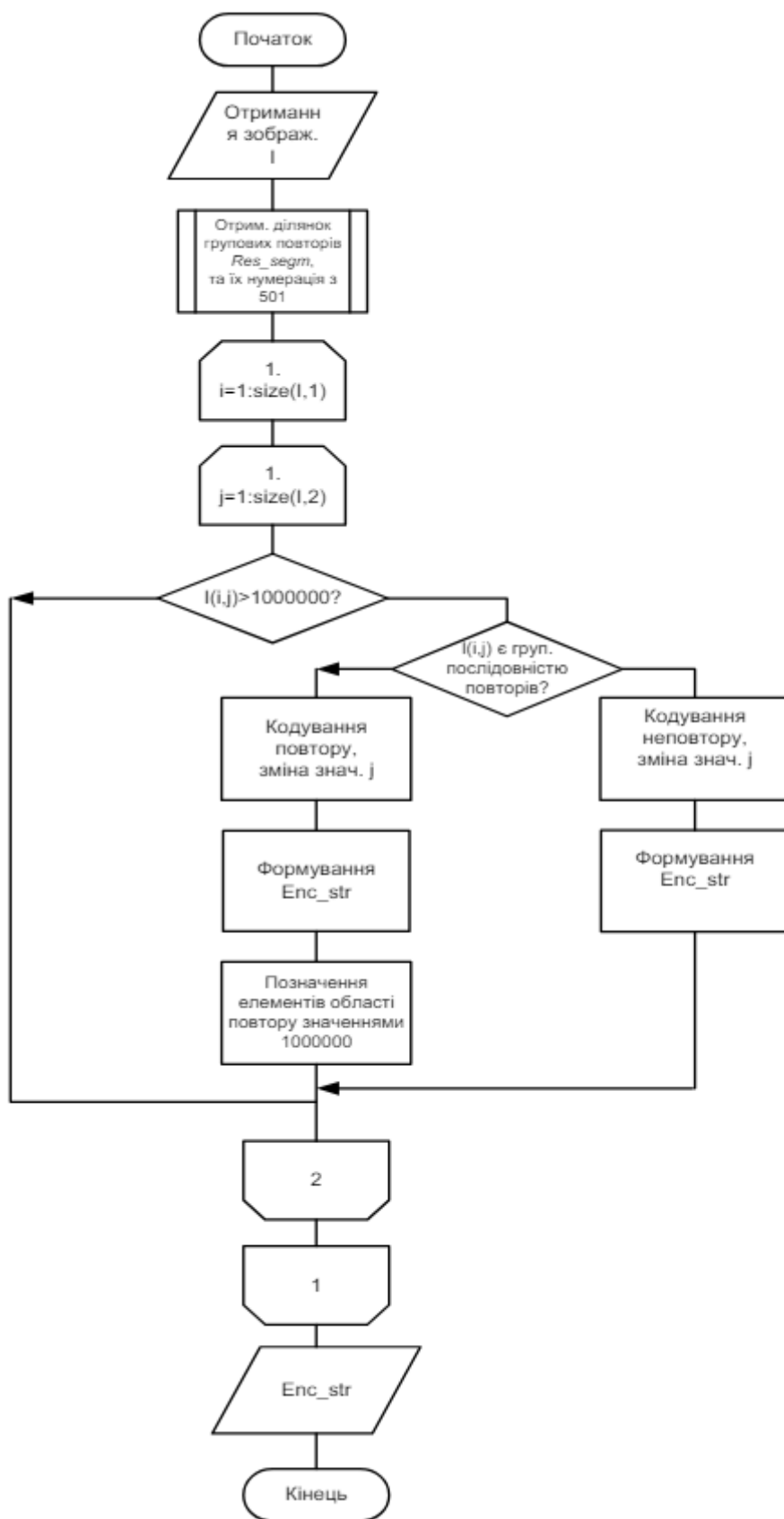


Рисунок 3.4 – Узагальнена схема роботи алгоритм стиснення зображень

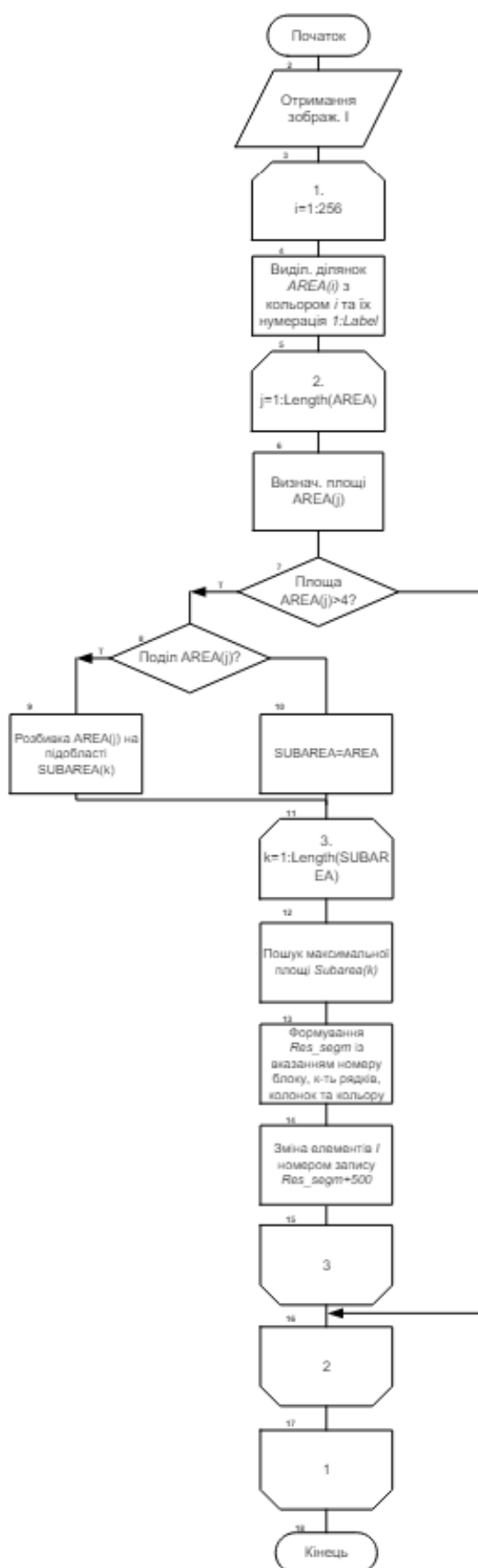


Рисунок 3.5 – Схема роботи процедури виділення групових послідовностей

3.4 Проектування інтерфейсу програмного забезпечення для стиснення зображень

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням для стиснення зображень виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для стиснення зображень враховані функціональні вимоги до програми:

- стиснення зображень без втрат якості;
- підтримка різних форматів файлів;
- зручний інтерфейс користувача;
- можливість обробляти одночасно кілька зображень;
- можливість аналізувати та порівнювати якість зображення до та після стиснення;
- можливість автоматично оптимізувати зображення для веб-сайтів.

Інтерфейс головної форми програмного забезпечення для стиснення зображень наведено на рис. 3.6.

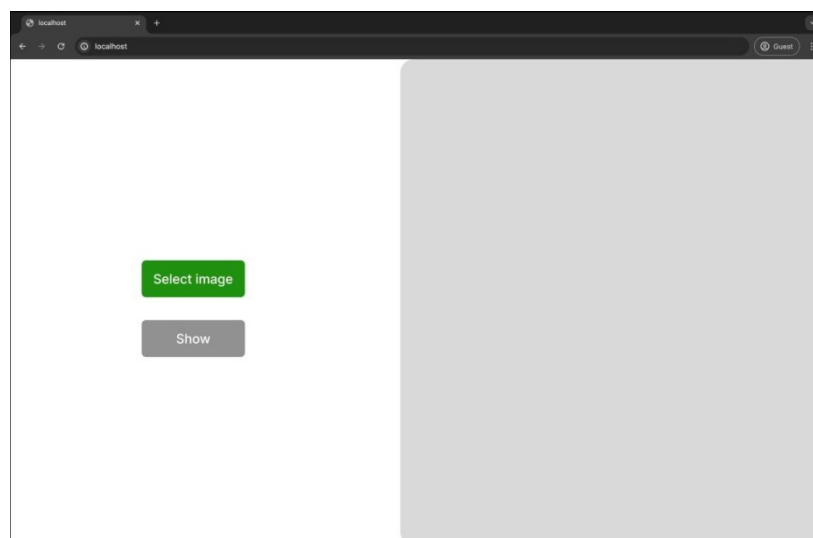


Рисунок 3.6 – Інтерфейс головної форми програмного забезпечення для стиснення зображень

На рисунку зображений прототип початкової сторінки сайту. При натисканні на кнопку “Select image” користувач обирає відповідні зображення для стиснення.

3.5 Висновки за розділом 3

Розроблено структуру програмного забезпечення для стиснення зображень. Визначено, що структура програмного забезпечення для стиснення зображень складається з клієнтської та серверної частин.

Розроблено алгоритм стиснення зображень для програмного забезпечення для стиснення зображень.

Розроблено програмне забезпечення для стиснення зображень.

Описано особливості реалізації програмного забезпечення для стиснення зображень. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для стиснення зображень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для стиснення зображень. Програма написана на мові JavaScript, яка є мовою загального призначення високого рівня та характеризується своєю зрозумілістю і простотою синтаксису. В якості фреймворку для створення програмного забезпечення для стиснення зображень обрано React, що є високо функціональним вебфреймворком, розробленим на мові програмування JavaScript для швидкого створення ефективних та чітко структурованих вебзастосунків.

Програмне забезпечення для стиснення зображень забезпечує виконання таких функцій:

- стиснення зображень без втрат якості;
- підтримка різних форматів файлів;
- зручний інтерфейс користувача;
- можливість обробляти одночасно кілька зображень;
- можливість аналізувати та порівнювати якість зображення до та після стиснення;
- можливість автоматично оптимізувати зображення для веб-сайтів.

Для експлуатації програмного забезпечення для стиснення зображень необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 4 Гб і більше;
- не менше 1 Гб вільного простору на запам'ятовуючому пристрою апаратного засобу;
- процесор із тактовою частотою не менше 2.5 ГГц;
- наявність вебсерверу;
- встановлена операційна система та веббраузер на стороні клієнта.

Функціональні характеристики розробленого програмного забезпечення для стиснення зображень наведено у технічному завданні

(додаток А). Фрагмент тексту програмного забезпечення для стиснення зображень наведено у додатку Б.

4.2 Характеристики програми для стиснення зображень

Програмне забезпечення для стиснення зображень складається з двох основних компонентів, серверного та клієнтського.

На стороні сервера програми стиснення зображень працює модуль, написаний на мові програмування JavaScript і використовує веб-фреймворк React. Запит програми обробляється сервером до того, як він досягне кінцевого сервера. Серверна частина складається з кількох модулів, які виконують роль сервісів і реалізують архітектуру MVC.

Клієнтська сторона стиснення зображень — це окремий програмний компонент, написаний мовою програмування JavaScript із використанням веб-фреймворку React. Кожен компонент клієнтської програми стиснення зображень є окремою сторінкою або блоком повторно використовуваних даних, які постійно оновлюються. Клієнтська сторона викликає API реалізації сервера для подальшого відображення або обробки даних.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для стиснення зображень на серверній частині необхідно запустити відповідний файл. Для звертання до клієнтської сторони необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для стиснення зображень є оригінал зображення, завантаженого користувачем.

Вихідними даними програмного забезпечення для стиснення зображень є стиснуті вихідні зображення.

4.3.3 Повідомлення

Користувач програмного забезпечення для стиснення зображень може отримати такі повідомлення: повідомлення про успішне закінчення процесу стиснення зображень, або помилку завантаження файлів.

4.4 Виконання програмного забезпечення для стиснення зображень

Початкова сторінка веб-сайту для стиснення зображень зображена на рисунку 4.1. На цій сторінці розміщено дві основні кнопки: "Select image" та "Show". Вони відіграють ключову роль у процесі взаємодії користувача з веб-сайтом і забезпечують простоту та зручність використання.

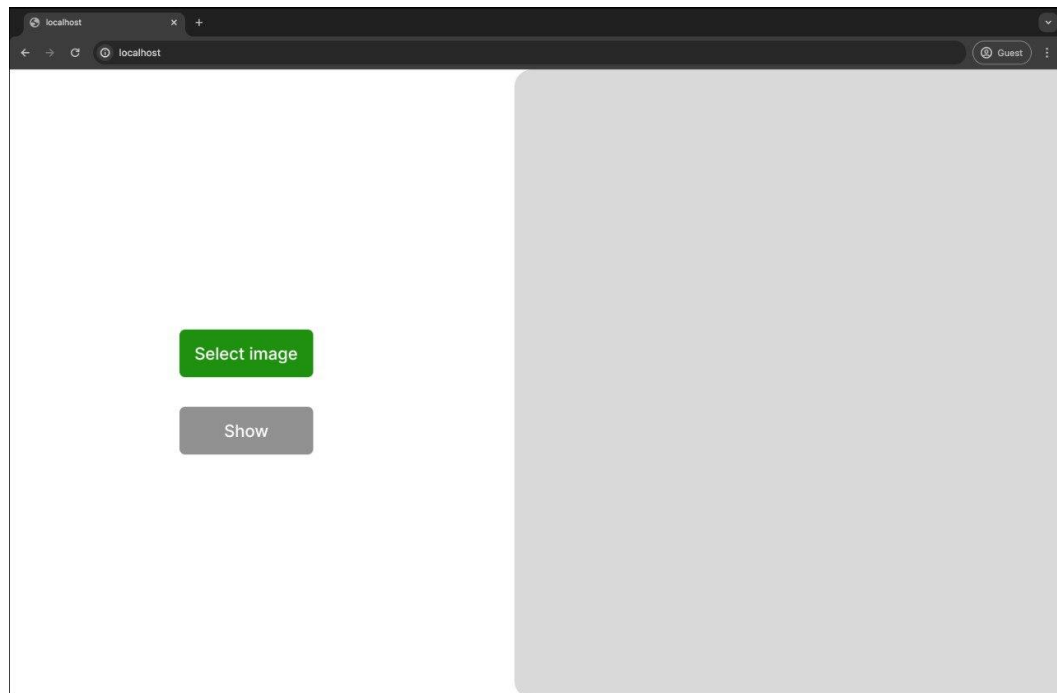


Рисунок 4.1 – Початкова сторінка веб-сайту для стиснення зображень

При натисканні на кнопку "Select image" користувачу відкривається вікно вибору файлів з його персонального диска, як показано на рисунку 4.2. Це стандартне вікно вибору файлів, яке дозволяє користувачу легко знайти та вибрати потрібний файл зображення зі свого комп'ютера або іншого пристрою зберігання. Інтерфейс вибору файлів є інтуїтивно зрозумілим і підтримує широкий спектр форматів зображень, таких як JPEG, PNG та інші.

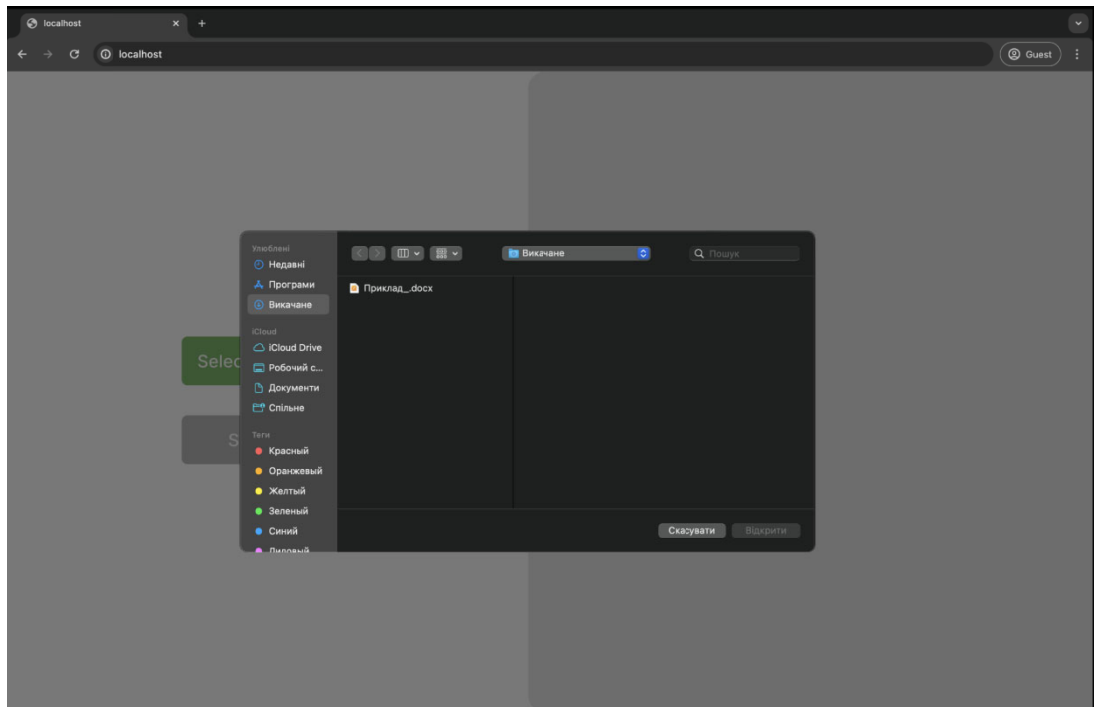


Рисунок 4.2 – Результат натискання на кнопку "Select image"

Після того, як користувач обрав потрібний файл і завантажив його на веб-сайт, він повинен активувати кнопку "Show". Натискання на цю кнопку запускає процес стиснення зображення. Система веб-сайту використовує спеціальні алгоритми для зменшення розміру файлу зображення без значної втрати якості. Ці алгоритми можуть включати методи зниження роздільної здатності, оптимізацію кольорів та інші техніки, спрямовані на ефективне стиснення даних.

Зжати файл відображається на екрані користувача, як показано на рисунку 4.3. Це дозволяє користувачу візуально оцінити результат стиснення і переконатися, що якість зображення залишилася на прийнятному рівні.

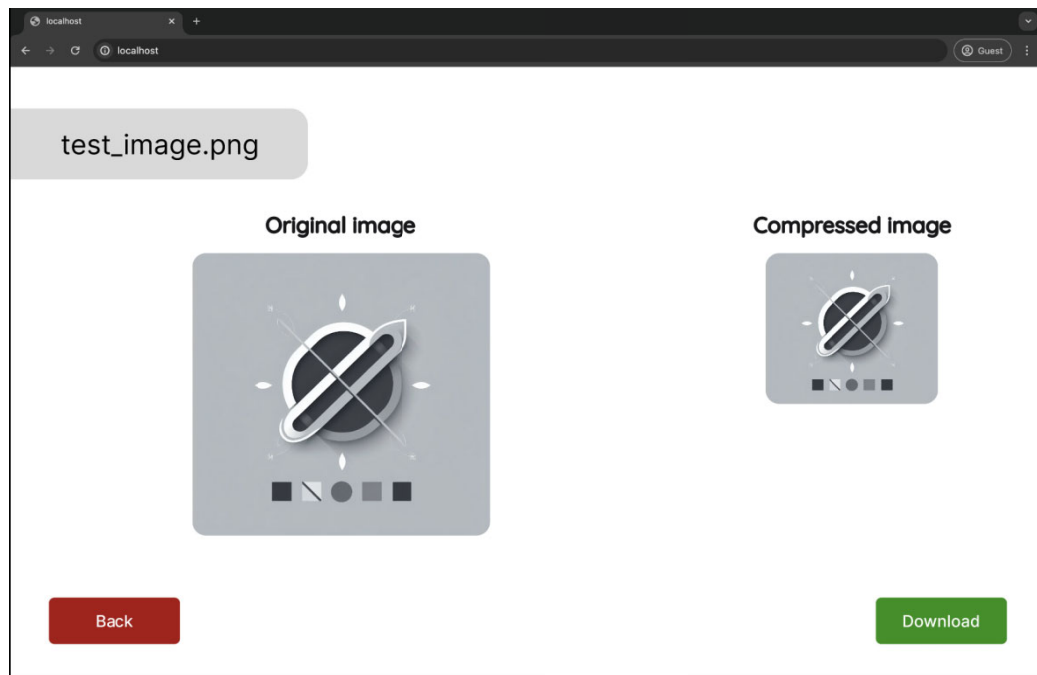


Рисунок 4.3 – Відображення результату стиснення файлу

При натисканні кнопки "Download" в правому нижньому куті користувач завантажує зображення на свій комп'ютер. Цей процес є важливим завершальним етапом всього циклу обробки зображень на веб-сайті. Після того, як зображення було успішно стиснуто і користувач задоволений результатом, функція завантаження дозволяє йому зберегти оптимізоване зображення на свій пристрій для подальшого використання.

Функціонал кнопки "Download" забезпечує зручність та ефективність, дозволяючи користувачу швидко отримати доступ до зжатого файлу без додаткових кроків. Після натискання на кнопку, файл автоматично завантажується. Це зручне рішення дозволяє користувачам економити час і зусилля, адже користувач може негайно використовувати зображення для своїх потреб.

Весь процес, від вибору файлу до отримання зжатого зображення, є інтуїтивно зрозумілим і легким у використанні. Система забезпечує високу швидкість обробки, що дозволяє користувачам швидко отримувати результати без тривалих затримок. Це робить веб-сайт ефективним інструментом для тих,

хто регулярно працює з великими обсягами зображень і потребує їх стиснення для збереження місця на диску або швидшої передачі файлів через Інтернет.

Загалом, головна сторінка веб-сайту для стиснення зображень є зручною та функціональною, забезпечуючи користувачам простий доступ до необхідних інструментів для стиснення зображень. Вона поєднує в собі простоту використання, інтуїтивний інтерфейс та потужні можливості стиснення, що робить її незамінним інструментом для багатьох користувачів.

4.5 Тестування програмного забезпечення для стиснення зображень

Виконано тестування програмного забезпечення для стиснення зображень.

Серверна частина програмного забезпечення для стиснення зображень піддалася модульному тестуванню та інтеграційному тестуванню. Для цього використовувалися стандартні інструменти фреймворку React, а також бібліотека React Testing Library. Протестовано було програмні компоненти, що реалізовували логіку роботи програми, зокрема, класи, які відповідають за завантаження файлів, розширені підкласи та класи, відповідальні за стиснення зображень.

Виявлено, що програма для стиснення зображень функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу - стиснення зображень.

Розроблене програмне забезпечення дозволяє забезпечувати процес стиснення зображень.

4.6 Висновки за розділом 4

Описано програмне забезпечення для стиснення зображень. Виконано тестування розробленого програмного забезпечення для стиснення зображень.

Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити дозволяє забезпечувати процес стиснення зображень.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення компресії зображень.

Аналіз різних методів стиснення зображень показав, що існують різні підходи до оптимізації розміру файлів зображень. Методи можуть варіюватися від безвтратних до втратних, з різними алгоритмами стиснення та рівнем збереження деталей. Оцінка ефективності методів варіюється в залежності від типу зображення, потреб користувача та вимог до якості. Завдяки ретельному аналізу різних методів стиснення зображень, стало очевидним, що кожен метод має свої переваги та обмеження. Аналіз програмного забезпечення для стиснення зображень виявив, що існує різноманітність програм, що пропонують різні функції для оптимізації файлів зображень.

Виявлено, що багато програм пропонують різні рівні автоматизації та налаштування, що дозволяє користувачам підлаштувати процес оптимізації під свої потреби. Зроблено висновок, що вибір програмного забезпечення для стиснення зображень повинен бути зроблений на основі конкретних потреб та вимог користувача, враховуючи різноманітність функцій та можливостей кожного інструменту. При проведенні аналізу зроблено висновок, що існує досить багато програмних засобів для стиснення зображень. Проте деякі програмні засоби для стиснення зображень є менш гнучкими у відношенні до специфічних потреб конкретного користувача. З метою специфічних користувацьких потреб важливо вибирати надійне програмне забезпечення або розробляти власне програмне забезпечення.

Сформульовано функціональні вимоги до програмного забезпечення для стиснення зображень.

Для реалізації програмного забезпечення для стиснення зображень обрано мову програмування JavaScript, яка є мовою загального призначення

високого рівня та характеризується своєю зрозумілістю і простотою синтаксису.

Для створення програмного забезпечення для стиснення зображень обрано фреймворк React, що є високо функціональним вебфреймворком, розробленим на мові програмування JavaScript для швидкого створення ефективних та чітко структурованих вебзастосунків.

Розроблено структуру програмного забезпечення для стиснення зображень. Визначено, що структура програмного забезпечення для стиснення зображень складається з клієнтської та серверної частин.

Розроблено алгоритм стиснення зображень для програмного забезпечення для стиснення зображень.

Розроблено програмне забезпечення для стиснення зображень.

Описано особливості реалізації програмного забезпечення для стиснення зображень. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для стиснення зображень.

Описано програмне забезпечення для стиснення зображень. Виконано тестування розробленого програмного забезпечення для стиснення зображень. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити дозволяє забезпечувати процес стиснення зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Analysis of Different Image Compression Techniques: A Review [Electronic resource]. – Access mode : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4031725
2. . Image Compression: A Review [Electronic resource]. – Access mode : <https://ijarsct.co.in/Paper1726.pdf>
3. A Review on Image Compression Techniques [Electronic resource]. – Access mode : https://link.springer.com/chapter/10.1007/978-981-15-5397-4_29
4. Showdown! 5 Online Image Compression Tools Compared [Electronic resource]. – Access mode : <https://www.sitepoint.com/image-compression-tools/>
5. Top 10 Image Compression Tools [Electronic resource]. – Access mode : <https://www.softlist.io/top-product-reviews/top-10-image-compression-tools/>
6. Nine Best Online Image Optimizer Tools Compared (Real Test Data) [Electronic resource]. – Access mode : <https://themeisle.com/blog/best-online-image-optimizer-tools/>
7. 21 Best Image Compression Tools [Electronic resource]. – Access mode : <https://www.softlist.io/21-best-image-compression-tools/>
8. JavaScript basics [Electronic resource]. – Access mode : https://developer.mozilla.org/enUS/docs/Learn/Getting_started_with_the_web/JavaScript_basics
9. JavaScript Articles [Electronic resource]. – Access mode : <https://www.codecademy.com/articles/language/javascript>
10. JavaScript Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/js/>
11. What is React.js? Uses, Examples, & More [Electronic resource]. – Access mode : <https://blog.hubspot.com/website/react-jstext=React20provides20stateDof2Dthetime20with20JavaScrip2Ddriven20pages>.
12. About react/review: a responsive journal for art & architecture [Electronic resource]. – Access mode : <https://escholarship.org/uc/reactreview/about>

13. How Difficult is it to Learn React? [Electronic resource]. – Access mode : <https://www.nobledesktop.com/learn/react/how-difficult-is-it-to-learn-react#:~:text=Learning%20React%20requires%20a%20solid,first%20before%20diving%20into%20React.>

14. MVC Reviews [Electronic resource]. – Access mode : <https://www.glassdoor.com/Reviews/MVC-Reviews-E388858.htm>

15. What is Client-Server Architecture? Everything You Should Know [Electronic resource]. – Access mode : https://www.simplilearn.com/what-is-client-server-architecture-article=Client2Dserver_architecture_alternativelycalled.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для стиснення зображень.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розробка вебсайту для компресії зображень», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Призначення розробки

Програмний продукт призначений для стиснення зображень.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для стиснення зображень забезпечує виконання таких функцій:

- стиснення зображень без втрат якості;
- підтримка різних форматів файлів;
- зручний інтерфейс користувача;
- можливість обробляти одночасно кілька зображень;
- можливість аналізувати та порівнювати якість зображення до та після стиснення;
- можливість автоматично оптимізувати зображення для веб-сайтів.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для стиснення зображень повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для стиснення зображень повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для стиснення зображень необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для стиснення зображень необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 4 Гб і більше;
- не менше 1 Гб вільного простору на запам'ятовуючому пристрою апаратного засобу;
- процесор із тактовою частотою не менше 2.5 ГГц;
- наявність вебсерверу;
- встановлена операційна система та веббраузер на стороні клієнта.

A.3.6 Вимоги до маркування і пакування

Програма для стиснення зображень може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для стиснення зображень».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для стиснення зображень є оригінал зображення, завантаженого користувачем.

Вихідними даними програмного забезпечення для стиснення зображень є стиснуті вихідні зображення.

ДОДАТОК Б
Фрагмент тексту програми

```

import { ClassSerializerInterceptor } from '@nestjs/common';
import { NestFactory, Reflector } from '@nestjs/core';
import { NestExpressApplication } from '@nestjs/platform-express';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';
import { EnvironmentTypes } from 'src/connectors/config/types/environmentTypes';
import { ApiServiceModule } from 'src/services/http/api/api-service.module';
import { generateValidationPipe } from 'src/services/http/common/pipes/validation.pipes';

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(ApiServiceModule);
  const config = app.get<ConfigConnectorService>(ConfigConnectorService);
  const { PORT, HOST, NODE_ENV } = config.getAppConfig();
  const { prefixes } = config.getApiConfig();

  app.setGlobalPrefix(prefixes.main);
  app.useGlobalPipes(generateValidationPipe(NODE_ENV === EnvironmentTypes.local));
  app.useGlobalInterceptors(new ClassSerializerInterceptor(new Reflector()));

  await app.listen(PORT, HOST);
}

bootstrap();

import { repl } from '@nestjs/core';
import { ApiServiceModule } from 'src/services/http/api/api-service.module';

async function bootstrap() {
  const replServer = await repl(ApiServiceModule);

  replServer.setupHistory('.nestjs_repl_history', (err) => {
    if (err) {
      console.error(err);
    }
  });
}

bootstrap();

import { Module } from '@nestjs/common';
import { APP_FILTER } from '@nestjs/core';
import { DatabaseConnectorModule } from 'src/connectors/database/database-connector.module';
import { AuthControllerModule } from 'src/controllers/auth-controller/auth-controller.module';
import { ImageControllerModule } from 'src/controllers/image-controller/image-controller.module';
import { CommonExceptionHandler } from 'src/services/http/common/exceptions-filters/common.exception-filter';

@Module({
  imports: [DatabaseConnectorModule, AuthControllerModule, ImageControllerModule],
  providers: [
    {

```

```

    provide: APP_FILTER,
    useValue: CommonExceptionHandler
  }
]
})
export class ApiServiceModule {}

```

```

import { ArgumentsHost, Catch, ExceptionFilter, HttpException } from '@nestjs/common';
import { Response } from 'express';

```

```

@Catch(HttpException)
export class CommonExceptionHandler implements ExceptionFilter {
  catch(exception: HttpException, host: ArgumentsHost) {
    const ctx = host.switchToHttp();
    const response = ctx.getResponse<Response>();
    const request = ctx.getRequest<Request>();
    const status = exception.getStatus();

    response.status(status).json({
      statusCode: status,
      timestamp: new Date().toISOString(),
      path: request.url
    });
  }
}

```

```

import { HttpException, ValidationError, ValidationPipe, ValidationPipeOptions } from
'@nestjs/common';

```

```

function shortExceptionFactory(errors: ValidationError[]) {
  return new HttpException(
    {
      errors: errors.map(({ value, property, constraints }) => ({
        value,
        property,
        constraints
      })))
    },
    400
  );
}

```

```

function fullExceptionFactory(validationError: ValidationError[]) {
  const errors = validationError.map((e) => ({
    value: e.value,
    property: e.property,
    constraints: e.constraints
  }));

  return new HttpException(

```

```

    {
      errors
    },
    400
  );
}

```

```

export function generateValidationPipe(printFullError = false, options: ValidationPipeOptions = {}) {
  const exceptionFactory = printFullError ? fullExceptionFactory : shortExceptionFactory;

  return new ValidationPipe({
    whitelist: true,
    transform: true,
    exceptionFactory,
    ...options
  });
}

```

```

import { NestFactory } from '@nestjs/core';
import { NestExpressApplication } from '@nestjs/platform-express';
import { ImageCompressorModule } from 'src/services/image-compressor/image-compressor.module';
import { ImageCompressorService } from 'src/services/image-compressor/image-compressor.service';

```

```

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(ImageCompressorModule);
  const imageCompressorService = app.get(ImageCompressorService);
  await imageCompressorService.run();
}

```

```
bootstrap();
```

```

import { Module } from '@nestjs/common';
import { DatabaseConnectorModule } from 'src/connectors/database/database-connector.module';
import { ImageConsumerModule } from 'src/modules/image-consumer/image-consumer.module';
import { ImageCompressorService } from 'src/services/image-compressor/image-compressor.service';

```

```

@Module({
  imports: [DatabaseConnectorModule, ImageConsumerModule],
  providers: [ImageCompressorService]
})
export class ImageCompressorModule {}

```

```

import { Injectable } from '@nestjs/common';
import { ImageConsumerService } from 'src/modules/image-consumer/image-consumer.service';

```

```

@Injectable()
export class ImageCompressorService {

```

```

constructor(private imageConsumerService: ImageConsumerService) {}

public async run() {
  while (true) {
    try {
      await this.imageConsumerService.process();
    } catch (e) {
      console.log(e);
    }
  }
}

import { Module } from '@nestjs/common';
import { CognitoService } from 'src/connectors/aws/cognito.service';
import { S3Service } from 'src/connectors/aws/s3.service';
import { SqsService } from 'src/connectors/aws/sqs.service';
import { ConfigConnectorModule } from 'src/connectors/config/config-connector.module';

@Module({
  imports: [ConfigConnectorModule],
  providers: [CognitoService, S3Service, SqsService],
  exports: [CognitoService, S3Service, SqsService]
})
export class AwsConnectorModule {}

import {
  AssociateSoftwareTokenCommand,
  AuthenticationResultType,
  ChallengeNameType,
  CognitoIdentityProvider,
  CognitoIdentityProviderClient,
  ConfirmSignUpCommand,
  GetUserCommand,
  InitiateAuthCommand,
  ResendConfirmationCodeCommand,
  RespondToAuthChallengeCommand,
  SetUserMFAPreferenceCommand,
  SignUpCommand,
  VerifySoftwareTokenCommand
} from '@aws-sdk/client-cognito-identity-provider';
import { Injectable } from '@nestjs/common';
import {
  IAuthenticationResult,
  IRawUserAttributesMap,
  IRefreshedAuthenticationResult,
  IUserAttributesMap
} from 'src/connectors/aws/types/cognito-service.types';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';

// TODO implement SecretHash support
@Injectable()

```

```

export class CognitoService {
  private readonly client: CognitoIdentityProviderClient;
  private readonly clientId: string;

  // private readonly clientSecret: string;

  constructor(private configConnectorService: ConfigConnectorService) {
    const { COGNITO_CLIENT_ID, /*COGNITO_CLIENT_SECRET,*/ COGNITO_REGION }
= configConnectorService.getCognitoConfig();
    const { SECRET_ACCESS_KEY, ACCESS_KEY_ID } =
configConnectorService.getAwsConfig();

    this.clientId = COGNITO_CLIENT_ID;
    // this.clientSecret = COGNITO_CLIENT_SECRET;
    this.client = new CognitoIdentityProvider({
      region: COGNITO_REGION,
      credentials: {
        accessKeyId: ACCESS_KEY_ID,
        secretAccessKey: SECRET_ACCESS_KEY
      }
    });
  }

  // private makeSecretHash(nickname: string): string {
  //   const hasher = createHmac('sha256', this.clientSecret);
  //   hasher.update(`${nickname}${this.clientId}`);
  //   //
  //   return hasher.digest('base64');
  // }

  async registerUser(nickname: string, email: string, name: string, password: string):
Promise<void> {
    const command = new SignUpCommand({
      ClientId: this.clientId,
      // SecretHash: this.makeSecretHash(nickname),
      Password: password,
      Username: nickname,
      UserAttributes: [
        {
          Name: 'email',
          Value: email
        },
        {
          Name: 'nickname',
          Value: nickname
        },
        {
          Name: 'name',
          Value: name
        }
      ]
    });
  }
}

```

```

    await this.client.send(command);
  }

  async confirmUserRegistration(nickname: string, confirmationCode: string): Promise<void> {
    const command = new ConfirmSignUpCommand({
      ClientId: this.clientId,
      // SecretHash: this.makeSecretHash(nickname),
      Username: nickname,
      ConfirmationCode: confirmationCode
    });
    await this.client.send(command);
  }

  async resendRegistrationConfirmCode(nickname: string): Promise<void> {
    const command = new ResendConfirmationCodeCommand({
      ClientId: this.clientId,
      // SecretHash: this.makeSecretHash(nickname),
      Username: nickname
    });
    await this.client.send(command);
  }

  async getUserByAuthToken(accessToken: string): Promise<IUserAttributesMap> {
    const command = new GetUserCommand({
      AccessToken: accessToken
    });
    const result = await this.client.send(command);
    const rawUserData = result.UserAttributes.reduce<IRawUserAttributesMap>(
      (accumulator, attribute) => {
        accumulator[attribute.Name] = attribute.Value;

        return accumulator;
      },
      { email: "", email_verified: "", name: "", nickname: "", sub: "" }
    );

    return {
      name: rawUserData.name,
      email: rawUserData.email,
      nickname: rawUserData.nickname,
      isEmailVerified: Boolean(rawUserData.email_verified),
      subscription: rawUserData.sub
    };
  }

  async setupMfa(accessToken: string): Promise<string> {
    const setupMfaCommand = new AssociateSoftwareTokenCommand({
      AccessToken: accessToken
    });
    const { SecretCode: secretCode } = await this.client.send(setupMfaCommand);

    return secretCode;
  }

```

```

    }

    async verifyMfa(accessToken: string, mfaCode: string): Promise<void> {
        const setupMfaCommand = new VerifySoftwareTokenCommand({
            AccessToken: accessToken,
            UserCode: mfaCode
        });
        await this.client.send(setupMfaCommand);
        await this.changeMfaSettings(accessToken, true);
    }

    async changeMfaSettings(accessToken: string, mfaStatus: boolean): Promise<void> {
        const changeMfaSettingsCommand = new SetUserMFAPreferenceCommand({
            AccessToken: accessToken,
            SMSMfaSettings: {
                Enabled: false,
                PreferredMfa: false
            },
            SoftwareTokenMfaSettings: {
                Enabled: mfaStatus,
                PreferredMfa: mfaStatus
            }
        });

        await this.client.send(changeMfaSettingsCommand);
    }

    async authenticateByRefreshToken(refreshToken: string):
    Promise<IRefreshedAuthenticationResult> {
        const command = new InitiateAuthCommand({
            ClientId: this.clientId,
            AuthFlow: 'REFRESH_TOKEN_AUTH',
            AuthParameters: {
                REFRESH_TOKEN: refreshToken
                // SECRET_HASH: this.makeSecretHash(nickname)
            }
        });
        const result = await this.client.send(command);

        return this.convertRefreshedAuthenticationResult(result.AuthenticationResult);
    }

    private convertRefreshedAuthenticationResult(
        authenticationResult: AuthenticationResultType
    ): IRefreshedAuthenticationResult {
        return {
            accessToken: authenticationResult.AccessToken,
            idToken: authenticationResult.IdToken,
            expireIn: authenticationResult.ExpiresIn
        };
    }
}

```

```

    async authenticateUser(emailOrNickname: string, password: string, mfaCode?: string):
    Promise<IAuthenticationResult> {
        const authCommand = new InitiateAuthCommand({
            ClientId: this.clientId,
            AuthFlow: 'USER_PASSWORD_AUTH',
            AuthParameters: {
                USERNAME: emailOrNickname,
                PASSWORD: password
                // SECRET_HASH: this.makeSecretHash(emailOrNickname)
            }
        });
        const authResponse = await this.client.send(authCommand);

        if (!authResponse.ChallengeName) {
            return this.convertAuthenticationResul(authResponse.AuthenticationResult);
        }

        switch (authResponse.ChallengeName) {
            case ChallengeNameType.SOFTWARE_TOKEN_MFA: {
                const mfaCommand = new RespondToAuthChallengeCommand({
                    ClientId: this.clientId,
                    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
                    ChallengeResponses: {
                        SOFTWARE_TOKEN_MFA_CODE: mfaCode,
                        USERNAME: emailOrNickname
                    },
                    Session: authResponse.Session
                });
                const mfaResult = await this.client.send(mfaCommand);

                return this.convertAuthenticationResul(mfaResult.AuthenticationResult);
            }

            default: {
                throw new Error('Unsupported channel name');
            }
        }
    }

    private convertAuthenticationResul(authenticationResult: AuthenticationResultType):
    IAuthenticationResult {
        return {
            accessToken: authenticationResult.AccessToken,
            refreshToken: authenticationResult.RefreshToken,
            idToken: authenticationResult.IdToken,
            expireIn: authenticationResult.ExpiresIn
        };
    }
}

import { GetObjectCommand, PutObjectCommand, S3Client, S3ClientConfig } from '@aws-
sdk/client-s3';

```

```
import { Injectable } from '@nestjs/common';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';
import { EnvironmentTypes } from 'src/connectors/config/types/environmentTypes';
import { Readable } from 'stream';
```

```
@Injectable()
```

```
export class S3Service {
  private client: S3Client;
  private readonly bucketName: string;
  private readonly nodeEnv: string;

  constructor(configConnectorService: ConfigConnectorService) {
    const { S3_REGION, S3_NAME } = configConnectorService.getS3Config();
    const { SECRET_ACCESS_KEY, ACCESS_KEY_ID } =
configConnectorService.getAwsConfig();
    const { NODE_ENV } = configConnectorService.getAppConfig();
```

```
    this.bucketName = S3_NAME;
```

```
    const config: S3ClientConfig = {
      region: S3_REGION,
      forcePathStyle: false,
      credentials: {
        accessKeyId: ACCESS_KEY_ID,
        secretAccessKey: SECRET_ACCESS_KEY
      }
    };
  };
}
```

```
  if (NODE_ENV === EnvironmentTypes.local) {
    config.endpoint = 'http://localhost:4566';
    config.forcePathStyle = true;
  }
  this.nodeEnv = NODE_ENV;
  this.client = new S3Client(config);
}
```

```
  public getBaseUrl(key: string): string {
    if (this.nodeEnv === EnvironmentTypes.local) {
      return `http://localhost:4566/${this.bucketName}/${key}`;
    }

    return `https://${this.bucketName}.s3.amazonaws.com/${key}`;
  }
}
```

```
  public async uploadFile(key: string, body: string | Uint8Array | Buffer): Promise<void> {
    const command = new PutObjectCommand({
      Bucket: this.bucketName,
      Key: key,
      Body: body
    });
  };
}
```

```
  await this.client.send(command);
}
```

```

    }

    public async getFile(key: string): Promise<Buffer> {
        const command = new GetObjectCommand({
            Bucket: this.bucketName,
            Key: key
        });

        const { Body } = await this.client.send(command);
        const body = Body as Readable;
        const arrayBuffer: Buffer[] = [];

        for await (const data of body) {
            arrayBuffer.push(Buffer.from(data));
        }

        body.destroy();

        return Buffer.concat(arrayBuffer);
    }
}

import {
    DeleteMessageCommand,
    ReceiveMessageCommand,
    SendMessageCommand,
    SQS,
    SQSClientConfig
} from '@aws-sdk/client-sqs';
import { MessageAttributeValue } from '@aws-sdk/client-sqs/dist-types/models/models_0';
import { Injectable } from '@nestjs/common';
import {
    IAttributes,
    IReceiveMessagesResult,
    ISendMessageParams,
    TMessageDataType
} from 'src/connectors/aws/types/sqs-service.types';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';
import { EnvironmentTypes } from 'src/connectors/config/types/environmentTypes';

@Injectable()
export class SqsService {
    private client: SQS;
    private readonly queueUrl: string;

    constructor(configConnectorService: ConfigConnectorService) {
        const { SQS_REGION, SQS_SECRET_ACCESS_KEY, SQS_ACCESS_KEY_ID, SQS_URL
    } = configConnectorService.getSqsConfig();
        const { NODE_ENV } = configConnectorService.getAppConfig();

        const config: SQSClientConfig = {
            region: SQS_REGION,

```

```

    credentials: {
      accessKeyId: SQS_ACCESS_KEY_ID,
      secretAccessKey: SQS_SECRET_ACCESS_KEY
    }
  };

  if (NODE_ENV === EnvironmentTypes.local) {
    config.endpoint = 'http://localhost:4566';
    config.credentials = undefined;
  }

  this.queueUrl = SQS_URL;
  this.client = new SQS(config);
}

private makeSqsAttributes<T>(attributes: IAttributes<T>) {
  return Object.entries(attributes).reduce<Record<string, MessageAttributeValue>>(
    (accumulator, [name, { value, makeStringValue }]) => {
      const dataType: TMessageDataType = typeof value === 'number' ? 'Number' : 'String';
      const stringValue = makeStringValue ? makeStringValue(value) : value.toString();

      accumulator[name] = {
        DataType: dataType,
        StringValue: stringValue
      };

      return accumulator;
    },
    {}
  );
}

public async sendMessage<T>({ attributes, messageBody, queueName }:
ISendMessageParams<T>): Promise<string> {
  const messageAttributes = this.makeSqsAttributes(attributes);
  const command = new SendMessageCommand({
    QueueUrl: this.queueUrl + queueName,
    MessageAttributes: messageAttributes,
    MessageBody: messageBody
  });
  const { MessageId: messageId } = await this.client.send(command);

  return messageId;
}

public async receiveMessage<T>(
  visibilityTimeoutInSeconds: number,
  queueName: string
): Promise<IReceiveMessagesResult<T> | null> {
  const command = new ReceiveMessageCommand({
    QueueUrl: this.queueUrl + queueName,
    VisibilityTimeout: visibilityTimeoutInSeconds,
  });

```

```

    MessageAttributeNames: ['All']
  });
  const { Messages: messages } = await this.client.send(command);

  if (!messages) {
    return null;
  }

  const [message] = messages;
  const { MessageAttributes: attributes, ReceiptHandle: handler } = message;

  const attributeEntries = Object.entries(attributes).map(([key, attributeValue]) => [
    key,
    attributeValue.StringValue
  ]);

  return {
    handler,
    attributes: Object.fromEntries(attributeEntries)
  };
}

public async deleteMessage(receiptHandle: string, queueName: string): Promise<void> {
  const command = new DeleteMessageCommand({
    QueueUrl: this.queueUrl + queueName,
    ReceiptHandle: receiptHandle
  });

  await this.client.send(command);
}

export interface IAuthenticationResult {
  accessToken: string;
  refreshToken: string;
  idToken: string;
  expiresIn: number;
}

export interface IUserAttributesMap {
  name: string;
  email: string;
  nickname: string;
  subscription: string;
  isEmailVerified: boolean;
}

export interface IRawUserAttributesMap {
  name: string;
  email: string;
  nickname: string;
  sub: string;
}

```

```

    email_verified: string;
  }

export interface IRefreshedAuthenticationResult {
  accessToken: string;
  idToken: string;
  expiresIn: number;
}

export interface IAttributes<T> {
  [k: string]: {
    value: T;
    makeStringValue?: (value: T) => string;
  };
}

export interface ISendMessageParams<T> {
  attributes: IAttributes<T>;
  messageBody: string;
  queueName: string;
}

export interface IReceiveMessagesResult<T = Record<string, string>> {
  attributes: T;
  handler: string;
}

export type TMessageDataType = 'String' | 'Number';

import { ApiEnvDto } from 'src/connectors/config/dtos/api-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

export const CONFIG_API_TOKEN = 'api-config';

const apiConf = {
  prefixes: {
    main: 'api'
  }
};

export default createConfigNamespace(CONFIG_API_TOKEN, apiConf, ApiEnvDto);

import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';
import { AppEnvDto } from 'src/connectors/config/dtos/app-env.dto';

export const CONFIG_APP_TOKEN = 'app-config';

export default createConfigNamespace(CONFIG_APP_TOKEN, process.env, AppEnvDto);

import { AwsEnvDto } from 'src/connectors/config/dtos/aws-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

```

```

export const CONFIG_AWS_TOKEN = 'aws-config';

export default createConfigNamespace(CONFIG_AWS_TOKEN, process.env, AwsEnvDto);

import { CognitoEnvDto } from 'src/connectors/config/dtos/cognito-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

export const CONFIG_COGNITO_TOKEN = 'cognito-config';

export default createConfigNamespace(CONFIG_COGNITO_TOKEN, process.env,
CognitoEnvDto);

import { DbEnvDto } from 'src/connectors/config/dtos/db-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

export const CONFIG_DB_TOKEN = 'db-config';

export default createConfigNamespace(CONFIG_DB_TOKEN, process.env, DbEnvDto);

import { S3EnvDto } from 'src/connectors/config/dtos/s3-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

export const CONFIG_S3_TOKEN = 's3-config';

export default createConfigNamespace(CONFIG_S3_TOKEN, process.env, S3EnvDto);
import { SqsEnvDto } from 'src/connectors/config/dtos/sqs-env.dto';
import { createConfigNamespace } from 'src/connectors/config/helpers/create-config-namespace';

export const CONFIG_SQS_TOKEN = 'sqs-config';

export default createConfigNamespace(CONFIG_SQS_TOKEN, process.env, SqsEnvDto);
import { Expose, Type } from 'class-transformer';
import { IsObject, IsString, ValidateNested } from 'class-validator';

class PrefixesDto {
  @Expose()
  @IsString()
  main;
}

export class ApiEnvDto {
  @Expose()
  @IsObject()
  @ValidateNested()
  @Type(() => PrefixesDto)
  prefixes: PrefixesDto;
}
import { Expose } from 'class-transformer';
import { IsEnum, IsNumberString, IsString } from 'class-validator';
import { EnvironmentTypes } from 'src/connectors/config/types/environmentTypes';

export class AppEnvDto {

```

```

@Expose()
@IsNumberString()
PORT: string;

@Expose()
@IsString()
HOST: string;

@Expose()
@IsEnum(EnvironmentTypes)
NODE_ENV: EnvironmentTypes;
}
import { Expose } from 'class-transformer';
import { IsString } from 'class-validator';

export class AwsEnvDto {
  @Expose()
  @IsString()
  ACCESS_KEY_ID: string;

  @Expose()
  @IsString()
  SECRET_ACCESS_KEY: string;
}
import { Expose } from 'class-transformer';
import { IsString } from 'class-validator';

export class CognitoEnvDto {
  @Expose()
  @IsString()
  COGNITO_POOL_ID: string;

  @Expose()
  @IsString()
  COGNITO_CLIENT_ID: string;

  // @Expose()
  // @IsString()
  // COGNITO_CLIENT_SECRET: string;

  @Expose()
  @IsString()
  COGNITO_REGION: string;
}
import { Expose } from 'class-transformer';
import { IsNumberString, IsString } from 'class-validator';

export class DbEnvDto {
  @Expose()
  @IsNumberString()
  DB_PORT: string;

```

```

@Expose()
@IsString()
DB_NAME: string;

@Expose()
@IsString()
DB_HOST: string;

@Expose()
@IsString()
DB_USER: string;

@Expose()
@IsString()
DB_PASSWORD: string;
}
import { Expose } from 'class-transformer';
import { IsString } from 'class-validator';

export class S3EnvDto {
  @Expose()
  @IsString()
  S3_REGION: string;

  @Expose()
  @IsString()
  S3_NAME: string;
}
import { Expose } from 'class-transformer';
import { IsString } from 'class-validator';

export class SqsEnvDto {
  @Expose()
  @IsString()
  SQS_URL: string;

  @Expose()
  @IsString()
  SQS_REGION: string;

  @Expose()
  @IsString()
  SQS_ACCESS_KEY_ID: string;

  @Expose()
  @IsString()
  SQS_SECRET_ACCESS_KEY: string;

  @Expose()
  @IsString()
  IMAGE_QUEUE_NAME: string;
}

```

```

import { ConfigObject } from '@nestjsjs/config/dist/types';
import { registerAs } from '@nestjsjs/config/dist/utils/register-as.util';
import { ClassConstructor, plainToInstance } from 'class-transformer';
import { validateConfig } from 'src/connectors/config/helpers/validate-config.helper';
import { JsonObject } from 'src/types/json-object.types';

export function createConfigNamespace<T extends ConfigObject>(
  name: string,
  configData: JsonObject,
  classConstructor: ClassConstructor<T>
) {
  return registerAs(name, () => {
    const config = plainToInstance(classConstructor, configData, { strategy: 'excludeAll' });
    validateConfig(config);

    return config;
  });
}
import { ValidationError } from '@nestjsjs/common';
import { validateSync } from 'class-validator';

export function validateConfig(obj: object) {
  const errors = validateSync(obj, {
    forbidNonWhitelisted: true
  });

  if (errors.length) {
    throw new Error(JSON.stringify(getErrorsConstraintsRecursive(errors), null, 2));
  }
}

function getErrorsConstraintsRecursive(errors: ValidationError[], constraints: Record<string, string>[] = []) {
  errors.forEach(error => {
    constraints.push(error.constraints);

    if (error.children) {
      getErrorsConstraintsRecursive(error.children, constraints);
    }
  });

  return constraints;
}
import { Module } from '@nestjsjs/common';
import { ConfigModule } from '@nestjsjs/config';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';
import apiConfigNamespace from 'src/connectors/config/config-namespaces/api.config-namespace';
import appConfigNamespace from 'src/connectors/config/config-namespaces/app.config-namespace';
import awsConfigNamespace from 'src/connectors/config/config-namespaces/aws.config-namespace';

```

```

import cognitoConfigNamespace from 'src/connectors/config/config-namespaces/cognito.config-
namespace';
import dbConfigNamespace from 'src/connectors/config/config-namespaces/db.config-
namespace';
import s3ConfigNamespace from 'src/connectors/config/config-namespaces/s3.config-namespace';
import sqsConfigNamespace from 'src/connectors/config/config-namespaces/sqs.config-
namespace';

@Module({
  imports: [
    ConfigModule.forRoot({
      load: [
        appConfigNamespace,
        apiConfigNamespace,
        dbConfigNamespace,
        cognitoConfigNamespace,
        s3ConfigNamespace,
        sqsConfigNamespace,
        awsConfigNamespace
      ]
    })
  ],
  providers: [ConfigConnectorService],
  exports: [ConfigConnectorService]
})
export class ConfigConnectorModule {}
import { Inject, Injectable } from '@nestjs/common';
import { ConfigType } from '@nestjs/config';
import apiConfigNamespace from 'src/connectors/config/config-namespaces/api.config-
namespace';
import appConfigNamespace from 'src/connectors/config/config-namespaces/app.config-
namespace';
import awsConfigNamespace from 'src/connectors/config/config-namespaces/aws.config-
namespace';
import cognitoConfigNamespace from 'src/connectors/config/config-namespaces/cognito.config-
namespace';
import dbConfigNamespace from 'src/connectors/config/config-namespaces/db.config-
namespace';
import s3ConfigNamespace from 'src/connectors/config/config-namespaces/s3.config-namespace';
import sqsConfigNamespace from 'src/connectors/config/config-namespaces/sqs.config-
namespace';
import { ApiEnvDto } from 'src/connectors/config/dtos/api-env.dto';
import { AppEnvDto } from 'src/connectors/config/dtos/app-env.dto';
import { AwsEnvDto } from 'src/connectors/config/dtos/aws-env.dto';
import { CognitoEnvDto } from 'src/connectors/config/dtos/cognito-env.dto';
import { DbEnvDto } from 'src/connectors/config/dtos/db-env.dto';
import { S3EnvDto } from 'src/connectors/config/dtos/s3-env.dto';
import { SqsEnvDto } from 'src/connectors/config/dtos/sqs-env.dto';

@Injectable()
export class ConfigConnectorService {
  constructor(

```

```

@Inject(apiConfigNamespace.KEY)
private apiConfig: ConfigType<typeof apiConfigNamespace>,
@Inject(dbConfigNamespace.KEY)
private dbConfig: ConfigType<typeof dbConfigNamespace>,
@Inject(appConfigNamespace.KEY)
private appConfig: ConfigType<typeof appConfigNamespace>,
@Inject(cognitoConfigNamespace.KEY)
private cognitoConfig: ConfigType<typeof cognitoConfigNamespace>,
@Inject(s3ConfigNamespace.KEY)
private s3Config: ConfigType<typeof s3ConfigNamespace>,
@Inject(awsConfigNamespace.KEY)
private awsConfig: ConfigType<typeof awsConfigNamespace>,
@Inject(sqsConfigNamespace.KEY)
private sqsConfig: ConfigType<typeof sqsConfigNamespace>
) {}

getDbConfig(): DbEnvDto {
  return this.dbConfig;
}

getAppConfig(): AppEnvDto {
  return this.appConfig;
}

getApiConfig(): ApiEnvDto {
  return this.apiConfig;
}

getCognitoConfig(): CognitoEnvDto {
  return this.cognitoConfig;
}

getS3Config(): S3EnvDto {
  return this.s3Config;
}

getAwsConfig(): AwsEnvDto {
  return this.awsConfig;
}

getSqsConfig(): SqsEnvDto {
  return this.sqsConfig;
}
}
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { ConfigConnectorModule } from 'src/connectors/config/config-connector.module';
import { ConfigConnectorService } from 'src/connectors/config/config-connector.service';

@Module({
  imports: [
    TypeOrmModule.forRootAsync({

```

```

imports: [ConfigConnectorModule],
inject: [ConfigConnectorService],
useFactory: (configConnectorService: ConfigConnectorService) => {
  const dbConfig = configConnectorService.getDbConfig();

  return {
    type: 'postgres',
    host: dbConfig.DB_HOST,
    port: parseInt(dbConfig.DB_PORT),
    username: dbConfig.DB_USER,
    password: dbConfig.DB_PASSWORD,
    database: dbConfig.DB_NAME,
    autoLoadEntities: true
  };
}
})
]
})
export class DatabaseConnectorModule {}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

РОЗРОБКА ВЕБСАЙТУ ДЛЯ КОМПРЕСІЇ ЗОБРАЖЕНЬ

ст. групи КНТ-111сп

к.т.н., доцент

Діана ЖЕВЕДЬ
Аліна КАЗУРОВА

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для компресії зображень.

Предмет дослідження – програмні засоби для обробки зображень.

Метою роботи є розробка програмного забезпечення для обробки даних зображень.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для компресії зображень;
- здійснити проектування вебсайту для компресії зображень;
- створити вебсайт для компресії зображень;
- виконати тестування вебсайту для компресії зображень

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Com press Now	JPEG Opti mizer	Tiny PNG	Opti mizill a	JPEG .io	Imag eRec ycle	Com press Ezgif	Com press Ezgif	Dy na mic Dri ve
Зручність використання	+—	+	+—	+—	+	+—	+—	+	+—
Відсоток зменшення у файлі .jpg	99,9%	95,3%	92%	91%	79,5%	92,7%	95%	92,7%	93%
Відсоткове зменшення у файлі .png	99,9%	—	83%	79%	86,9%	80,4%	78%	24,5%	77%
Можливість завантаження різних типів файлів для стиснення	+	+	+—	+	+	+	+	+	+—
Можливість отримання файлів різних типів в якості результату	—	+	—	+	+	+	+	+	+—
Різноманітність налаштувань перед початком стиснення	+—	+	+	+—	+	+	+—	+	+

Рисунок В.4 – Слайд 4

Обґрунтування вибору мови програмування

Критерій порівняння мов програмування	Мова програмування		
	JavaScript	Python	Ruby
Зручність кодування	+	+-	+
Багатозадачність та паралельність	+	+	+-
Динамічна типізація	+	+	-
Зручність для створення ПЗ для стиснення зображень	+	-	-
Поширеність та спільнота розробників	+	+	+-

Рисунок В.5 – Слайд 5

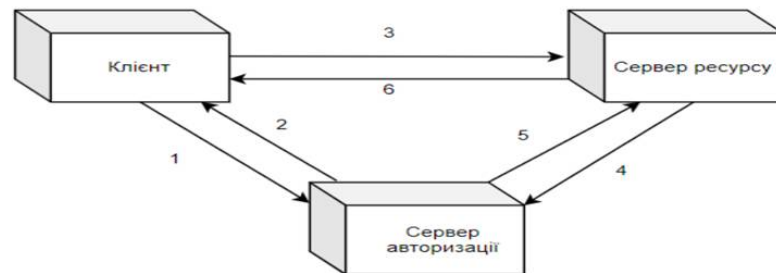
Вибір фреймворку для розробки

Критерій порівняння фреймворків	Фреймворки		
	React	Vue	Angular
Зручність використання	+	-	-
Швидкість виконання запитів	+	+	+-
Широкий вибір вбудованих та сторонніх розширень	+	+-	+-
Підтримка REST-протоколу	+	+-	+-
Використання великою спільнотою	+	+-	+

6

Рисунок В.6 – Слайд 6

Структура програмного забезпечення для стиснення зображень



1. Клієнт надсилає запит на сервер авторизації для отримання веб-токена JSON.
2. Сервер авторизації перевіряє ідентифікатор клієнта та секретний ключ і повертає маркер.
3. Клієнт надсилає запит на сервер для отримання необхідної інформації, додаючи маркер до заголовка запиту.
4. Потім сервер ресурсів надсилає запит на сервер авторизації для перевірки маркера.
5. Сервер авторизації надсилає відповідь, чи маркер дійсний чи недійсний.
6. Після отримання правильної інформації Сервер ресурсів забезпечує доступ до клієнтів.

Рисунок В.7 – Слайд 7

Схема функціонування програмного забезпечення

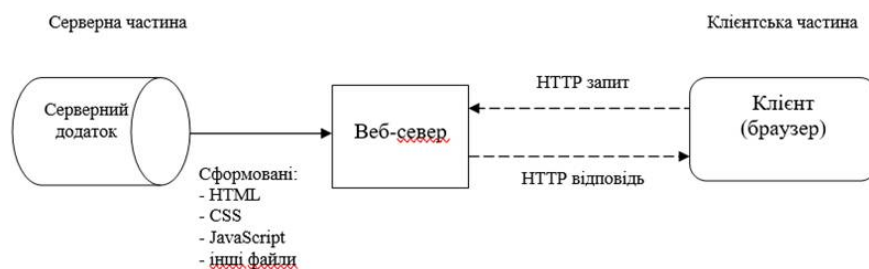


Рисунок В.8 – Слайд 8

Структура серверної частини програмного забезпечення для стиснення зображень на основі патерна MVC

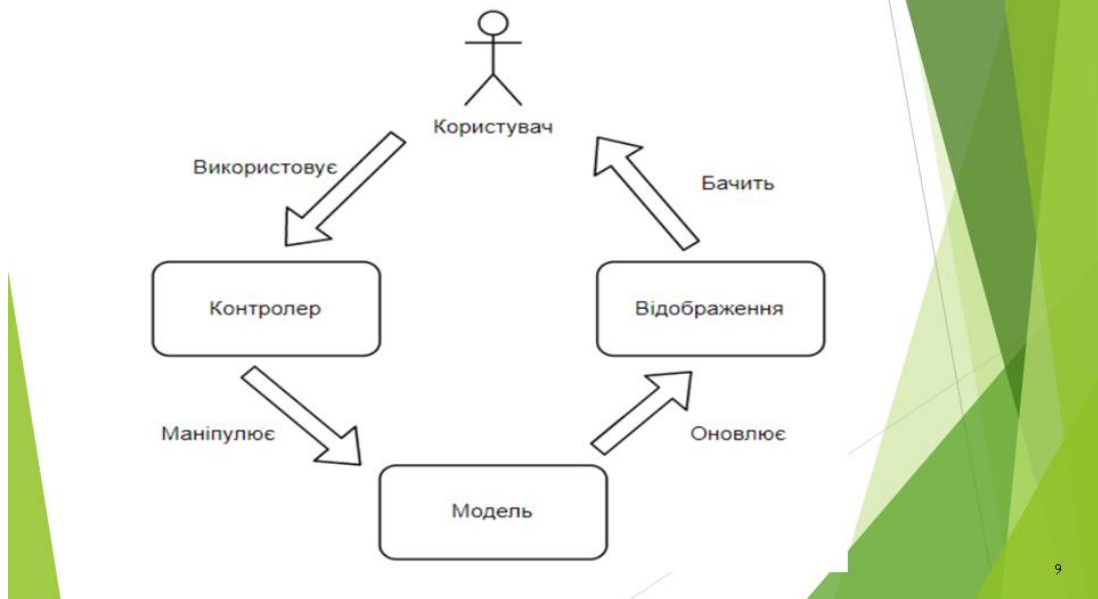


Рисунок В.9 – Слайд 9

Робота з програмою. Початкова сторінка веб-сайту для стиснення зображень.

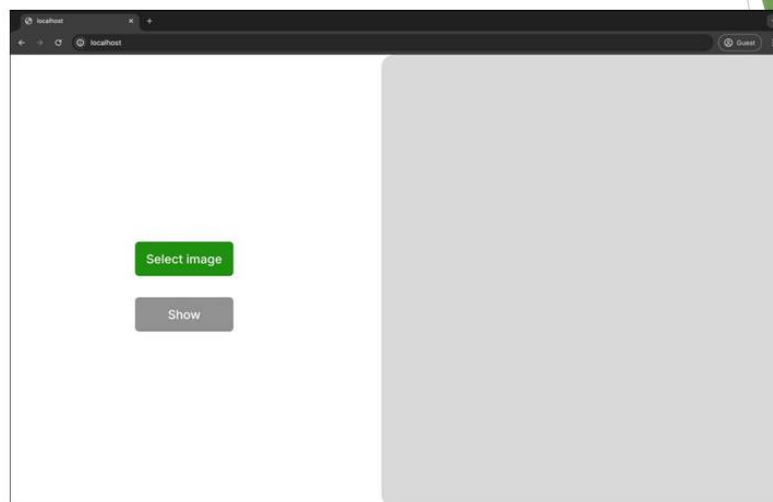


Рисунок В.10 – Слайд 10

Робота з програмою. Вікно вибору файлів користувача з його персонального диска

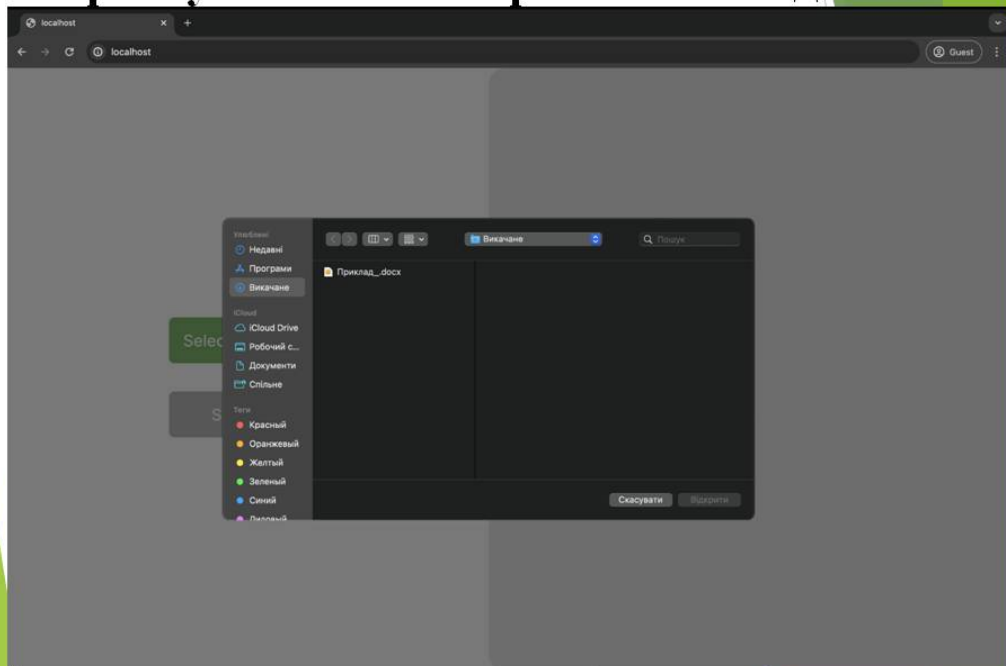


Рисунок В.11 – Слайд 11

Робота з програмою. Відображення результату стиснення файлу.

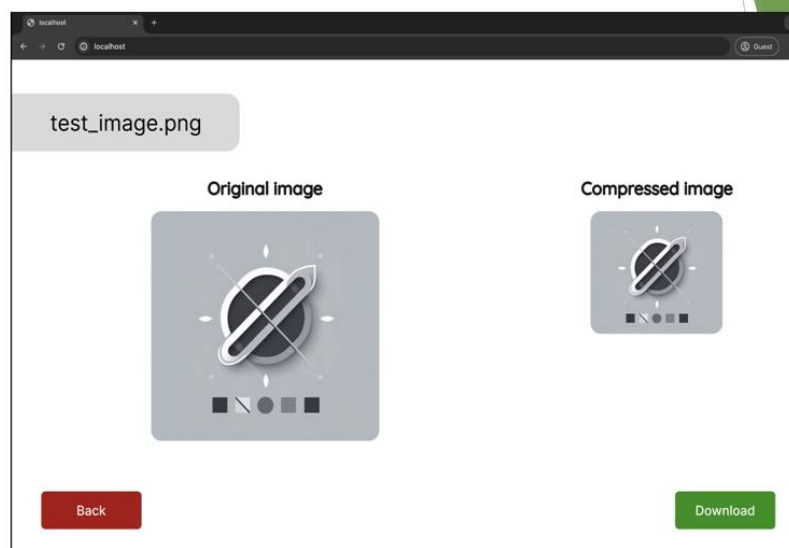


Рисунок В.12 – Слайд 12

Висновки

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для компресії зображень.

Сформульовано функціональні вимоги до програмного забезпечення для компресії зображень. Для створення програмного забезпечення для компресії зображень обрано мову програмування JavaScript та фреймворк React.

Розроблено програмне забезпечення для компресії зображень. Запропоновано структуру та описано основні модулі програмного забезпечення компресії зображень. Виконано проектування інтерфейсу взаємодії користувача з програмним забезпеченням для компресії зображень. Виконано тестування розробленого програмного забезпечення для компресії зображень. За результатами тестування програмного забезпечення для компресії зображень було встановлено, що програма виконує функції та вимоги, зазначені у технічному завданні, та може використовуватися за призначенням.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

Рисунок В.13 – Слайд 13