Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ З РЕДАГУВАННЯ ЗОБРАЖЕНЬ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Куделя В.О.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Куделі Вікторії Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебзастосунку з редагування зображень

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, дані інтернет-мережі, мова програмування JavaScript, бібліотека для створення користувацьких інтерфейсів React, середовище розробки Microsoft Visual Studio Code.

4. Перелік питань, що потрібно опрацювати в роботі
1. Проблематика задачі редагування зображень.
2. Моделі, методи та алгоритми редагування зображень.
3. Розроблення вебзастосунку з редагування зображень.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми редагування зображень, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз технічних і програмних засобів	21.04.23-30.04.23	
5	Розробка методу	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	05.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____проф. Машталір С.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 68 с., 22 рис., 31 джерело.

ВЕБЗАСТОСУНОК, РЕДАГУВАННЯ ЗОБРАЖЕНЬ, REACT, NODE.JS, VISUAL STUDIO CODE.

Об’єктом роботи є методи, які дозволяють редагувати зображення.

Метою роботи є розробка зручного та ефективного вебзастосунку, що надасть можливість користувачам просто та швидко виконувати базові операції з редагування зображень в онлайн режимі, не потребуючи встановлення та запуску складних програм на власному комп’ютері.

Для досягнення мети було використано комбінацію зовнішніх технологій, таких як HTML, CSS і JavaScript, щоб створити інтуїтивно зрозумілий інтерфейс користувача. Також було використано бібліотеки та фреймворки для покращення функціональності та забезпечення плавної взаємодії користувача.

У результаті роботи здійснена програмна реалізація вебзастосунку з редагування зображень.

WEB APPLICATION, IMAGE EDITING, REACT, NODE.JS, VISUAL STUDIO CODE.

The object of the work is the methods that allow you to edit images.

The goal of the work is to develop a convenient and effective web application that will enable users to easily and quickly perform basic image editing operations online, without the need to install and run complex programs on their own computer.

To achieve the goal, a combination of external technologies such as HTML, CSS and JavaScript were used to create an intuitive user interface. Libraries and frameworks have also been used to improve functionality and provide a smooth user experience.

As a result of the work, a software implementation of a web application for image editing was carried out.

ЗМІСТ

ВСТУП

У сучасну епоху цифрових технологій візуальний контент відіграє вирішальну роль у комунікації та маркетингу. Можливість редагувати та покращувати зображення є важливою як для компаній, так і для окремих осіб, оскільки це дозволяє їм створювати високоякісний візуальний вміст, який може захопити та утримати увагу цільової аудиторії. Для задоволення цієї потреби розробка вебзастосунків для редагування зображень стає все більш важливою.

У даній роботі будуть розглянуті моделі, методи та алгоритми редагування зображень, а також інструменти та технології для розробки вебзастосунків. Далі буде описано архітектуру вебзастосунку та його складові частини, розроблення та інтеграція модулів редагування зображень, оцінка продуктивності вебзастосунку та виявлення можливих проблем.

Таким чином, дана робота має на меті стати важливим внеском у сферу обробки та редагування зображень та має велике практичне значення для користувачів, які щодня працюють з великою кількістю фотографій та графічних зображень.

1 ПРОБЛЕМАТИКА ЗАДАЧІ РЕДАГУВАННЯ ЗОБРАЖЕНЬ

1.1 Актуальність задачі

В цей час завдання розробки вебзастосунку для редагування зображень є більш важливим, ніж будь-коли. У світі, де мільярди зображень завантажуються в Інтернет щодня, здатність легко й ефективно редагувати зображення є важливою задачею як для компаній, так і для окремих осіб. Це пояснюється тим, що високоякісний візуальний вміст має вирішальне значення для ефективної комунікації та маркетингу, а інструменти редагування зображень можуть допомогти досягти цього.

Вплив соціальних медіа та інших цифрових платформ означає, що зображення стали основним засобом комунікації, що робить життєво важливим для окремих людей і компаній створювати візуально привабливий контент, який може виділятися в багатолюдному цифровому ландшафті. Інструменти редагування зображень дозволяють користувачам регулювати кольори, контраст та інші візуальні елементи зображення, дозволяючи їм створювати привабливі візуальні ефекти, які можуть привернути увагу аудиторії. Це може бути особливо важливим для компаній, яким необхідно професійно та ретельно представити свої продукти чи послуги.

Дослідження Shutterstock, одного з найбільших майданчиків для продажу стокових зображень, показали, що оброблені та відредаговані зображення популярніші та краще продаються, що вказує на важливість редагування зображень у маркетингу та рекламі. Створення візуально привабливого вмісту може збільшити конверсію на вебсайті до 86%, як зазначено в дослідженні MDG Advertising. У таких галузях, як дизайн, фотографія, медицина та технології, редагування зображень також є важливим завданням, яке потребує спеціальних інструментів і досвіду [1].

Вебзастосунки для редагування зображень можуть бути корисними для широкого кола людей, від особистих блогерів і користувачів соціальних

мереж до студентів і дослідників, яким потрібні зображення для своїх проєктів або презентацій. Надаючи зручний і доступний інструмент для редагування зображень, вебзастосунок може допомогти користувачам створювати високоякісний візуальний вміст, який зможе привернути увагу аудиторії та ефективно донести своє повідомлення. Приклад покращення зображення за допомогою інструментів редагування зображень показано на рисунку 1.1 [2].

Рисунок 1.1 – Приклад обробки зображення

Підсумовуючи, завдання розробки вебзастосунку для редагування зображень є актуальним і важливим у сучасну цифрову епоху. Інструменти редагування зображень можуть допомогти окремим особам і компаніям створювати візуально привабливий вміст, а також підвищити ефективність їхніх комунікаційних і маркетингових зусиль. Надаючи зручний і доступний інструмент для редагування зображень, вебзастосунок може допомогти користувачам досягти їхніх творчих цілей і покращити якість візуального вмісту.

1.2 Аналіз потреб користувачів

Аналіз потреб користувачів є важливою складовою у розробці будь-якого продукту, в тому числі й вебзастосунків з редагування зображень. Аналіз потреб користувачів включає визначення та розуміння конкретних вимог і переваг цільової групи користувачів.

Першим кроком в аналізі потреб користувачів є визначення демографічних показників цільової групи користувачів. Це включає такі фактори, як вік, стать, освіта, професія та рівень досвіду редагування зображень. Демографічні дані цільової групи користувачів впливатимуть на функції, які необхідно включити у вебзастосунок.

Згідно з опитуванням, проведеним компанією Adobe, більшість користувачів фоторедакторів – люди у віці від 18 до 34 років, причому 52% респондентів належать до цієї вікової групи. Опитування також показало, що жінки частіше користуються інструментами для редагування фотографій, ніж чоловіки: 53% жінок повідомили, що редагують свої фотографії, порівняно з 47% чоловіків.

Іншим важливим аспектом аналізу потреб користувачів є розуміння цілей користувачів у використанні вебзастосунку для редагування зображень. Це включає визначення конкретних завдань, які користувачі хочуть виконувати, типи зображень, які вони хочуть редагувати, і рівень складності, який їм подобається. Наприклад, користувачі-любителі можуть захотіти виконувати базові завдання редагування, такі як обрізка та зміна розміру, тоді як професійні користувачі можуть потребувати розширених функцій, таких як корекція кольорів і маніпуляції зображеннями.

Середовище, в якому користувачі матимуть доступ до вебзастосунку, також є важливим фактором. Це включає такі фактори, як тип використовуваного пристрою, розмір і роздільна здатність екрана, швидкість Інтернету та операційна система. Вебзастосунок потрібно оптимізувати, щоб

забезпечити безперебійну роботу користувачів на різних пристроях і платформах.

Розуміння вподобань користувачів є ще одним важливим аспектом аналізу потреб користувачів. Це включає визначення бажаного дизайну інтерфейсу користувача, колірної схеми та макета застосунку. Налаштування користувача також поширюються на простоту використання програми, швидкість роботи та доступний рівень налаштування.

Збір відгуків користувачів є постійним процесом у розробці вебзастосунку для редагування зображень. Зворотний зв’язок можна отримати за допомогою опитувань користувачів, фокус-груп або сеансів тестування користувачів. Відгуки користувачів надають цінну інформацію про взаємодію з користувачем, яку можна використовувати для покращення функцій вебзастосунку.

Доступність – це важливий аспект аналізу потреб користувачів, який не можна забувати. Важливо враховувати користувачів з обмеженими можливостями, такими як порушення зору або моторики, і переконатися, що вебзастосунок доступний для них. Вебзастосунок має бути розроблено відповідно до Рекомендацій щодо доступності вебвмісту (WCAG), які містять рекомендації щодо підвищення доступності вебвмісту.

Аналіз конкуренції є важливим аспектом аналізу потреб користувачів. Це передбачає ідентифікацію існуючих вебзастосунків для редагування зображень і оцінку їхніх можливостей і функцій. Розуміння сильних і слабких сторін існуючих програм може допомогти розробникам виявити прогалини на ринку та розробити вебзастосунок, який пропонує унікальні функції та переваги.

Вартість – це ще один важливий момент при аналізі потреб користувачів. Користувачі можуть мати різні бюджети та очікування щодо вартості використання вебзастосунку. Вебзастосунок має пропонувати модель ціноутворення, яка є прозорою та доступною, водночас пропонуючи необхідні функції для задоволення потреб користувача.

Локалізація є ще одним важливим аспектом аналізу потреб користувачів, особливо якщо вебзастосунок призначений для глобальної аудиторії. Локалізація передбачає адаптацію вебзастосунку до конкретних потреб і вподобань різних культурних і мовних груп. Це включає підтримку мови, конвертацію валюти та місцеві варіанти оплати.

Підсумовуючи, аналіз потреб користувачів є важливим кроком у розробці вебзастосунку для редагування зображень. Розуміння демографічних показників, цілей, середовища, уподобань, доступності, конкуренції, вартості, локалізації та відгуків потенційних користувачів має важливе значення для розробки вебзастосунку, яка відповідає конкретним потребам користувачів. Аналізуючи потреби користувачів, розробники можуть створити вебзастосунок, оптимізований для продуктивності, зручний для користувача та здатний забезпечувати високоякісні результати [3, 4].

1.3 Опис функціональних вимог до вебзастосунку

Функціональні вимоги до вебзастосунку залежать від конкретного застосування та потреб користувачів. Проте, існує декілька загальних вимог, які повинен відповідати будь-який вебзастосунок з редагування зображень.

Імпорт і експорт зображень: програма повинна дозволяти користувачам імпортувати зображення зі своїх локальних пристроїв, хмарних сховищ або платформ соціальних мереж, а також експортувати відредаговані зображення в різних форматах, таких як JPEG, PNG і TIFF.

Базові інструменти редагування: програма має надавати базові інструменти редагування, такі як обрізання, зміна розміру, обертання та перевертання, що дозволяє користувачам легко коригувати свої зображення. На рисунку 1.2 продемонстровано використання інструменту обрізання.

Рисунок 1.2 – Приклад використання інструменту обрізання

Розширені інструменти редагування: програма має надавати ряд розширених інструментів редагування, таких як корекція кольору, налаштування експозиції, підвищення контрастності та підвищення різкості зображення, що дозволяє користувачам вносити складніші зміни до своїх зображень. На рисунку 1.3 продемонстровано приклад підвищення контрастності.

Спеціальні ефекти та фільтри: програма має надавати низку спеціальних ефектів і фільтрів, таких як розмиття, віньєтка, чорно-білий колір і сепія, що дозволяє користувачам додавати художній стиль своїм зображенням. На рисунку 1.4 продемонстровано використання чорно-білого фільтра.

Текст: програма повинна дозволяти користувачам додавати текст до своїх зображень із параметрами стилю шрифту, розміру, кольору та вирівнювання. На рисунку 1.5 продемонстровано додавання тексту на зображення.

Рисунок 1.3 – Приклад використання інструменту підвищення контрастності

Рисунок 1.4 – Приклад використання чорно-білого фільтра

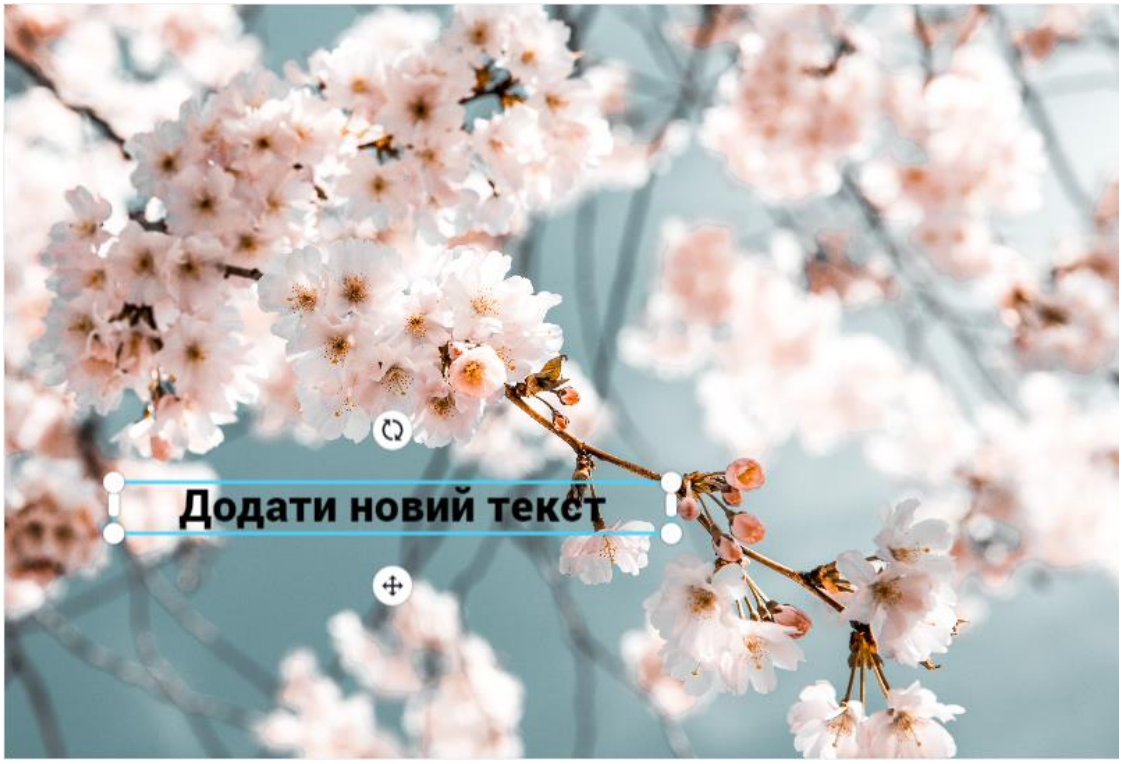

Рисунок 1.5 – Приклад додавання тексту на зображення

Скасувати та повторити: програма має дозволяти користувачам скасовувати та повторювати свої дії, надаючи захист у разі помилок.

Збереження та спільний доступ: програма має дозволяти користувачам зберігати свої відредаговані зображення та ділитися ними на платформах соціальних мереж, таких як Facebook, Twitter та Instagram, або надсилати їх друзям і родині електронною поштою.

Продуктивність: програму слід розробляти з урахуванням продуктивності, щоб вона могла обробляти великі обсяги користувачів і зображень без шкоди для швидкості чи функціональності.

Таким чином, вебзастосунок для редагування зображень має надавати ряд потужних і зручних інструментів редагування, підтримку шарів і масок, функції скасування та повторення, а також параметри для збереження та обміну зображеннями. Він також має бути розроблений з урахуванням продуктивності. Задовольняючи ці функціональні вимоги, програма може надати користувачам позитивний і задовільний досвід редагування зображень.

Щоб переконатися, що вебзастосунок відповідає цим функціональним вимогам, важливо дотримуватися орієнтованого на користувача підходу до проєктування, який передбачає взаємодію з потенційними користувачами протягом усього процесу розробки. Це може включати проведення досліджень користувачів, тестування зручності використання та збір відгуків від користувачів для покращення дизайну та функціональності програми.

Наприклад, згідно з дослідженням компанії Adobe, більшість користувачів хочуть мати можливість змінювати кольори та насиченість зображень у редакторі зображень. А згідно з опитуванням від Stack Overflow, зміна розміру та обрізка зображень є одними з найбільш популярних операцій з редактором зображень.

Також слід зазначити, що відповідно до опитування, проведеного Crello, платформою графічного дизайну, 60% власників малого бізнесу повідомили, що вони використовують візуальні засоби для просування своїх продуктів або послуг, і 59% з них вважають, що візуальні ефекти є вирішальними для успіху їх бренду. Крім того, це ж опитування показало, що 85% респондентів використовують візуальний контент у своєму маркетингу в соціальних мережах, що вказує на важливість візуального контенту в онлайн-комунікації.

Користувачі цінують можливість легко та швидко редагувати зображення. В опитуванні, проведеному компанією Adobe, 67% респондентів повідомили, що вони редагують свої фотографії, перш ніж поділитися ними, а 52% із них сказали, що хотіли б бачити більше автоматизації та опцій редагування одним клацанням миші у своєму програмному забезпеченні для редагування фотографій.

Підсумовуючи, функціональні вимоги до вебзастосунку для редагування зображень зосереджені на наданні ряду потужних і зручних інструментів редагування.

Враховуючи ці висновки, стає зрозуміло, що зростає попит на потужні та зручні інструменти для редагування зображень. Вебзастосунок, який може

задовольнити ці вимоги, надаючи ряд інструментів для редагування та простий, інтуїтивно зрозумілий інтерфейс, матиме хороші можливості для досягнення успіху на ринку.

1.4 Порівняння розробки програмного забезпечення та вебзастосунку

Розробка програмного забезпечення та веброзробка – це дві різні галузі зі своїми унікальними наборами проблем і вимог. У той час як розробка програмного забезпечення передбачає створення автономних програм, які запускаються на комп’ютері або пристрої користувача, веброзробка зосереджена на створенні програм, до яких можна отримати доступ через веббраузер.

Розробка вебзастосунку має кілька переваг перед традиційним програмним забезпеченням. Одна з головних переваг полягає в тому, що доступ до вебзастосунку можна отримати з будь-якого місця, де є підключення до Інтернету, що робить їх більш доступними та зручними для користувачів. Крім того, вебзастосунки легше розповсюджувати та оновлювати, оскільки в програму можна вносити зміни на стороні сервера, не вимагаючи від користувачів завантажувати та встановлювати оновлення.

Ще однією перевагою веброзробки є використання стандартизованих вебтехнологій, таких як HTML, CSS і JavaScript. Ці технології широко підтримуються та мають великі спільноти розробників, що полегшує пошук ресурсів і підтримки для проєктів веброзробки. Крім того, фреймворки веброзробки, такі як React і Angular, надають готові компоненти та інструменти, які можуть прискорити процес розробки.

Навпаки, розробка програмного забезпечення вимагає складнішого середовища розробки, і його розповсюдження та обслуговування може бути складнішим. Крім того, для програмного забезпечення можуть знадобитися

різні версії для різних операційних систем, що ускладнює розробку та розповсюдження.

Також слід зазначити, що веброзробка дозволяє інтегрувати соціальні медіа та інструменти для співпраці, що полегшує користувачам спільний доступ до проєктів редагування зображень і співпрацю в них. Це особливо важливо для сучасних користувачів, які часто працюють віддалено або співпрацюють з членами команди в різних місцях.

Ще однією перевагою веброзробки є можливість скористатися перевагами хмарних обчислень і зберігання. Розміщуючи програму на хмарних серверах, користувачі можуть легко зберігати та отримувати доступ до своїх зображень із будь-якого пристрою, не турбуючись про те, що на локальному пристрої вичерпається простір для зберігання. Це може бути особливо корисно для користувачів, які працюють з великими файлами зображень.

Крім того, веброзробка дозволяє створювати адаптивні та зручні для мобільних пристроїв програми. Зі збільшенням використання мобільних пристроїв для доступу до Інтернету важливо, щоб вебзастосунок був оптимізовані для мобільних екранів. Розробляючи вебзастосунок, можна зробити інструмент редагування зображень доступним і функціональним на різних пристроях і розмірах екрана.

Враховуючи ці фактори, розробка вебзастосунку для редагування зображень пропонує кілька переваг перед традиційною розробкою програмного забезпечення. Доступ до вебзастосунку можна отримати з будь-якого місця, де є підключення до Інтернету, що робить її більш доступною та зручною для користувачів. Використання стандартизованих вебтехнологій і фреймворків розробки також може пришвидшити процес розробки та полегшити підтримку та оновлення застосунку з часом.

1.5 Постановка задачі

Підсумовуючи вище сказане, можна зробити висновок, що проблема редагування зображень стає все більш актуальною в сучасному світі. Завдяки широкому поширенню цифрових технологій та зростанню кількості зображень, які потрібно редагувати, з’являється потреба у більш ефективних та зручних інструментах для обробки зображень. Тому розробка вебзастосунку з редагування зображень є важливою задачею, яка може сприяти полегшенню процесу обробки та підвищенню якості зображень.

Об’єктом роботи є методи, які дозволяють редагувати зображення.

Метою роботи є розробка зручного та ефективного вебзастосунку, що надасть можливість користувачам просто та швидко виконувати базові операції з редагування зображень в онлайн режимі, не потребуючи встановлення та запуску складних програм на власному комп’ютері.

Для досягнення мети роботи необхідно вирішити наступні завдання.

– розробити зручний та інтуїтивно зрозумілий інтерфейс користувача, який дозволить швидко та ефективно виконувати завдання по редагуванню зображень;

– реалізувати функціонал для редагування зображень, який буде містити різноманітні інструменти для обробки зображень, такі як зміна розміру зображення, зміна колірної гами, вирівнювання, обрізання, додавання тексту та інші;

– забезпечити високу швидкість та продуктивність вебзастосунку, щоб користувачі могли швидко та ефективно виконувати завдання по редагуванню зображень;

– забезпечити можливість зберігання зображень, щоб користувачі могли легко зберігати свої роботи;

– провести тестування вебзастосунку для підтвердження його ефективності та функціональності.

2 МОДЕЛІ, МЕТОДИ ТА АЛГОРИТМИ РЕДАГУВАННЯ ЗОБРАЖЕНЬ

2.1 Методи та технології для редагування зображень

Редагування зображень стосується процесу маніпулювання цифровими зображеннями для покращення або зміни їх зовнішнього вигляду. Це важливий аспект сучасних цифрових засобів масової інформації та має багато застосувань у таких сферах, як графічний дизайн, фотографія та цифрове мистецтво. Редагування зображень включає різні методи та технології, які дозволяють користувачам змінювати, покращувати або маніпулювати цифровими зображеннями. У цьому підрозділі будуть розглянуті різні методи та технології, які використовуються для редагування зображень та будуть реалізовані пізніше.

Методи, які будуть реалізовані, можна поділити на чотири категорії: маніпулювання зображенням, візуальне коригування, фільтри та обробка тексту.

Маніпулювання зображеннями є фундаментальним аспектом програм для редагування зображень, що дозволяє користувачам змінювати загальний вигляд і структуру зображень. У вебзастосунку буде реалізовано кілька методів і технологій, які забезпечують функції маніпулювання зображеннями. Далі буде детально описано можливості маніпулювання зображеннями, які будуть реалізовані [5–8].

Зміна розміру та обрізання є одними з основних операцій обробки зображень, які складають ядро багатьох програм для редагування зображень. Ці функції дають змогу користувачам налаштовувати розміри та композицію зображень відповідно до їхніх конкретних потреб. У вебзастосунку буде реалізовано передові методи та технології, щоб забезпечити надійні можливості зміни розміру та обрізання.

Зміна розміру зображення передбачає зміну його розміру, зберігаючи початкове співвідношення сторін. Ця операція необхідна для адаптації зображень до різних контекстів, таких як відображення в Інтернеті, друк або спільний доступ у соціальних мережах. Вебзастосунок буде використовувати розширені алгоритми, які використовують математичні розрахунки для точного масштабування зображень. Застосунок наддасть користувачам гнучкість у введенні власних значень ширини та висоти. Використовуючи ці параметри користувачі можуть точно контролювати розмір своїх зображень і пристосовувати їх до запланованих цілей.

Під час зміни розміру зображення збереження співвідношення сторін має вирішальне значення, щоб уникнути спотворень і зберегти оригінальні пропорції зображення. У застосунку будуть використовуватися спеціальні алгоритми, які автоматично регулюють розміри, зберігаючи співвідношення сторін. Це означає, що користувачам потрібно вказати лише один розмір (ширину або висоту), а програма обчислить інший розмір відповідно. Це автоматизоване збереження співвідношення сторін спростить процес зміни розміру та допоможе користувачам підтримувати візуальну цілісність своїх зображень.

Лістинг 2.1 Реалізація алгоритму збереження співвідношення сторін:

```
const { width, height } = this.state;
const originalWidth = this.props.currentWidth;
const originalHeight = this.props.currentHeight;
const aspectRatio = originalWidth / originalHeight;
let resizedWidth, resizedHeight;
if (width && !height) {
resizedWidth = parseInt(width);
resizedHeight = Math.round(resizedWidth / aspectRatio);
} else if (!width && height) {
resizedHeight = parseInt(height);
```

```
resizedWidth = Math.round(resizedHeight * aspectRatio);
} else {
resizedWidth = parseInt(width);
resizedHeight = parseInt(height);
}
this.props.submitResizeValues(resizedWidth, resizedHeight);
```

Обрізання зображення дає змогу користувачам вибирати конкретні цікаві області на зображенні та відкидати решту. Це потужний інструмент для видалення небажаних частин, фокусування на конкретних деталях або коригування композиції зображення. Вебзастосунок наддасть інтуїтивно зрозумілі та інтерактивні інструменти обрізання для визначення потрібної області обрізання. Користувачі зможуть використовувати маркери, які можна перетягувати та прямокутне поле вибору, щоб точно вибрати область, яку вони хочуть зберегти. Цей механізм візуального зворотного зв’язку покращує взаємодію з користувачем, забезпечуючи точне та інтуїтивно зрозуміле обрізання зображення.

Після визначення області обрізання вебзастосунок виконує необхідні обчислення, щоб витягти вибрану область як нове зображення. Основна технологія аналізує піксельні дані в межах визначеної області кадрування та створює нове зображення на основі цієї інформації. Відокремлюючи потрібну область від решти зображення, користувачі можуть отримати бажану композицію та видалити непотрібні елементи. Ця функція особливо корисна для фокусування на конкретних об’єктах, усунення відволікаючих факторів або налаштування рамки зображення.

Функції зміни розміру та обрізання, які будуть реалізовані в вебзастосунку, дають користувачам змогу налаштовувати розміри та композицію своїх зображень. Незалежно від того, чи це зміна розміру зображень для різних платформ чи обрізання для виділення окремих елементів, вебзастосунок наддасть необхідні інструменти та технології для

точного та легкого виконання цих завдань. Використовуючи інтелектуальні алгоритми та інтерактивні інтерфейси користувачі зможуть ефективно маніпулювати своїми зображеннями, зберігаючи їх візуальну цілісність.

Обертання та віддзеркалення є одними з основних операцій обробки зображень, які дозволяють користувачам змінювати орієнтацію зображень і досягати певних візуальних ефектів. Ці функції відіграють важливу роль у програмах для редагування зображень, дозволяючи користувачам регулювати кут огляду, вирівнювати зображення або створювати унікальні художні композиції. У вебзастосунку будуть реалізовані вдосконалені методи та технології, щоб забезпечити надійне обертання та віддзеркалення.

Функція обертання дає змогу користувачам змінювати орієнтацію зображення. Незалежно від того, чи це невелике коригування, чи повний поворот на 90, 180 або 270 градусів, вебзастосунок буде підтримувати широкий діапазон кутів повороту, щоб задовольнити вимоги користувачів. Можливість повертати зображення є безцінною в сценаріях, коли зображення потрібно правильно вирівняти або налаштувати для певного формату відображення. Вебзастосунок буде використовувати ефективні алгоритми, які обробляють піксельні дані для точного повороту зображення, зберігаючи його візуальну якість.

Функція віддзеркалення дозволяє користувачам створювати дзеркальні ефекти, відображаючи зображення вздовж горизонтальної або вертикальної осі. Застосунок буде підтримувати як горизонтальні, так і вертикальні перевороти, надаючи користувачам творчі можливості для досягнення певних візуальних результатів. Віддзеркалення зазвичай використовується в різних контекстах, таких як створення симетричних дизайнів, коригування напрямку об’єктів на зображенні або створення унікальних композицій.

Горизонтальне віддзеркалення передбачає зміну порядку пікселів зліва направо, фактично віддзеркалюючи зображення по горизонталі. Ця функція особливо корисна під час роботи з об’єктами, які мають вибрану орієнтацію, або під час створення дзеркальних композицій.

Вертикальне віддзеркалення, з іншого боку, передбачає зміну порядку пікселів зверху вниз, фактично віддзеркалюючи зображення по вертикалі. Ця операція часто використовується для створення вертикально перевернутих композицій.

Функції обертання та віддзеркалення в вебзастосунку будуть надавати користувачам потужні інструменти для керування орієнтацією зображень і створення унікальних візуальних ефектів. Використовуючи ефективні алгоритми та математичні перетворення, буде забезпечено точні та візуально привабливі повороти та перевороти. Ці можливості розширюють творчу свободу користувачів і дозволяють їм створювати бажані композиції, будь то корекція вирівнювання, створення симетричних дизайнів або дослідження художніх трансформацій. Завдяки цим функціям вебзастосунок наддасть користувачам змогу висловлювати своє бачення та втілювати свої ідеї в життя за допомогою обробки зображень.

Візуальні налаштування відіграють важливу роль у програмах для редагування зображень, дозволяючи користувачам покращувати певні візуальні аспекти зображень для досягнення бажаних результатів. У вебзастосунку буде розроблено ряд функцій візуального налаштування, які дозволяють користувачам змінювати такі параметри, як яскравість, контраст і насиченість. Ці інструменти наддадуть користувачам більший контроль над візуальним виглядом своїх зображень і дозволяють їм точно налаштувати загальну естетику [9–12].

Функції регулювання яскравості, контрастності та насиченості дозволяють користувачам маніпулювати ключовими візуальними властивостями своїх зображень. Ці налаштування широко використовуються для покращення загального вигляду, покращення яскравості кольорів і досягнення бажаних візуальних ефектів.

Регулювання яскравості дозволяє користувачам контролювати загальну яскравість зображення. Збільшуючи яскравість, користувачі можуть зробити зображення світлішим, тоді як зменшення яскравості робить зображення

темнішим. У вебзастосунку будуть використані алгоритми, які регулюють значення пікселів зображення для досягнення бажаного рівня яскравості. Це налаштування допомагає користувачам компенсувати недоекспонування або переекспонування на фотографіях, створити певний настрій або виділити певні елементи на зображенні.

Регулювання контрастності дає змогу користувачам посилити різницю між світлими та темними ділянками зображення. Збільшуючи контрастність, користувачі можуть зробити світлі ділянки яскравішими, а тіні – темнішими, створюючи більш динамічне та візуально вражаюче зображення. І навпаки, зменшення контрасту може створити більш м’який і приглушений вигляд. У вебзастосунку будуть використані алгоритми, які регулюють тональний діапазон зображення, забезпечуючи рівномірне посилення контрастності по всьому зображенню. Ця функція особливо корисна в ситуаціях, коли зображення здаються пласкими або їм бракує глибини, оскільки допомагає виділити деталі та покращити загальний візуальний ефект.

Регулювання насиченості дозволяє користувачам керувати інтенсивністю та насиченістю кольорів зображення. Збільшення насиченості покращує яскравість і насиченість кольорів, тоді як зменшення насиченості створює більш приглушений і ненасичений вигляд. У вебзастосунку будуть використані алгоритми для налаштування інформації про колір у зображенні, забезпечуючи рівномірне застосування змін насиченості в різних колірних каналах. Ця функція є цінною, коли користувачі хочуть підкреслити кольори, створити особливу атмосферу чи викликати певні емоції.

Крім того, вебзастосунок буде підтримувати неруйнівне редагування, зберігаючи вихідні дані зображення протягом усього процесу налаштування. Це означає, що користувачі можуть повернутися до вихідного зображення або змінити налаштування в будь-який момент, забезпечуючи гнучкість і непостійні зміни. Застосунок буде використовувати структури даних і алгоритми для зберігання та керування параметрами коригування окремо від

оригінального зображення, що дозволяє користувачам експериментувати з різними параметрами без шкоди для цілісності вихідних даних [13].

Функції візуального налаштування в вебзастосунку надають користувачам потужні інструменти для покращення загального вигляду та візуального впливу їхніх зображень. Використовуючи складні алгоритми, інтерактивні інтерфейси та можливості неруйнівного редагування користувачі мають гнучкість і контроль для досягнення бажаних візуальних результатів. Ці функції дозволяють користувачам покращувати свої зображення, точно налаштовувати естетику та розкривати справжній потенціал своїх зображень.

Фільтри – це популярна та потужна функція в програмах для редагування зображень, яка дозволяє користувачам застосовувати різні візуальні трансформації до своїх зображень. Фільтри можуть кардинально змінити зовнішній вигляд зображення, додаючи художні ефекти, покращуючи кольори або створюючи унікальні стилі. У вебзастосунку буде реалізовано наступний спектр фільтрів: сепія, інвертування, градація сірого, відтінки, лінійний градієнт, радіальний градієнт і суцільний фон, що наддасть користувачам різноманітні творчі можливості.

Фільтр сепії – це класичний ефект, який надає зображенням ностальгічний, вінтажний вигляд, що нагадує старі фотографії. Після застосування фільтр сепії перетворює зображення на коричневі тони, імітуючи вигляд вицвілих або постарілих фотографій. У вебзастосунку буде використано алгоритми, які регулюють значення кольорів кожного пікселя, перетворюючи вихідні кольори на бажаний тон сепії. Цей фільтр особливо популярний для створення теплої та позачасової атмосфери на зображеннях, викликаючи почуття ностальгії чи історичного шарму.

Фільтр інверсії – це простий, але вражаючий ефект, який змінює кольори зображення. Після застосування інвертований фільтр перетворює значення кольору кожного пікселя на додатковий колір на колірному колі. Це призводить до вражаючої трансформації, коли яскраві ділянки стають

темними, і навпаки. Інвертуючий фільтр може створювати драматичні та художні ефекти, підкреслюючи контрасти та висвітлюючи деталі. У вебзастосунку буде використано алгоритми, які виконують необхідні перетворення кольорів для ефективного інвертування зображення, надаючи користувачам простий спосіб експериментувати з цим візуально захоплюючим ефектом.

Фільтр градації сірого перетворює зображення на відтінки сірого, видаляючи всю інформацію про колір. Цей фільтр створює монохроматичний ефект, який підкреслює значення яскравості зображення. Зображення у відтінках сірого можуть викликати відчуття простоти, елегантності або позачасу. У вебзастосунку буде використано алгоритми, які обчислюють значення яскравості кожного пікселя та призначають відповідний відтінок сірого. Видаляючи інформацію про колір, користувачі можуть зосередитися на текстурах, формах і тональних варіаціях зображення, посилюючи його візуальний ефект.

Фільтр відтінків дозволяє користувачам маніпулювати загальним відтінком кольору зображення, зміщуючи відтінки вздовж колірного спектра. Цей фільтр дозволяє користувачам створювати унікальні художні ефекти, змінюючи домінуючі кольори або вводячи нові палітри кольорів. У вебзастосунку буде використано алгоритми, які регулюють значення відтінку кожного пікселя, перетворюючи представлення кольору зображення. Користувачам будуть надані інтуїтивно зрозумілі елементи керування, такі як повзунки, щоб вони могли легко експериментувати з різними варіаціями відтінків і досягати бажаних колірних ефектів.

Лістинг 2.2 Застосування фільтрів до зображення:

```
const node = this.canvasRef.current;
const context = node.getContext("2d");
let brightness = "brightness(" + (50 + this.props.brightnessValue).toString() + "%) ";
```

```
let contrast = "contrast(" + (50 + this.props.contrastValue).toString() + "%) ";
let blur = "blur(" + (this.props.blurValue / 18).toString() + "px) ";
let saturate = "saturate(" + (50 + this.props.saturateValue).toString() + "%) ";
context.filter = brightness + contrast + blur + saturate + `grayscale(${this.props.grayScale}%) hue-rotate(${this.props.hueRotate}deg) invert(${this.props.invert}%) sepia(${this.props.sepia}%)`;
```

Фільтр лінійного градієнта застосовує поступовий перехід кольорів на зображенні, створюючи ефект плавного градієнта від одного кольору до іншого. Користувачі можуть вказати початковий і кінцевий кольори, а також напрямок градієнта. У вебзастосунку буде використано алгоритми, які обчислюють значення кольору для кожного пікселя на основі його положення в градієнті, забезпечуючи плавний і візуально приємний перехід. Фільтр лінійного градієнта – це універсальний інструмент, який можна використовувати для додавання глибини, створення візуально привабливого фону або створення унікальних візуальних ефектів.

Фільтр радіального градієнта створює ефект кругового або еліптичного градієнта, який виходить із центральної точки. Користувачі можуть визначати початковий і кінцевий кольори, а також розмір і форму градієнта. У вебзастосунку буде використано алгоритми, які обчислюють значення кольору для кожного пікселя на основі його відстані від центру градієнта, що забезпечує плавний і візуально вражаючий ефект радіального градієнта. Цей фільтр часто використовується, щоб підкреслити певну область, створити ефект віньєтки або створити захоплюючий фон.

Додавання тексту є однією з функцій застосунків для редагування зображень, яка дозволяє користувачам додавати текстові елементи до своїх зображень. Вебзастосунок буде забезпечувати надійну та зручну функцію додавання тексту, надаючи користувачам змогу додавати підписи, заголовки,

водяні знаки чи будь-яку іншу форму тексту для покращення своїх зображень.

Вебзастосунок буде пропонувати користувачам ряд параметрів налаштування для створення візуально привабливих і вражаючих текстових накладень. Користувачі можуть вводити спеціальний текст, вибирати шрифти, регулювати розміри, вибирати кольори та застосовувати різні параметри стилю тексту. Текст буде динамічно відображатися на зображенні, що дозволяє користувачам візуалізувати кінцевий результат перед застосуванням змін.

Лістинг 2.3 Застосування обраних текстових даних:

```
context.font = this.props.selectedStyle + " " + this.props.textSize + "px " + this.props.selectedFont;
context.fillStyle = this.props.inputColor;
context.textAlign = "center";
context.fillText(this.props.textInput, this.props.width / 2, this.props.height / 2 );
```

Позиціювання та вирівнювання накладеного тексту є ключовими аспектами додавання тексту. У вебзстосунок будуть додані інтуїтивно зрозумілі елементи керування, які дозволяють користувачам точно розміщувати текст на зображенні. Користувачі зможуть вручну перетягнути текст у потрібне місце.

Щоб забезпечити гнучкість і легкість редагування, вебзастосунок буде підтримувати неруйнівне додавання тексту. Це означає, що вихідне зображення зберігається, а текстове накладання зберігається як окремий об’єкт. Користувачі можуть будь-коли змінити або видалити текстове накладання, не впливаючи на базове зображення. Цей недеструктивний підхід дозволяє користувачам експериментувати з різними стилями тексту,

положеннями та ефектами, надаючи їм свободу вдосконалювати свої композиції, доки не досягнуть бажаного результату.

Підсумовуючи, редагування зображень є життєво важливим процесом у світі цифрових медіа. Методи та технології, які використовуються для редагування зображень, пройшли довгий шлях за ці роки, дозволяючи будь-кому легко редагувати та маніпулювати зображеннями [14].

Поєднання різних технологій і методів у редагуванні зображень революціонізувало спосіб роботи цифрових художників і фотографів. З використанням вебзастосунків редагування зображень стало ще доступнішим, дозволяючи користувачам редагувати зображення на ходу, на будь-якому пристрої.

Загалом, методи та технології, які використовуються для редагування зображень, постійно розвиваються, і можна очікувати ще більше інноваційних рішень у майбутньому. Оскільки попит на високоякісні зображення зростає, потреба в передових інструментах редагування зображень продовжуватиме зростати, що призведе до більш захоплюючих розробок у цій галузі.

2.2 Інструменти та технології для розробки вебзастосунків

Вебзастосунки – це застосунки, розроблені для роботи у веббраузерах, що робить їх доступними для користувачів з будь-якої точки світу, де є підключення до Інтернету. Розробка вебзастосунків передбачає використання комбінації мов програмування, бібліотек програмного забезпечення та засобів розробки для створення застосунку, який є надійним, ефективним і зручним для користувача. У цьому підрозділі буде розглянуто різні інструменти та технології, які будуть використанні під час розробки вебзастосунку з редагування зображень [15].

HTML (Hypertext Markup Language) – стандартна мова розмітки, яка використовується для створення вебсторінок. Це основа кожного вебсайту, яка забезпечує структуру та вміст вебсторінок. HTML дозволяє розробникам створювати заголовки, абзаци, списки, таблиці та інші типи вмісту та вказувати, як вони представлені у веббраузері.

HTML має ряд важливих функцій, які роблять його важливим інструментом для веброзробки. Однією з його ключових особливостей є його здатність підтримувати різні типи медіа, наприклад зображення та відео. HTML також підтримує гіперпосилання, що дозволяє користувачам переходити між вебсторінками та іншими онлайн-ресурсами. Крім того, HTML забезпечує підтримку форм, що дозволяє користувачам надсилати дані на вебсервер.

HTML є важливим інструментом для веброзробки та широко використовується розробниками в усьому світі. Це проста у вивченні мова, яка забезпечує потужну основу для створення сучасних інтерактивних вебсайтів. HTML є важливою частиною будь-якого інструментарію веброзробки [16].

CSS (Cascading Style Sheets) – це мова таблиць стилів, яка використовується для опису представлення документів HTML. CSS використовується для відділення вмісту вебсторінки від її представлення, дозволяючи розробникам контролювати зовнішній вигляд вебсторінок, не змінюючи їх основний вміст.

CSS працює, призначаючи стилі елементам HTML. Стилі визначають, як елементи HTML мають відображатися у веббраузері, включаючи такі речі, як розмір шрифту, колір, фон і макет. Стилі CSS можна застосовувати до окремих елементів HTML, груп елементів або всієї вебсторінки.

Однією з ключових особливостей CSS є його здатність використовувати селектори для націлювання на певні елементи HTML. Селектори дозволяють розробникам застосовувати стилі до певних частин вебсторінки, полегшуючи створення складних макетів і дизайнів сторінок.

CSS також надає широкий спектр властивостей і значень, які можна використовувати для налаштування зовнішнього вигляду вебсторінок, від розмірів і кольорів шрифту до більш розширених функцій, таких як анімація та переходи.

CSS необхідний для створення сучасних адаптивних вебдизайнів. За допомогою CSS можливо створювати зручні для мобільних пристроїв макети, які адаптуються до різних розмірів екрана, що полегшує створення вебсайтів, які чудово виглядатимуть як на комп'ютері, так і на мобільних пристроях. CSS також надає потужні інструменти для створення візуальних ефектів, таких як градієнти, тіні та анімація, що дозволяє розробникам створювати насичений, привабливий вебдосвід.

CSS є потужним інструментом для веброзробки, який широко використовується розробниками в усьому світі. CSS також є важливою частиною будь-якого інструментарію веброзробки. За допомогою CSS можливо створювати красиві адаптивні вебдизайни, які чудово виглядатимуть на будь-якому пристрої [17].

JavaScript – це мова програмування високого рівня, яка використовується для створення динамічних та інтерактивних вебсторінок. Це одна з найпопулярніших мов програмування, яка використовується сьогодні, як для зовнішньої, так і для внутрішньої веброзробки.

JavaScript працює, дозволяючи розробникам писати сценарії, які взаємодіють із кодом HTML і CSS вебсторінки. Ці сценарії можна використовувати для створення широкого спектра інтерактивних функцій, від простої перевірки форм до складних анімацій та ігор.

Однією з ключових особливостей JavaScript є його здатність маніпулювати об'єктною моделлю документа (DOM) вебсторінки. DOM – це ієрархічне представлення елементів і вмісту вебсторінки, яким можна керувати за допомогою JavaScript для створення динамічних та інтерактивних вебсторінок.

JavaScript також широко використовується для внутрішньої веброзробки, де він використовується для створення сценаріїв на стороні сервера, які виконують такі завдання, як обробка даних, автентифікація користувачів і керування базами даних. Популярні серверні фреймворки, як Node.js і Express.js, створені на основі JavaScript, що дозволяє легко створювати вебзастосунки з повним набором за допомогою однієї мови.

JavaScript – це потужна та універсальна мова, яка має широкий спектр застосувань, крім веброзробки. Він використовується для всього, від розробки додатків для настільних комп'ютерів до програмування ігор, і підтримується великою та активною спільнотою розробників [18].

Підсумовуючи, JavaScript є важливим інструментом для веброзробки, і його використовують розробники в усьому світі для створення динамічних, інтерактивних та привабливих вебсторінок. JavaScript є важливою частиною будь-якого інструментарію веброзробки.

Інтерфейсні фреймворки – це готові набори інструментів і бібліотек, які дозволяють швидко й ефективно створювати сучасні, адаптивні та інтерактивні вебзастосунки. Ці інфраструктури забезпечують стандартизовану структуру для організації коду, що полегшує створення та підтримку складних програм. Серед них такі фреймворки, як React, Angular, Vue.js, Bootstrap та Materialize.

В даній роботі буде використаний один із найпопулярніших інтерфейсних фреймворків – React, розроблений Facebook. React – це компонентний фреймворк, який дозволяє розробникам розбивати складні інтерфейси користувача на менші компоненти, які можна багаторазово використовувати.

React дозволяє створювати повторно використовувані компоненти інтерфейсу користувача, які можна легко об'єднати для створення складних інтерфейсів користувача. Він використовує віртуальну DOM (модель об'єктів документа) для ефективного відтворення змін в інтерфейсі користувача,

мінімізуючи кількість оновлень, необхідних для відображення змін у стані програми [19–22].

Одна з переваг використання React у розробці вебзастосунків полягає в тому, що він дозволяє чітко розділити проблеми між рівнем презентації (інтерфейс користувача) та бізнес-логікою (обробка даних і керування станом програми). Це полегшує підтримку та масштабування складних вебзастосунків.

React в даній роботі використовується в поєднанні з іншою зовнішньою технологією, як Redux. Redux – це контейнер передбачуваного стану для програм JavaScript, який використовується для керування станом програми, полегшуючи її налагодження та тестування. React також можна інтегрувати з серверними фреймворками, такими як Node.js і Express, які будуть описані далі [23, 24].

Використання інтерфейсних фреймворків надає широкий спектр переваг, зокрема покращену продуктивність, послідовність і якість коду. Забезпечуючи готові компоненти та структури, вони дозволяють зосередитися на розробці функцій, а не турбуватися про базові деталі інтерфейсу користувача.

Однак інтерфейсні фреймворки не позбавлені недоліків. Вони можуть додати значних накладних витрат на проєкт і можуть не підходити для всіх випадків використання. Крім того, вони можуть обмежити гнучкість і налаштування проєкту, а також можуть ускладнити інтеграцію з іншими інструментами та системами.

Серверні фреймворки – це інфраструктури розробки програмного забезпечення, які надають інструменти та бібліотеки для створення програм на стороні сервера. Ці інфраструктури дозволяють розробникам легко створювати складні вебзастосунки, надаючи стандартизовану структуру та набір інструментів для виконання таких завдань, як доступ до бази даних, автентифікація та безпека.

Серверний фреймворк обраний для розробки вебзастосунку з редагування зображень – Node.js. Node.js – це кросплатформне середовище виконання з відкритим кодом, яке дозволяє розробникам створювати серверні програми за допомогою JavaScript. Він надає керовану подіями неблокуючу модель вводу-виводу, що робить його ідеальним для створення програм реального часу, які можуть обробляти велику кількість одночасних з'єднань.

Node.js має величезну екосистему пакетів і бібліотек, доступних через його менеджер пакетів, NPM, що дозволяє легко інтегрувати різні функції у свої програми. Ці пакети містять модулі для роботи з базами даних, реалізації автентифікації та авторизації, обробки завантаження файлів і багато іншого.

Використання Node.js з інтерфейсним фреймворком React дозволяє використовувати ту саму мову та інструменти як на інтерфейсі, так і на сервері, що може призвести до більш ефективних робочих процесів розробки та кращої зручності обслуговування коду [25–27].

Серверні фреймворки надають широкий спектр переваг, зокрема покращену продуктивність, послідовність і якість коду. Забезпечуючи готові компоненти та структури, вони дозволяють зосередитися на створенні функцій, а не турбуватися про базові деталі архітектури на стороні сервера.

Однак серверні фреймворки також мають свої недоліки. Вони можуть додати значних накладних витрат на проєкт і можуть не підходити для всіх випадків використання. Крім того, вони можуть обмежити гнучкість і налаштування проєкту, а також можуть ускладнити інтеграцію з іншими інструментами та системами.

Інтегроване середовище розробки (IDE) – це програма, яка забезпечує комплексне середовище для розробки програмного забезпечення. IDE зазвичай включають редактор вихідного коду, засоби налагодження та інші функції, які полегшують написання, тестування та розгортання коду.

IDE стають все більш популярними у веброзробці, оскільки складність вебзастосунків зростає. Вони забезпечують зручний спосіб керування

багатьма файлами та компонентами, які складають вебзастосунок, і пропонують такі функції, як підсвічування коду, завершення коду та інтеграція контролю версій, які допомагають розробникам писати код ефективніше.

У веброзробці використовуються кілька популярних IDE, кожна з яких має свої сильні та слабкі сторони. Деякі з найбільш широко використовуваних IDE – Visual Studio Code, WebStorm, Atom та Sublime Text, крім того, багато веброзробників також використовують текстові редактори, такі як Notepad++ або Vim.

Для розробки вебзастосунку з редагування зображень був використано IDE Visual Studio Code. Visual Studio Code – це безкоштовний редактор коду з відкритим кодом, розроблений Microsoft для Windows, Linux і macOS. Він включає підтримку налагодження, вбудований елемент керування Git, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти та рефакторинг коду, серед інших функцій.

Visual Studio Code дуже розширюваний, а ринок пропонує тисячі розширень, які можуть покращити його функціональність. Він надає можливість установлювати розширення для популярних інтерфейсних фреймворків, таких як React і Angular, а також для серверних фреймворків, таких як Node.js і Flask. Також доступні розширення для лінінгування коду, налагодження, тестування тощо.

Однією з переваг використання Visual Studio Code є простота використання та можливість налаштувати редактор відповідно до потреб. Він також має велику спільноту розробників, які роблять внесок у його постійний розвиток, надаючи виправлення помилок і нові функції.

Підсумовуючи, процес розробки вебзастосунку передбачає використання різноманітних інструментів і технологій для створення безперебійної та ефективної взаємодії з користувачем. HTML, CSS і JavaScript формують основу будь-якого проєкту, а такий фреймворк, як React може значно прискорити процес розробки. На серверній частині буде

використано такий фреймворк, як Node.js, щоб керувати обробкою та зберіганням даних на стороні сервера. Інтегровані середовища розробки (IDE), такі як Visual Studio Code, забезпечують комплексне середовище для кодування, налагодження та тестування.

Вибір інструментів і технологій залежить від конкретних вимог проєкту та досвіду команди розробників. Використовуючи правильну комбінацію інструментів і технологій, розробники можуть створювати потужні й ефективні вебзастосунки, які відповідають потребам користувачів. Оскільки технології продовжують розвиватися, з’являтимуться нові інструменти та технології, і розробникам важливо бути в курсі останніх тенденцій і досягнень у цій галузі. Загалом розробка вебзастосунків – це складний процес, який вимагає ретельного планування, уваги до деталей і глибокого розуміння доступних інструментів і технологій.

3 РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ З РЕДАГУВАННЯ ЗОБРАЖЕНЬ

3.1 Опис архітектури вебзастосунку та його складових частин

Для розробки вебзастосунку з редагування зображень була розроблена архітектура, яка забезпечує зручну роботу користувача та можливості ефективної обробки зображень. Застосунок відповідає моделі клієнт-сервер, де клієнтська сторона розробляється за допомогою React, а серверна сторона реалізується за допомогою Node.js.

Клієнтські компоненти вебзастосунку для редагування зображень розроблено з використанням React, бібліотеки JavaScript для створення користувальницьких інтерфейсів, і вдосконалено додатковими бібліотеками, такими як Material-UI, react-dnd, prop-types і react-tooltip, щоб забезпечити бездоганний досвід редагування.

React дозволяє створювати повторно використовувані компоненти інтерфейсу користувача, що робить процес розробки ефективним і сприяє супроводжуваності коду. Компонентна архітектура React дозволяє розбивати складні елементи інтерфейсу користувача на менші керовані частини, які можна легко об’єднати для створення інтерфейсу програми. Цей підхід полегшує повторне використання коду та спрощує обслуговування, оскільки зміни, внесені до одного компонента, автоматично поширюються на всі екземпляри, де він використовується.

Компоненти інтерфейсу користувача вебзастосунку розроблено для забезпечення інтуїтивно зрозумілого та візуально привабливого середовища редагування. Наприклад, Material-UI, популярна бібліотека компонентів React, пропонує набір попередньо розроблених елементів інтерфейсу, які відповідають інструкціям Material Design. Ці компоненти, такі як кнопки, повзунки та поля введення, є естетично привабливими та забезпечують узгоджений вигляд і відчуття в усій програмі. Використовуючи компоненти

Material-UI, програма набуває сучасного та відшліфованого вигляду, покращуючи взаємодію з користувачем [28–31].

Бібліотека prop-types використовується для визначення та забезпечення очікуваних типів і властивостей вхідних даних компонентів. Вказуючи очікувані типи даних, програма може виконувати перевірку під час виконання, запобігаючи потенційним помилкам і підвищуючи надійність коду. Ця бібліотека допомагає гарантувати, що компоненти отримують правильні дані, і зменшує ймовірність неочікуваної поведінки.

Бібліотека react-dnd спеціально розроблена для реалізації функції перетягування в програмах React. Він дозволяє користувачам взаємодіяти з інтерфейсом користувача шляхом перетягування елементів.

Щоб допомогти користувачам зрозуміти різноманітні функції редагування та їхні функції, програма використовує бібліотеку підказок react-tooltip. Підказки – це невеликі інформаційні вікна, які з’являються, коли користувач наводить курсор на певний елемент або взаємодіє з ним. Надаючи підказки, додаток покращує зручність використання та дозволяє користувачам швидко дізнаватися про різні варіанти редагування без потреби у великій документації.

Серверні компоненти вебзастосунку для редагування зображень реалізовані за допомогою Node.js, який забезпечує масштабоване та ефективне середовище для виконання коду JavaScript на стороні сервера. Серверні компоненти зосереджені на обробці запитів, обробці операцій редагування зображень і спілкуванні з клієнтською стороною.

Бібліотека React-Redux поєднує React із Redux, контейнером передбачуваного стану для програм JavaScript. Підтримуючи централізоване сховище, компоненти на стороні сервера можуть керувати відповідними даними та обмінюватися ними між різними операціями редагування зображень, забезпечуючи узгодженість і уникаючи надлишкових обчислень.

Компоненти на стороні сервера можуть надсилати дії для зміни стану на стороні сервера, запускаючи необхідні операції редагування зображень і

оновлення. React-Redux надає необхідні інструменти для підключення компонентів React до сховища на стороні сервера, дозволяючи їм отримувати доступ до спільного стану та керувати ним. Ця інтеграція гарантує, що компоненти на стороні сервера та компоненти на стороні клієнта залишаються синхронізованими та можуть ефективно спілкуватися.

Вебзастосунок для редагування зображень має структуровану архітектуру, яка організовує файли та папки, щоб забезпечити зручність обслуговування, модульність і поділ проблем. Архітектура охоплює як клієнтські, так і серверні компоненти, забезпечуючи комплексну структуру для ефективних операцій редагування зображень.

Структура папок вебзастосунку має наступний вигляд:

– папка «public/» містить статичні файли, такі як HTML, зображення та інші ресурси, які безпосередньо подаються на стороні клієнта;

– папка «src/components/» містить компоненти, які використовуються всередині програми для редагування зображень;

– папка «src/components/actions/» містить файл actionTypes.js, який визначає типи дій, що використовуються в Redux для оновлення стану програми;

– папка «src/components/css/» містить файли CSS, кожен з яких відповідає за стилізацію певних компонентів або розділів у застосунку;

– папка «src/components/filters/» містить компоненти, які забезпечують роботу фільтрів;

– папка «src/components/modals/» містить файли, пов’язані зі стартовою сторінкою;

– папка «src/components/sliders/» містить компоненти, пов’язані з повзунками, які дозволяють користувачеві налаштовувати значення параметрів зображення;

– папка «src/reducers/» містить файл rootReducer.js, який об’єднує різні редюсери у застосунку.

Нижче представлена структура деяких файлів, які забезпечують роботу застосунку:

– компонент «src/components/Canvas.js» відповідає за відображення полотна, на якому відбувається редагування зображень;

– компонент «src/components/CropElement.js» представляє елемент обрізки та відображає його на полотні;

– компонент «src/components/CropSection.js» відображає розділ, який дозволяє користувачеві обрізати зображення;

– компонент «src/components/Filters.js» представляє розділ, що містить фільтри та налаштування для застосування фільтрів до зображення;

– компонент «src/components/Footer.js» відображає нижню частину сторінки, на якій відображається поточний розмір зображення та повзунок для редагування масштабу;

– компонент «src/components/Header.js» представляє верхню частину сторінки, на якій відображають кнопки для завантаження нового зображення та збереження відредагованого зображення;

– компонент «src/components/Navbar.js» представляє панель навігації, яка містить кнопки для переходу між різними функціональними частинами програми;

– компонент «src/components/ResizeSection.js» представляє розділ, який дозволяє користувачеві змінювати розмір зображення;

– компонент «src/components/RotateSection.js» представляє розділ, що дозволяє користувачеві повертати зображення;

– компонент «src/components/TextInput.js» представляє поле введення тексту, яке може бути використане для додавання тексту до зображення;

– файл «src/App.js» представляє основний компонент програми, який поєднує різні компоненти та визначає основну структуру та маршрутизацію програми;

– файл «src/index.js» є точкою входу для програми та використовується для рендерингу головного компонента в DOM;

– файл «package.json» містить список залежностей і конфігурацій для проєкту.

Згадана вище структура папок і файлів забезпечує логічну організацію кодової бази вебзастосунку. Він розділяє компоненти, сторінки, файли, пов’язані з Redux, маршрути API та службові модулі в окремі каталоги, забезпечуючи модульну кодову базу, яку можна підтримувати.

Дотримуючись цієї структури, стає легше переміщатися та знаходити певні файли та папки, сприяючи повторному використанню коду та простоті обслуговування. Крім того, ця організація забезпечує ефективну співпрацю між членами команди, оскільки кожен компонент або функціональність займає своє місце в структурі проєкту.

3.2 Розроблення та інтеграція модулів редагування зображень

Модулі редагування зображень є важливими компонентами вебзастосунку, які дозволяють користувачам виконувати різні операції редагування своїх зображень. Нижче буде описано ключові модулі редагування зображень, які були розроблені.

В першу чергу було створено редуктор, який відповідає за обробку дій і відповідне оновлення стану програми. Початковий стан визначається як об’єкт з різними властивостями, що представляють різні аспекти програми.

Функція rootReducer приймає поточний стан і дію як параметри та повертає новий стан на основі типу дії. Функція використовує серію операторів if і else if для обробки різних дій і відповідного оновлення стану. Кожен оператор if представляє певний тип дії та повертає новий об’єкт стану з оновленими властивостями.

Наприкінці функція rootReducer експортується як стандартний експорт модуля, що робить його доступним для використання в інших частинах програми.

Далі потрібно визначити генератори дій – функції, які створюють і повертають дії, які є простими об’єктами JavaScript, що описують зміну, яку слід внести в стан програми. Кожна з цих функцій дотримується однієї моделі: вони повертають об’єкт дії з типом і корисним навантаженням. Тип – це рядок, який описує дію, а корисне навантаження містить дані, пов’язані з дією.

Ці генератори дій можуть бути відправлені, щоб ініціювати зміни в сховищі Redux. Відправлені дії потім обробляються редукторами, які визначають, як має оновлюватися стан програми у відповідь на дії.

Наступним кроком є розроблення можливості завантаження зображення. Для цього на стартовій сторінці є панель, куди зображення або скидається або при натисканні на неї обирається необхідний файл. Після завантаження зображення відображається на компоненті Canvas, готове до редагування. Далі буде описано реалізацію функції завантаження зображення.

У коді використовується компонент DropTarget із бібліотеки react-dnd, щоб створити ціль вилучення файлів. Він визначає тип файлу як NativeTypes.FILE і визначає цільову поведінку відкидання в об’єкті nativeFileTarget.

Коли файл скидається на ціль скидання або коли користувач вручну вибирає файл за допомогою введення файлу, викликається функція onImageChange. Він зчитує файл за допомогою API FileReader, перетворює вміст файлу на URL-адресу даних за допомогою readAsDataURL і надсилає дію із даними файлу як корисним навантаженням. Ця дія виконується сховищем Redux, яке оновлює властивість зображення за допомогою даних завантаженого файлу.

Наступним було розроблено компонент Canvas, який служить центральним елементом, де відбуваються дії з редагування зображення. Він діє як контейнер для інших модулів редагування та керує відображенням і маніпулюванням зображенням. Компонент Canvas відповідає за візуалізацію

зображення, отримання даних від користувача та застосування запитаних змін.

У конструкторі компонент встановлює посилання на елемент canvas за допомогою React.createRef() і призначає його властивості canvasRef.

Функція drawCanvas призначена для малювання полотна на основі наданого зображення та попередніх атрибутів. Він налаштовує контекст полотна, застосовує трансформації (перевертання, обертання, фільтри), малює зображення та обробляє рендеринг тексту.

Метод життєвого циклу componentDidUpdate реалізовано для виклику drawCanvas під час оновлення компонента. Він завантажує зображення та викликає drawCanvas, якщо певні властивості, пов’язані з розділами інтерфейсу (обрізання, обертання, текстове поле, фільтр), змінилися.

Лістинг 3.1 Метод життєвого циклу componentDidUpdate:

```
componentDidUpdate(prevProps) {
if (
prevProps.showCropCanvas !== this.props.showCropCanvas ||
prevProps.showRotateSection !== this.props.showRotateSection ||
prevProps.showSlider !== this.props.showSlider ||
prevProps.showTextField !== this.props.showTextField ||
prevProps.showFilter !== this.props.showFilter
) {
} else {
var img = new Image();
img.src = this.props.image;
img.onload = () => this.drawCanvas(img, prevProps);
}
}
```

Під кінець елемент canvas візуалізується за допомогою canvasRef і реквізитів для ширини, висоти, масштабування та обробників подій миші.

Загалом компонент Canvas обробляє візуалізацію та маніпуляції з елементом canvas, включаючи такі функції, як обрізання, обертання, фільтрування та рендеринг тексту.

Компонент Crop дозволяє користувачам вибрати певну область зображення та обрізати її відповідно. Щоб інтегрувати компонент Crop, його потрібно підключити до компонента Canvas і отримати необхідні дані зображення. Цього можна досягти, передавши дані зображення та відповідні параметри до компонента Crop як props. Props, скорочення від «properties», є фундаментальною концепцією в React, яка дозволяє передавати дані від батьківського компонента до його дочірнього компонента. Props — це спосіб спілкування та обміну даними між компонентами, що дозволяє їм взаємодіяти та обмінюватися інформацією.

Компонент Crop забезпечує інтерактивні елементи керування для вибору області обрізання – обмежувальну рамку зі змінним розміром. Вибрану область можна регулювати шляхом переміщення ручок, забезпечуючи гнучкість і точність кадрування. Коли користувач завершить вибір кадрування, компонент Crop має передати оновлені дані зображення компонента Canvas для відображення та подальшого редагування. Далі буде описано реалізацію функції обрізання зображення.

У коді використовується компонент DragSource з бібліотеки react-dnd, щоб зробити елемент обрізання перетягуваним. Він визначає поведінку джерела перетягування в об’єкті cardSource, включно з можливістю перетягування лише тоді, коли атрибут «flag» має значення false. Функція beginDrag викликається, коли починається перетягування, і повертає дані елемента, які потрібно перетягнути.

Компонент Crop обробляє різні події миші, щоб уможливити зміну розміру та оновлення області обрізання. Функція handleMouseDown запускається, коли користувач натискає кнопку миші всередині елемента

обрізання та визначає область елемента, де було клацнуто мишею (наприклад, верхній лівий, верхній правий, нижній правий, нижній лівий кути). Він також зберігає початкові координати клацання миші.

Лістинг 3.2 Функція handleMouseDown:

```
handleMouseDown = e => {
let mouseXInCropElement = e.nativeEvent.offsetX;
let mouseYInCropElement = e.nativeEvent.offsetY;
if (mouseXInCropElement < 10 && mouseYInCropElement < 10) {
console.log("LT");
this.props.setCropDivClickedResizeRegion("LT");
this.props.setCropDivInitialCoor(
mouseXInCropElement,
mouseYInCropElement
 );
} else if (
this.props.cropDivWidth - mouseXInCropElement < 10 &&
mouseYInCropElement < 10
) {
console.log("RT");
this.props.setCropDivClickedResizeRegion("RT");
this.props.setCropDivInitialCoor(
mouseXInCropElement,
mouseYInCropElement
);
} else if (
this.props.cropDivWidth - mouseXInCropElement < 10 &&
this.props.cropDivHeight - mouseYInCropElement < 10
) {
console.log("RB");
```

```
this.props.setCropDivClickedResizeRegion("RB");
this.props.setCropDivInitialCoor(
mouseXInCropElement,
mouseYInCropElement
);
} else if (
mouseXInCropElement < 10 &&
this.props.cropDivHeight - mouseYInCropElement < 10
) {
console.log("LB");
this.props.setCropDivClickedResizeRegion("LB");
this.props.setCropDivInitialCoor(
mouseXInCropElement,
mouseYInCropElement
);
}
};
```

Функція handleMouseUp запускається, коли користувач відпускає кнопку миші. Він обчислює різницю в координатах між початковим клацанням і позицією відпускання, щоб визначити нову ширину та висоту елемента обрізання. Конкретна логіка зміни розміру залежить від клацаної області зміни розміру (наприклад, «RB» для нижнього правого кута, «RT» для верхнього правого кута тощо). Він надсилає дії для оновлення розміру та положення елемента обрізання в сховищі Redux.

Лістинг 3.3 Функція handleMouseUp:

```
handleMouseUp = e => {
let mouseXInCropElement = e.nativeEvent.offsetX;
let mouseYInCropElement = e.nativeEvent.offsetY;
```

```
let diffX = mouseXInCropElement - this.props.cropDivClickInitialX;
let diffY = mouseYInCropElement - this.props.cropDivClickInitialY;
if (this.props.cropDivClickedResizeRegion === "RB") {
this.props.setCropDivSize("RB", diffX, diffY);
} else if (this.props.cropDivClickedResizeRegion === "RT") {
this.props.setCropDivLeftAndTop(diffY, 0);
this.props.setCropDivSize("RT", diffX, -diffY);
} else if (this.props.cropDivClickedResizeRegion === "LT") {
this.props.setCropDivLeftAndTop(diffY, diffX);
this.props.setCropDivSize("LT", -diffX, -diffY);
} else if (this.props.cropDivClickedResizeRegion === "LB") {
this.props.setCropDivLeftAndTop(0, diffX);
this.props.setCropDivSize("LT", -diffX, diffY);
 }
 };
```

Функція handleMouseMove запускається, коли миша переміщується в межах елемента обрізання. Він визначає поточний стиль курсора на основі положення миші відносно меж елемента обрізання. Стиль курсора оновлюється в стані компонента.

Лістинг 3.4 Функція handleMouseMove:

```
handleMouseMove = e => {
let mouseXInCropElement = e.nativeEvent.offsetX;
let mouseYInCropElement = e.nativeEvent.offsetY;
if (mouseXInCropElement < 10 && mouseYInCropElement < 10) {
this.setState({ cursor: "se-resize" });
} else if (
this.props.cropDivWidth - mouseXInCropElement < 10 &&
mouseYInCropElement < 10
```

```
  ) {
this.setState({ cursor: "ne-resize" });
} else if (
this.props.cropDivWidth - mouseXInCropElement < 10 &&
this.props.cropDivHeight - mouseYInCropElement < 10
) {
this.setState({ cursor: "nw-resize" });
} else if (
mouseXInCropElement < 10 &&
this.props.cropDivHeight - mouseYInCropElement < 10
) {
this.setState({ cursor: "sw-resize" });
} else {
this.setState({ cursor: null });
}
};
```

Компонент Crop рендерить елемент обрізання, який можна перетягувати та змінювати розмір. Він обгортає компонент функцією connectDragSource, щоб увімкнути функцію перетягування. Коли компонент перетягується (isDragging має значення true), відображається прозорий , щоб зберегти поведінку перетягування. В іншому випадку обрізаний елемент візуалізується з указаною шириною, висотою, стилем курсора та положенням.

Загалом, компонент Crop надає можливість перетягувати та змінювати розмір елемента обрізання, обробляє події миші для визначення області зміни розміру, яка була клацнута, і обчислює новий розмір, а також оновлює сховище Redux розміром і положенням елемента обрізання.

Компонент Filters пропонує низку фільтрів зображень і налаштувань для покращення або зміни зовнішнього вигляду зображення. Він включає

такі функції, як яскравість, контраст, насиченість, розмиття, сепія, градація сірого, зміна відтінку, градієнт. Користувачі можуть регулювати інтенсивність або параметри цих фільтрів за допомогою повзунків або полів введення.

Щоб інтегрувати компонент Filters, його слід підключити до компонента Canvas і отримувати дані зображення як props. Компонент Filters надає інтуїтивно зрозумілий інтерфейс користувача для вибору та налаштування бажаних фільтрів.

Коли користувач змінює налаштування фільтра, компонент Filters застосовує зміни до даних зображення та передає оновлене зображення компонента Canvas для відображення. Компонент Canvas оновлює відтворене зображення в режимі реального часу, щоб відображати застосовані фільтри.

Такі функції як яскравість, контраст, насиченість, розмиття, сепія, градації сірого, зміна відтінку та інвертування розроблені за однією схемою: є контейнер усередині якого знаходиться компонент Typography і компонент Slider. Компонент Typography відображає назву функції та її поточне значення, а компонент Slider це повзунок який надає можливість змінювати значення функції. Наприкінці кожна функція експортується за допомогою функції підключення з react-redux. Ця функція підключає компонент до сховища Redux, відображаючи стан. Далі в компоненті Canvas отриманні значення застосовуються до зображення.

Також були реалізовані функції лінійного градієнта, кругового градієнта та залиття фону. Для цього потрібно обрати тип функції, колір або кольори та положення для градієнта. Обрані значення зберігаються та експортуються за допомогою функції підключення з react-redux. Далі на компоненті Canvas отримані значення присвоюються до context.fillStyle.

Компонент Resize дозволяє користувачам регулювати розмір зображення. Він надає параметри для зміни розмірів зображення, вказавши настроювані значення ширини та висоти або використовуючи попередньо визначені пропорції.

Щоб інтегрувати компонент Resize, його слід підключити до компонента Canvas і отримувати дані зображення як props. Компонент Resize надає поля введення за допомогою яких користувачі можуть вводити потрібні значення ширини та висоти.

Коли користувач змінює розміри або вибирає співвідношення сторін, компонент Resize застосовує зміни до даних зображення та передає оновлене зображення компонента Canvas для відображення. Компонент Canvas оновлює відтворене зображення в режимі реального часу, щоб відобразити змінені розміри. Далі буде описано реалізацію компоненту Resize.

Користувач взаємодіє з інтерфейсом користувача, вводячи нові значення ширини та висоти в поля введення, надані компонентом.

Коли користувач вводить текст у поля введення, запускаються обробники подій handleWidthChange і handleHeightChange. Ці функції оновлюють стан компонента за допомогою нових значень ширини та висоти, введених користувачем.

Коли користувач введе бажані значення ширини та висоти, він може натиснути кнопку «Застосувати». Обробник події onClick, пов’язаний із кнопкою, запускає функцію submitResizeValues, яка надсилає дію до сховища Redux.

Відправлену дію отримує редуктор, відповідальний за обробку типу. Редуктор оновлює відповідні властивості в стані сховища Redux, такі як resizedWidth і resizedHeight.

Після оновлення сховища Redux зміни в стані запускають повторне відтворення компонентів, які покладаються на оновлені властивості.

Компонент Rotate дозволяє користувачам повертати зображення на потрібний кут. Він забезпечує інтуїтивно зрозумілі елементи керування обертанням, такі як кнопки та повзунок, що дозволяє користувачам повертати зображення за або проти годинникової стрілки. Оновлене значення повороту миттєво застосовується до зображення, забезпечуючи зворотний

зв’язок у реальному часі. Щоб інтегрувати компонент Rotate, його потрібно підключити до компонента Canvas і отримати дані зображення як props.

Коли користувач змінює кут повороту, компонент Rotate застосовує обертання до даних зображення та передає оновлене зображення компонента Canvas для відображення. Компонент Canvas оновлює відтворене зображення в режимі реального часу, щоб відобразити зміни обертання.

Компонент Flip дозволяє користувачам віддзеркалювати зображення горизонтально або вертикально. Він надає користувачам кнопки для легкого виконання цих операцій гортання.

Щоб інтегрувати компонент Flip, його слід підключити до компонента Canvas і отримувати дані зображення як props. Компонент Flip надає інтуїтивно зрозумілі елементи керування: кнопки з позначками «Віддзеркалити горизонтально» і «Віддзеркалити вертикально», що запускають відповідні операції гортання.

Коли користувач натискає кнопки повороту, компонент Flip застосовує відповідне перетворення повороту до даних зображення та передає оновлене зображення компонента Canvas для відображення. Компонент Canvas оновлює візуалізоване зображення в режимі реального часу, щоб відображати перевернуту орієнтацію.

Далі надано описання коду, який реалізує функціональні можливості для обертання та віддзеркалення зображення.

Код містить дві кнопки для горизонтального та вертикального гортання зображення. Коли користувач натискає будь-яку кнопку, запускається відповідний обробник подій toggleHorizontalFlip або toggleVerticalFlip. Ці обробники подій надсилають дії до сховища Redux разом із поточним станом опції гортання (horizontalFlip або verticalFlip). Редуктор, пов’язаний із цими діями, оновлює відповідну властивість відображення в сховищі Redux, що запускає повторне відтворення зображення із застосованим ефектом відображення.

Існує дві кнопки для повороту зображення на 90 градусів вліво або вправо. Коли користувач натискає будь-яку кнопку, запускається відповідний обробник події, rotate90DegreeLeft або rotate90DegreeRight. Ці обробники подій надсилають дії до сховища Redux, а також кут повороту як дані (-90 для лівого та 90 для правого). Редуктор, пов’язаний із цими діями, оновлює властивість обертання в сховищі Redux, і зображення повторно відтворюється із застосуванням обертання.

Код містить компонент Slider, який дозволяє користувачам точно налаштовувати кут повороту. Коли користувач перетягує повзунок, запускається обробник події onChange, handleFineTuneRotate. Цей обробник подій надсилає дію до сховища Redux із поточним значенням повзунка як корисним навантаженням. Редуктор оновлює значення обертання в сховищі Redux, і зображення повторно відтворюється з точно налаштованим обертанням.

Компонент Text дозволяє користувачам додавати текстові накладення на зображення. Це дозволяє користувачам вводити спеціальний текст, вибирати стилі шрифту, регулювати розмір і розміщувати текст на зображенні. Щоб інтегрувати компонент Text, його слід підключити до компонента Canvas і отримувати дані зображення як props. Компонент Text надає інтерфейс для введення тексту, вибору стилів шрифту, налаштування розміру та розміщення тексту на зображенні.

Коли користувач додає або змінює текстове накладання, компонент Text застосовує зміни до даних зображення та передає оновлене зображення компонента Canvas для відображення. Компонент Canvas оновлює відтворене зображення в режимі реального часу, щоб відображати текстове накладання. Далі буде описано реалізацію функції додавання тексту до зображення.

Існує поле введення тексту, де користувачі можуть вводити потрібний текст. Вхідне значення контролюється властивістю стану Redux textInput. Коли користувач вводить або змінює текст, запускається обробник події

handleTextChange. Він надсилає дію до сховища Redux, оновлюючи властивість textInput новим текстовим значенням.

Існує компонент повзунка, який дозволяє користувачам регулювати розмір тексту. Початкове значення повзунка контролюється властивістю стану Redux textSize. Коли користувач переміщує повзунок, запускається обробник події handleTextSizeChange, який відправляє дію для оновлення властивості textSize у сховищі Redux.

Існує компонент вибору кольору (HuePicker), який дозволяє користувачам вибирати колір для тексту. Коли користувач вибирає колір, запускається обробник події handleChangeComplete, який відправляє дію для оновлення значення кольору в сховищі Redux.

Існує спадний компонент, де користувачі можуть вибрати шрифт для тексту. Вибраний шрифт контролюється властивістю selectedFont стану Redux. Коли користувач вибирає шрифт зі спадного меню, запускається обробник події handleFontChange, який надсилає дію, щоб оновити властивість selectedFont у сховищі Redux.

Також існує інший спадний компонент, де користувачі можуть вибрати стиль (звичайний, курсив або жирний) для тексту. Вибраний стиль контролюється властивістю стану Redux selectedStyle. Коли користувач вибирає стиль зі спадного списку, запускається обробник події handleStyleChange, який надсилає дію, щоб оновити властивість selectedStyle у сховищі Redux.

Протягом усього процесу розробки використовуються різноманітні бібліотеки та залежності, щоб оптимізувати впровадження та покращити взаємодію з користувачем. Бібліотека React-DND (Drag and Drop) використовується для ввімкнення функції перетягування для таких компонентів, як «Обрізання» та «Текст», що дозволяє користувачам інтерактивно переміщувати та розміщувати елементи на полотні. React-Redux використовується для керування станом програми та полегшення зв’язку між компонентами.

Використовуючи ці компоненти та бібліотеки, застосунок з редагування зображень пропонує комплексний і зручний досвід. Користувачі можуть обрізати зображення, застосовувати різноманітні фільтри та коригування, обертати зображення, додавати текстові накладення та безперешкодно завантажувати нові зображення. Модуль забезпечує оновлення в реальному часі та інтерактивний інтерфейс, що дозволяє користувачам візуалізувати зміни, які вони вносять у зображення.

3.3 Тестування роботи вебзастосунку

Після розробки вебзастосунку дуже важливо ретельно протестувати його роботу, щоб переконатися, що всі функції працюють належним чином. Далі буде описано процес тестування та огляд різних функцій та їхньої очікуваної поведінки. Супровідні рисунки надають візуальне представлення інтерфейсу програми та демонструють правильне функціонування кожної функції.

Щоб почати використовувати вебзастосунок, користувачі переходять на стартову сторінку (рис. 3.1), на якій помітно відображається панель для завантаження зображень.

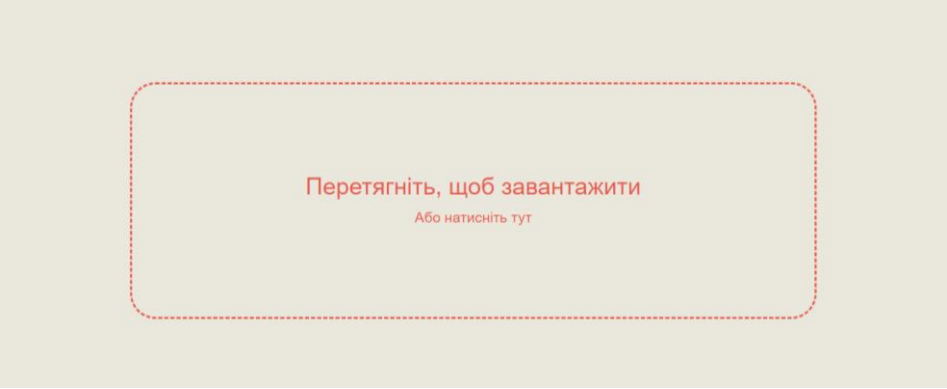

Рисунок 3.1 – Стартова сторінка вебзастосунку

Щоб завантажити своє зображення та почати роботу над його редагуванням слід слідувати наданим інструкціям або просто клацнути на панелі, щоб відкрити вікно, де вони можуть вибрати потрібний файл зображення. Крім того, вони можуть перетягнути файл на панель, як показано на рисунку 3.2.

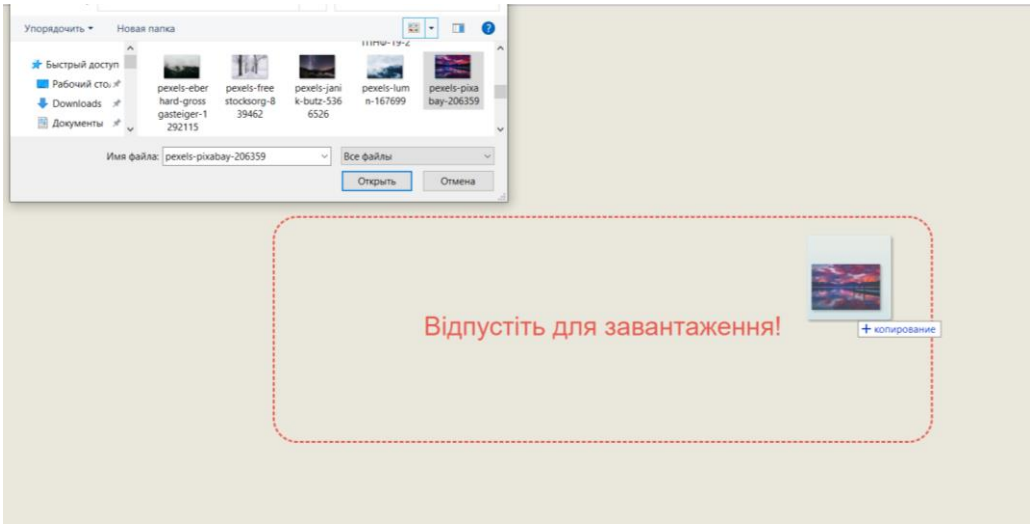

Рисунок 3.2 – Приклад перетягування зображення на панель

Після завантаження зображення та початку процесу редагування користувачі переходять на головну сторінку. Тут відображається робоча зона та навігаційне меню, що містить різні елементи візуального коригування (рис. 3.3). Ці елементи дозволяють користувачам змінювати яскравість, контраст, насиченість і розмиття за допомогою повзунків. Приклад функції візуального коригування в дії показано на рисунку 3.4.

Ще одна тестована функція – зміна розміру зображення. Поточний розмір зображення відображається в нижньому лівому куті інтерфейсу, і зображення відповідно масштабується на полотні. На рисунку 3.5 наведено приклад масштабу зображення до сорока відсотків, висота якого дорівнює тисячі чотириста шістдесяти семи та ширині – дві тисячі двісті один піксель. Тестування цієї функції передбачає введення нових значень для ширини та висоти та спостереження за відповідними змінами в інтерфейсі. Зображення

повинно візуально виглядати меншим, зберігаючи той самий масштаб, що вказує на правильну функціональність (рис. 3.6).

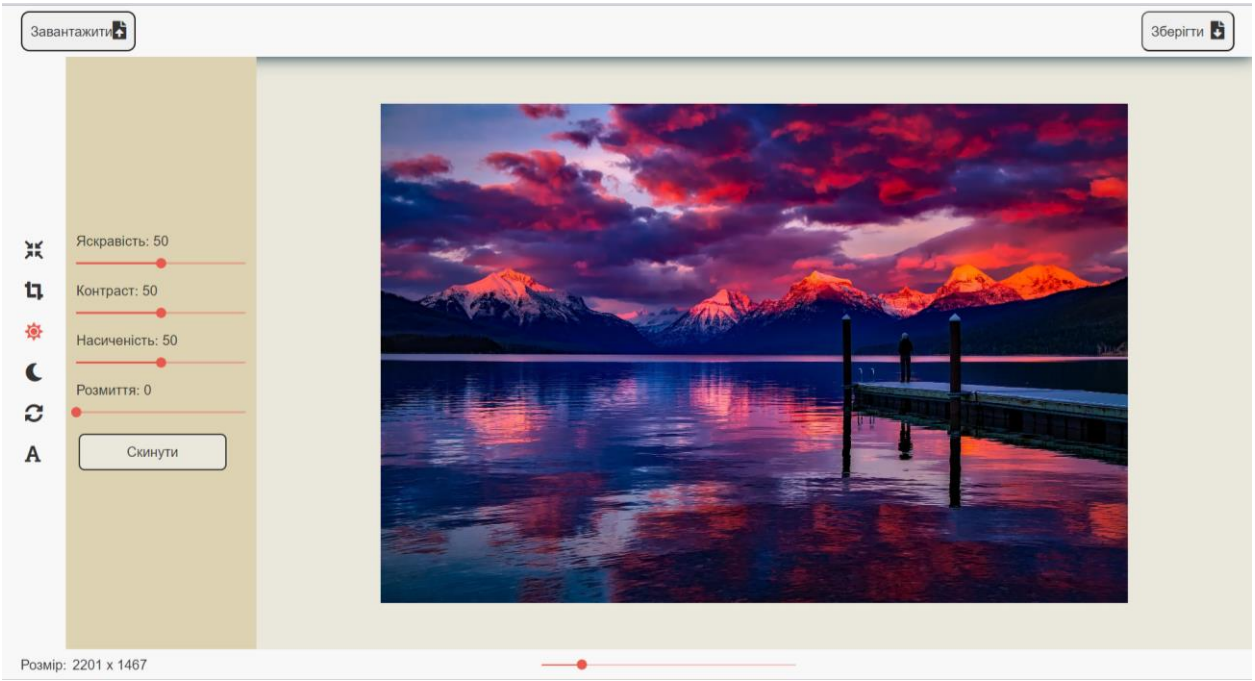

Рисунок 3.3 – Головна сторінка з відкритими елементами візуального коригування

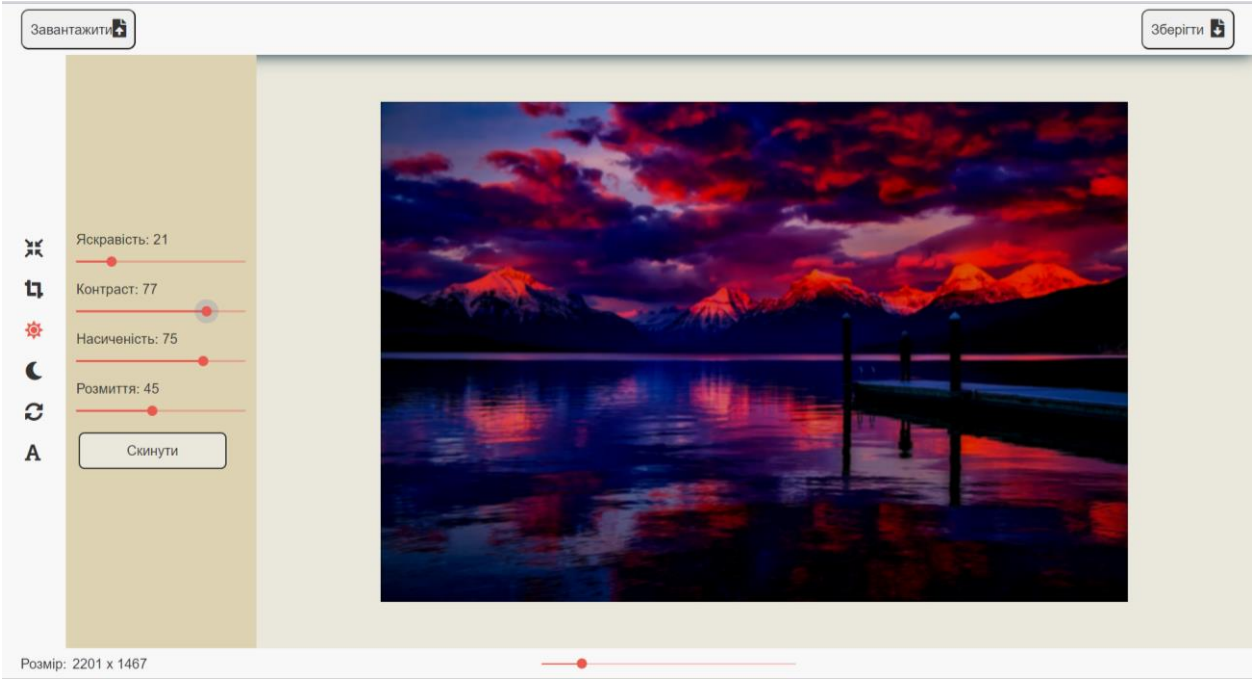

Рисунок 3.4 – Приклад використання елементів візуального коригування

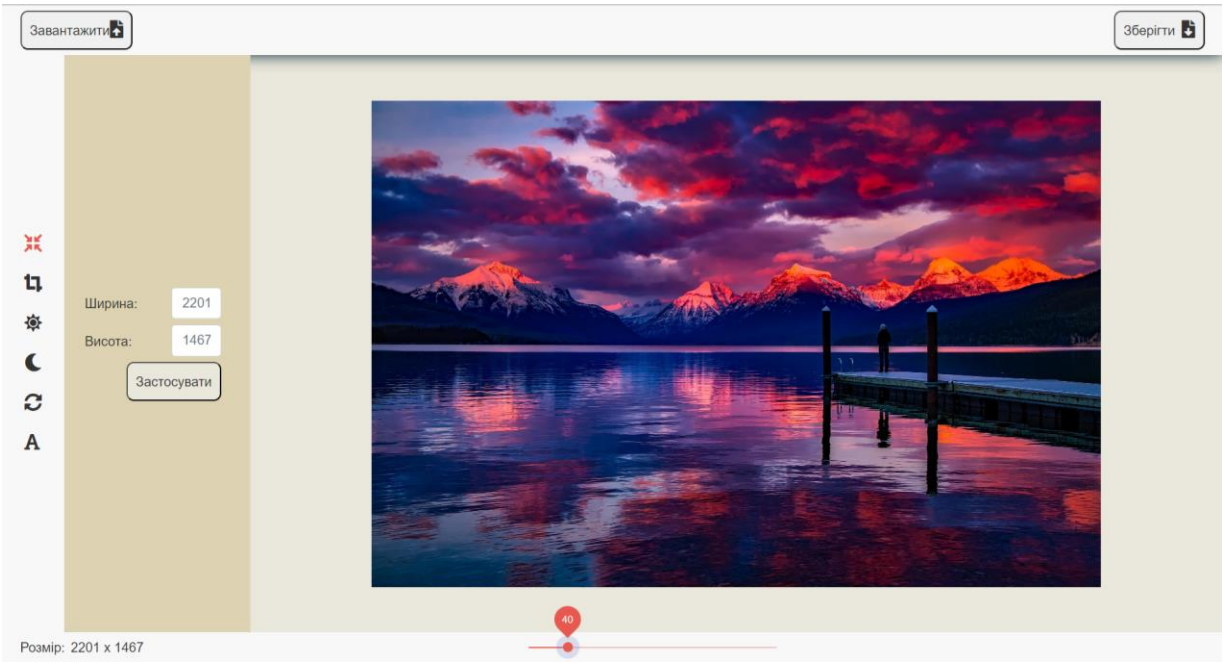

Рисунок 3.5 – Вигляд параметрів зображення та його масштаб

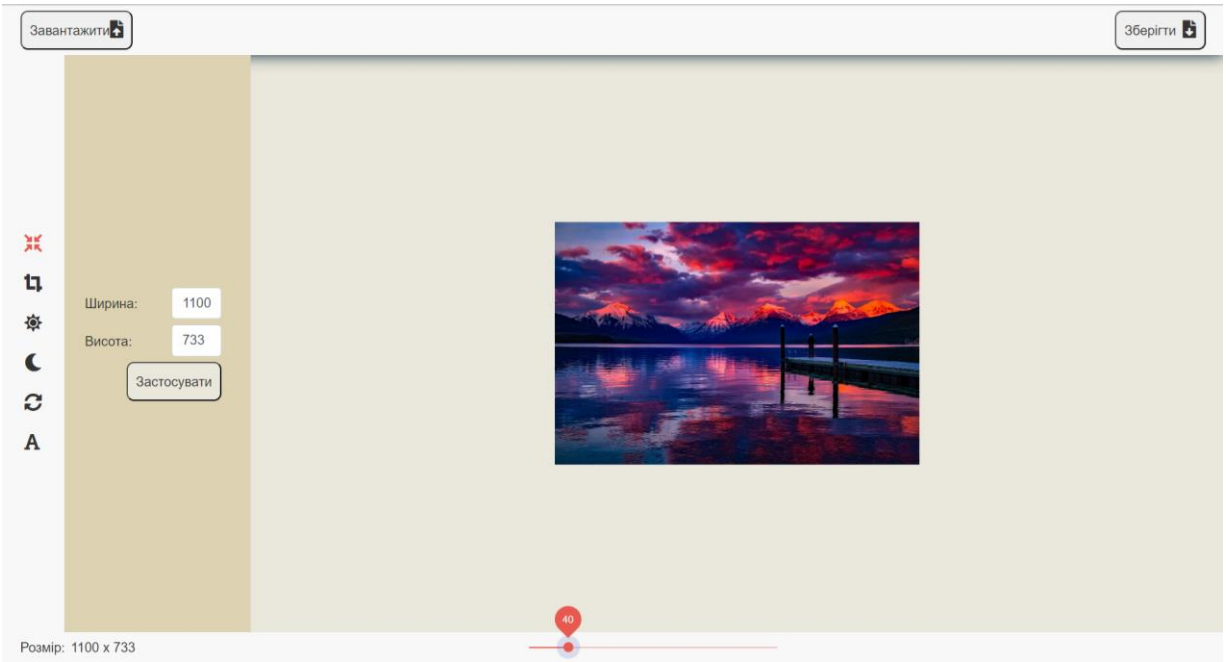

Рисунок 3.6 – Вигляд зображення, його параметрів та масштабу після змін

Вебзастосунок також включає можливості горизонтального та вертикального віддзеркалення, як показано на рисунку 3.7. Користувачі можуть застосовувати ефекти дзеркального відображення до зображення, і інтерфейс має точно відображати ці зміни.

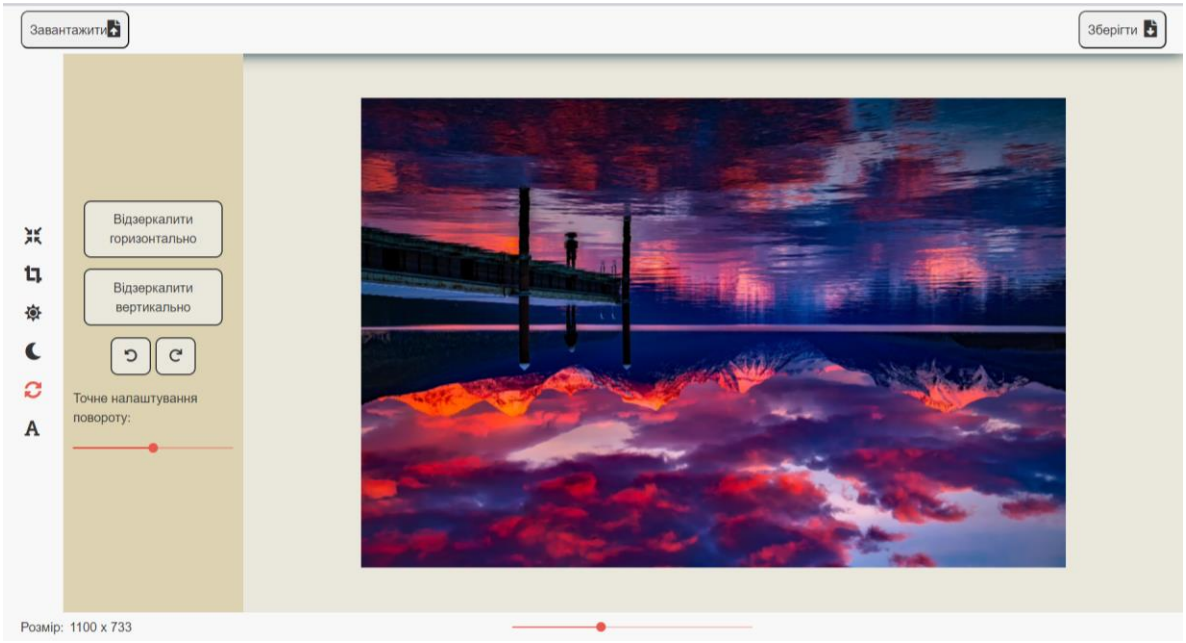

Рисунок 3.7 – Відзеркалення зображення по горизонталі та вертикалі

Також, застосунок дозволяє користувачам додавати текст до зображення, надаючи параметри для налаштування його розміру, кольору, шрифту та стилю (рис. 3.8). Тестування цієї функції передбачає перевірку того, що текст можна успішно додати, а його властивості можна змінити відповідно до вподобань користувача.

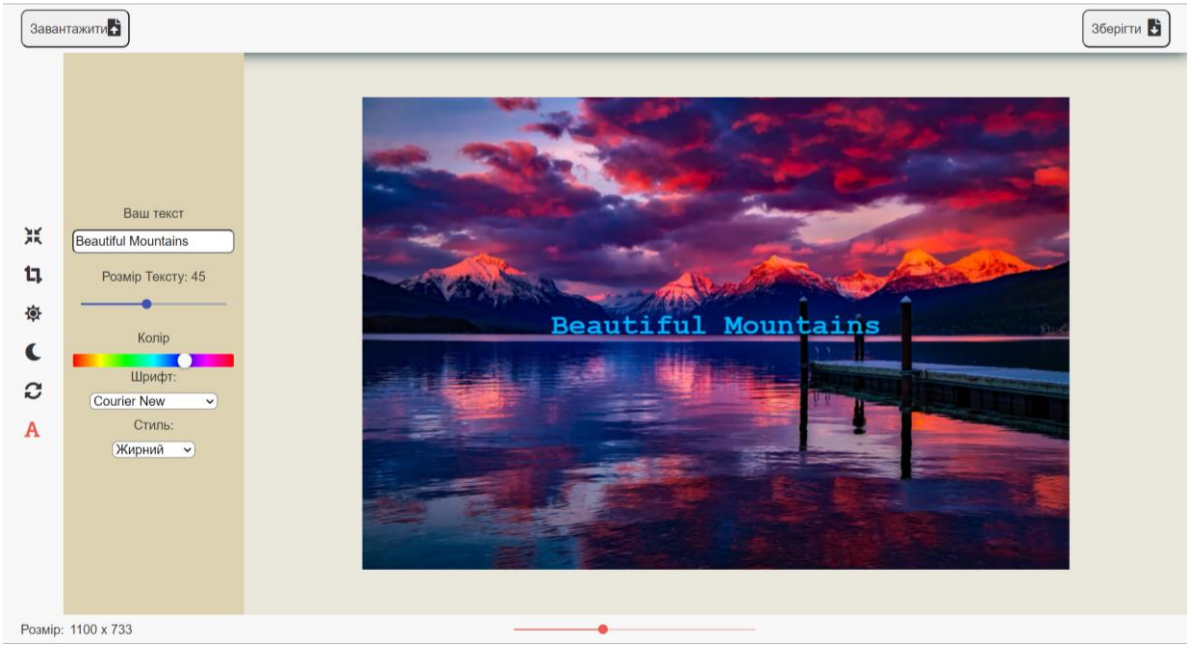

Рисунок 3.8 – Демонстрація додавання тексту

Користувачам доступно кілька фільтрів для покращення своїх зображень. На рисунках 3.9–3.12 показано роботу фільтрів відтінку, інверсії, градації сірого та сепії відповідно. Кожен фільтр слід застосовувати правильно, відповідно змінюючи вигляд зображення.

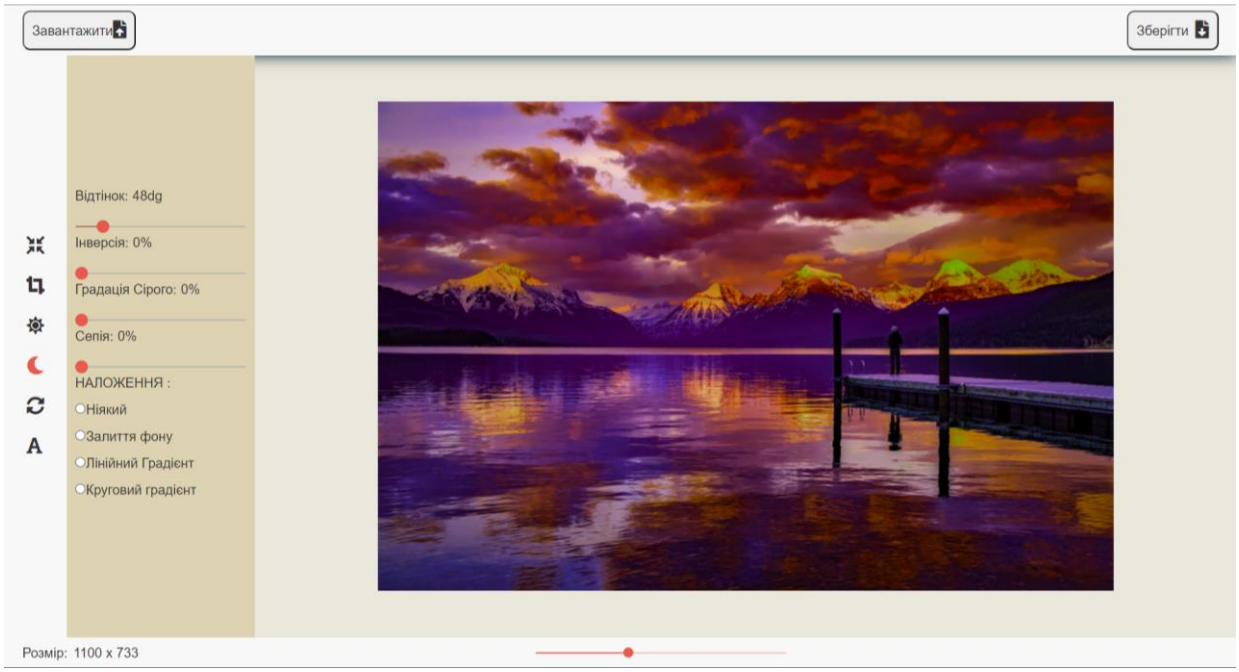

Рисунок 3.9 – Демонстрація застосування фільтру зміна відтінку

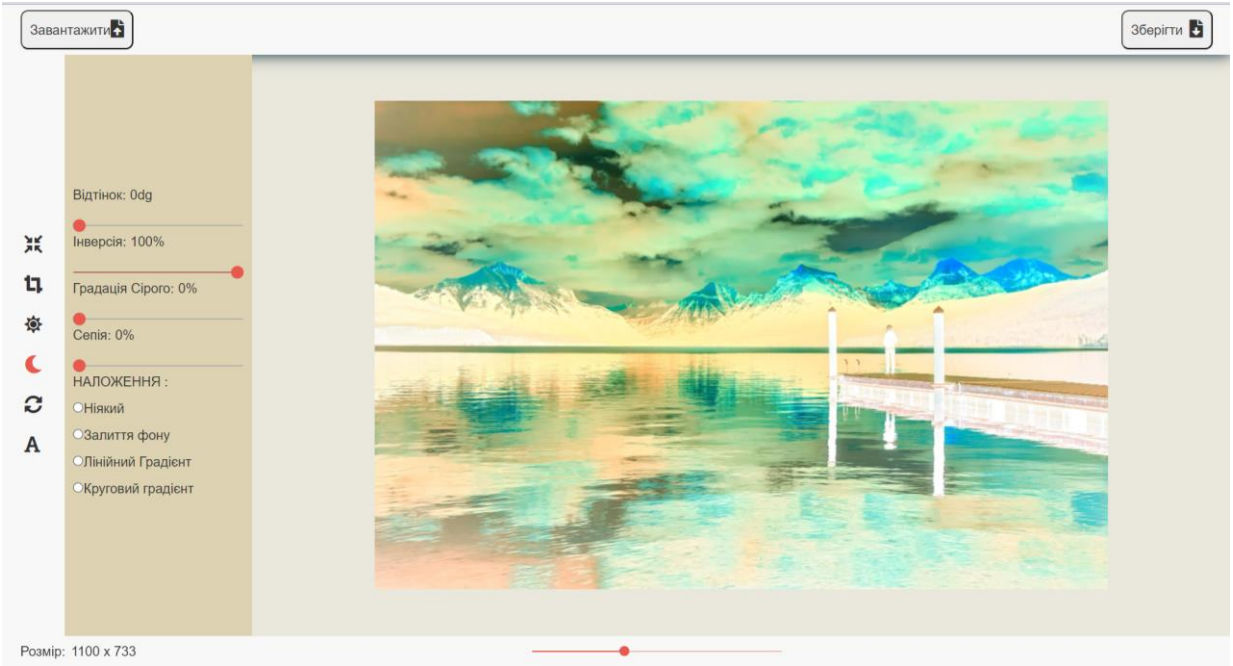

Рисунок 3.10 – Демонстрація застосування фільтру інверсія

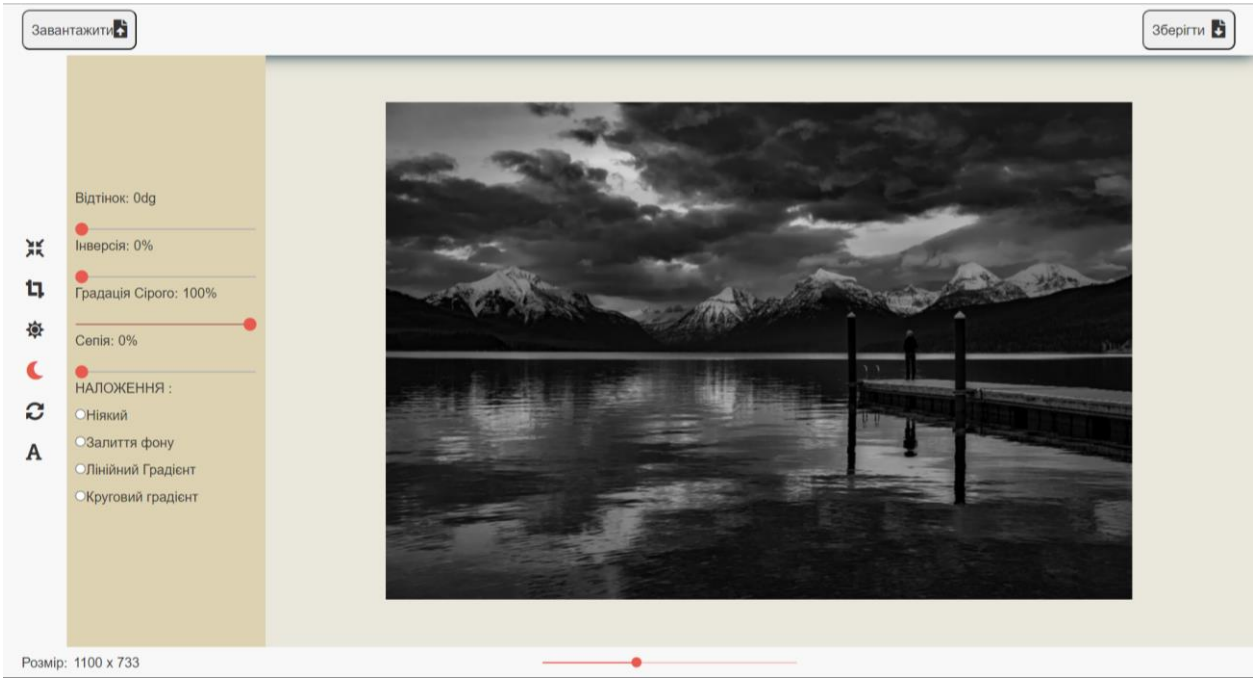

Рисунок 3.11 – Демонстрація застосування фільтру градація сірого

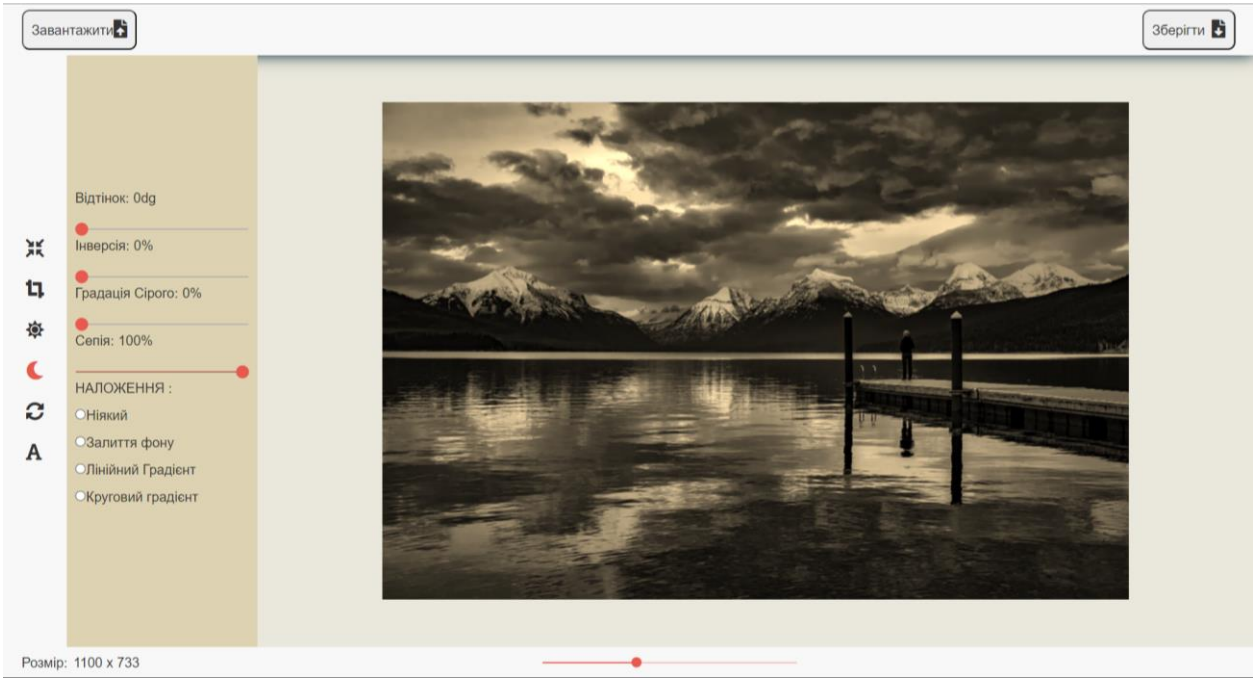

Рисунок 3.12 – Демонстрація застосування фільтру сепія

Крім того, застосунок пропонує функції заливання фону, лінійного градієнта та кругового градієнта, які показані на рисунках 3.13–3.15

відповідно. Ці функції мають точно заповнювати фон зображення за допомогою вибраного кольору або параметрів градієнта.

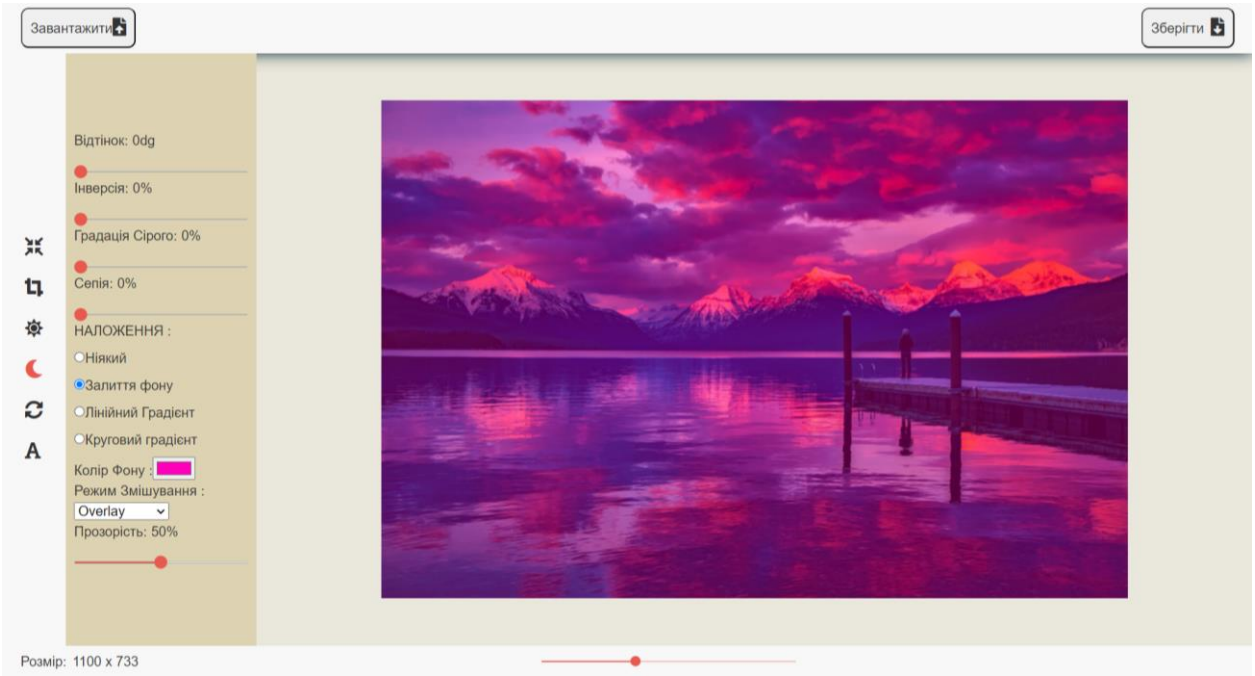

Рисунок 3.13 – Демонстрація застосування фільтру заливання фону

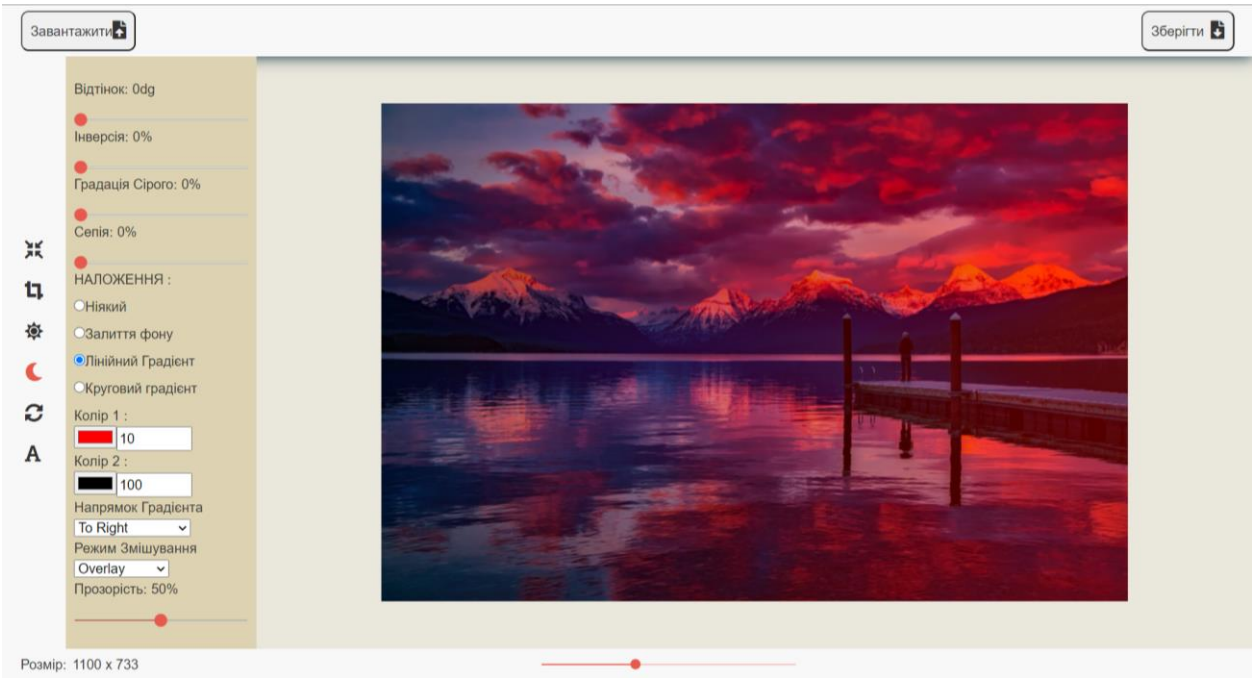

Рисунок 3.14 – Демонстрація застосування фільтру лінійний градієнт

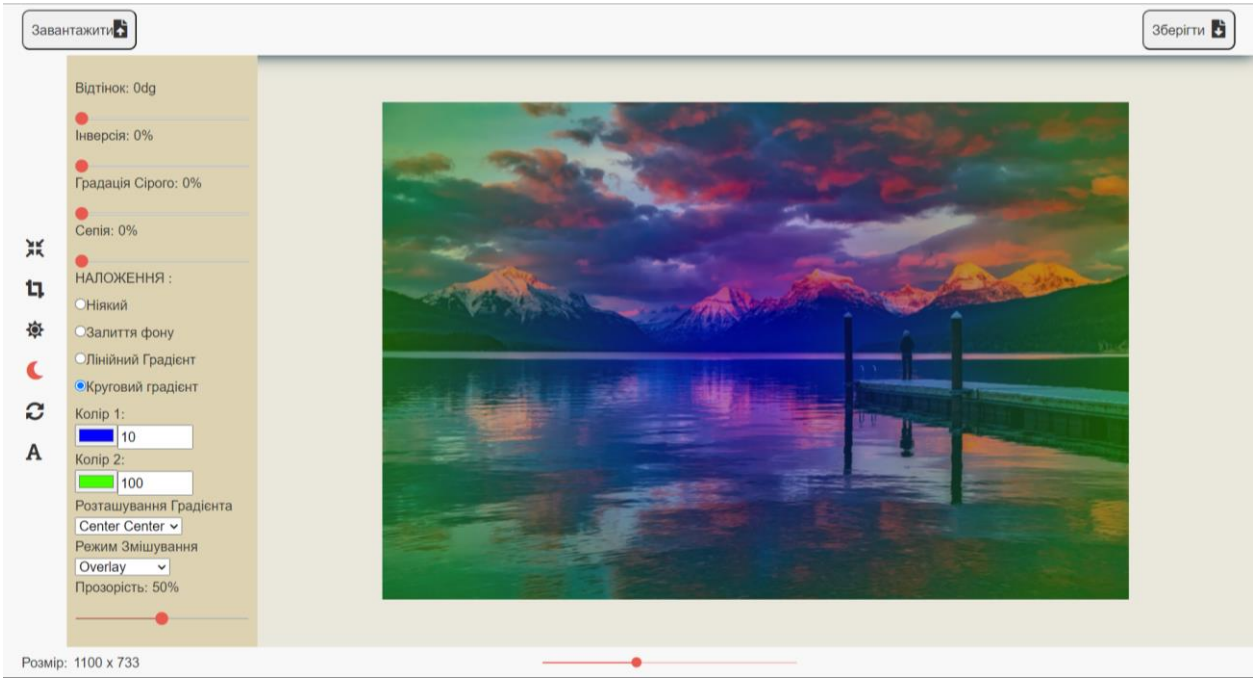

Рисунок 3.15 – Демонстрація застосування фільтру круговий градієнт

Функція обрізання зображення протестована та візуалізована на рисунках 3.16 та 3.17. Користувачі повинні мати можливість вибрати певну частину зображення та обрізати її відповідно, а інтерфейс точно відображатиме обрізаний результат.

Рисунок 3.16 – Вибір області для обрізання на зображенні

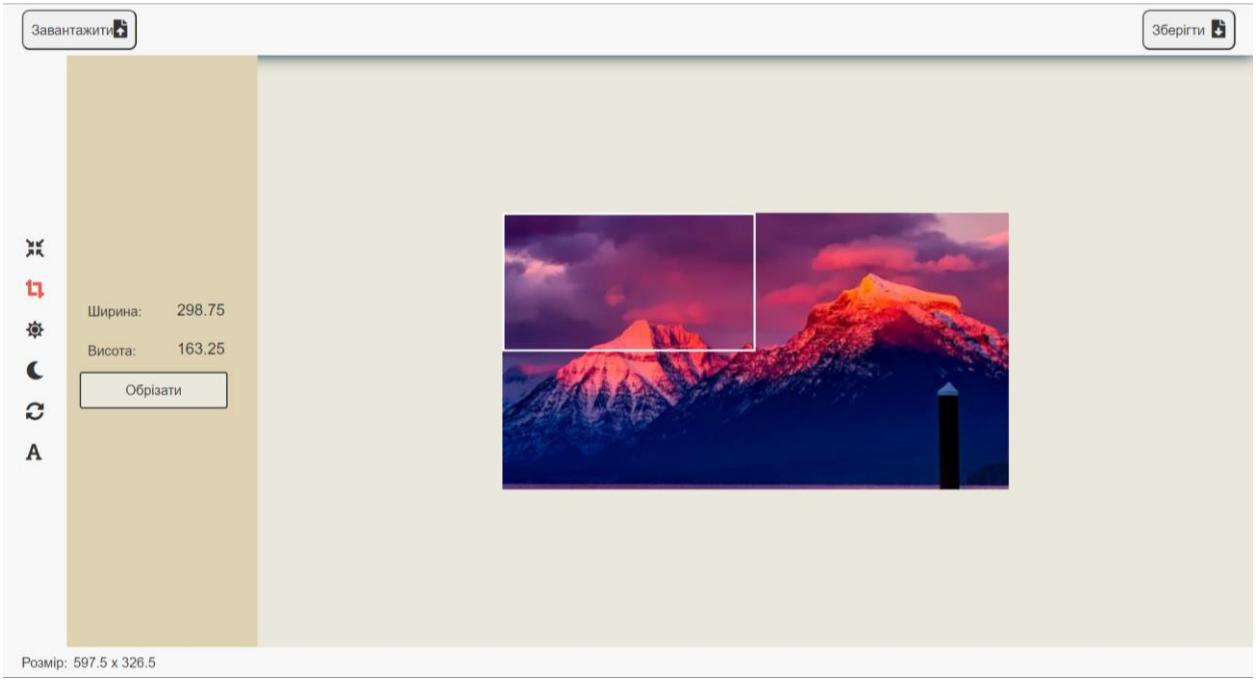

Рисунок 3.17 – Демонстрація обрізаного зображення

З наданих рисунків видно, що всі перевірені функції вебзастосунку працюють належним чином. Такі функції, як завантаження зображень, візуальне коригування, зміна розміру, дзеркальне відображення, додавання тексту, фільтри, заливка фону, ефекти градієнта та обрізання, функціонують належним чином, забезпечуючи плавну та надійну роботу користувача.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено вебзастосунок з редагування зображень. У ході роботи було розглянуто актуальність завдання редагування зображень, проведено аналіз потреб користувачів, визначені функціональні вимоги, досліджені моделі, методи та алгоритми для редагування зображень, і в кінцевому підсумку розроблений і протестований вебзастосунок.

Було створено функціональний та зручний вебзастосунок з редагування зображень. Ця програма задовольняє визначені потреби користувачів. Інтегровані модулі редагування зображень дозволяють користувачам виконувати широкий спектр завдань редагування у веббраузері.

Попри те, що процес розробки досяг бажаних цілей, важливо зазначити, що завжди є можливості для майбутніх покращень і вдосконалень. Наприклад, додаткові функції, такі як розширені фільтри або інтелектуальне розпізнавання зображень, можна інтегрувати у вебзастосунок для подальшого покращення її функціональності.

Підсумовуючи в рамках кваліфікаційної роботи було успішно розроблено вебзастосунок з редагування зображень. Ця програма відповідає завданням редагування зображень, задовольняє потреби користувачів і надає повний набір інструментів редагування. Проєкт не тільки робить внесок у сферу редагування зображень, але й демонструє можливості створення надійних вебзастосунків для складних завдань. З подальшим удосконаленням і розширенням цей вебзастосунок може стати цінним ресурсом для людей, які шукають зручну та потужну платформу для редагування своїх зображень в Інтернеті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mustafa, A. (2022, November 28). 40+ Best Photography & Photo Editing Statistics for 2023. PixelPhant. URL: https://pixelphant.com/blog/photography-and-photo-editing-statistics (дата звернення 16.05.2023).

2. Beautiful scene of alps road. Location National Park Tre Cime di Lavaredo, Dolomiti alp, Italy, Europe. Stock Photo. (n.d.). Adobe Stock. URL: https://stock.adobe.com/images/beautiful-scene-of-alps-road-location-national-park-tre-cime-di-lavaredo-dolomiti-alp-italy-europe/299942878?prev_url=detail&asset_id=299942858 (дата звернення 16.05.2023).

3. Bodyanskiy, Y., Grimm, P., Mashtalir, S., & Vinarski, V. (2010). Fast training of neural networks for image compression. In Advances in Data Mining. Applications and Theoretical Aspects: 10th Industrial Conference, ICDM 2010, Berlin, Germany, July 12-14, 2010. Proceedings 10 (pp. 165-173). Springer Berlin Heidelberg.

4. Машталір, С. В. (2016). Моделі та методи темпоральної обробки відео для інформаційного пошуку.

5. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2021). Methods of classification of images on the basis of the values of statistical distributions for the composition of structural description components. IEEE Access, 9, 92964-92973.

6. Mashtalir, S. V., Stolbovyi, M. I., & Yakovlev, S. V. (2019). Clustering video sequences by the method of harmonic k-means. Cybernetics and Systems Analysis, 55, 200-206.

7. Mashtalir, S., & Mashtalir, V. (2020). Spatio-temporal video segmentation. Advances in Spatio-Temporal Segmentation of Visual Data, 161-210.

8. Mashtalir, S., & Mashtalir, V. (2016, August). Sequential temporal video segmentation via spatial image partitions. In 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP) (pp. 239-242). IEEE.

9. Mashtalir, S., Mikhnova, O., & Stolbovyi, M. (2019). Multidimensional sequence clustering with adaptive iterative dynamic time warping. International Journal of Computing, 18(1), 53-59.

10. Mashtalir, S., & Mikhnova, O. (2017). Detecting Significant Changes in Image Sequences. Intelligent Systems Reference Library, 161–191.

11. Mashtalir, S., Mikhnova, O., & Stolbovyi, M. (2018, August). Sequence matching for content-based video retrieval. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 549-553). IEEE.

12. Marion, A. (2013). Introduction to image processing. Springer.

13. Sonka, M., Hlavac, V., & Boyle, R. (2014). Image processing, analysis, and machine vision. Cengage Learning.

14. Bilonoh, B., Bodyanskiy, Y., Kolchygin, B. V., & Mashtalir, S. (2021). Tunable Activation Functions for Deep Neural Networks. Springer eBooks, 624–633.

15. Gellersen, H. W., & Gaedke, M. (1999). Object-oriented web application development. IEEE Internet Computing, 3(1), 60-68.

16. Bodyanskiy, Y., Tyshchenko, O. K., & Mashtalir, S. (2019). Fuzzy Clustering High-Dimensional Data Using Information Weighting. Lecture Notes in Computer Science.

17. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2019). Online Robust Fuzzy Clustering of Data with Omissions Using Similarity Measure of Special Type. Advances in Intelligent Systems and Computing.

18. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 44(7), 3523-3542.

19. Banks, A., & Porcello, E. (2017). Learning React: functional web development with React and Redux. " O'Reilly Media, Inc.".

20. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2017). Video shot boundary detection via sequential clustering. International Journal “Information Theories and Applications, 24(1), 50-59.

21. Hu, Z., Mashtalir, S. V., Tyshchenko, O. K., & Stolbovyi, M. I. (2017). Video shots’ matching via various length of multidimensional time sequences. International Journal of Intelligent Systems and Applications, 9(11), 10.

22. Bodyanskiy, Y., Kinoshenko, D., Mashtalir, S., & Mikhnova, O. (2012). On-line video segmentation using methods of fault detection in multidimensional time sequences. International Journal of Electronic Commerce Studies, 3(1), 1-20.

23. Bogucharskiy, S. I., & Mashtalir, S. V. (2014). IMAGE SEQUENCES TEXTURE ANALYSIS BASED ON VECTOR QUANTIZATION. Radio Electronics, Computer Science, Control, (2).

24. Banks, A., & Porcello, E. (2020). Learning React: modern patterns for developing React apps. O'Reilly Media.

25. Haverbeke, M. (2018). Eloquent javascript: A modern introduction to programming. No Starch Press.

26. Herron, D. (2020). Node. js Web Development: Server-side web development made easy with Node 14 using practical examples. Packt Publishing Ltd.

27. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

28. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

29. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

30. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

31. Elrom, E., & Elrom, E. (2021). React Router and Material-UI. React and Libraries: Your Complete Guide to the React Ecosystem, 79-113.