МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА СЕРВІСУ ОНЛАЙН ТРАНСФОРМАЦІЇ ЗОБРАЖЕНЬ»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.С. Шейєрман
(ініціали та прізвище)
Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри комп'ютерних наук, професор, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шейєрману Володимиру Сергійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка сервісу онлайн трансформації зображень

керівник роботи Лісняк Андрій Олександрович, к.ф.-м.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Провести огляд та аналіз предметної області.
2. Здійснити проєктування системи.
3. Розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2025	
2.	Збір вихідних даних.	03.02.2025	
3.	Обробка методичних та теоретичних	03.03.2025	
	джерел.		
4.	Розробка першого та другого розділу.	07.04.2025	
5.	Розробка третього розділу.	12.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2025	

Студент _____ В.С. Шейєрман
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка сервісу онлайн трансформації зображень»: 50 с., 25 рис., 18 джерел.

ВЕБЗАСТОСУНОК, ЗОБРАЖЕННЯ, РЕДАГУВАННЯ, ТРАНСФОРМАЦІЯ, BACKEND, FRONTEND, JAVASCRIPT, KONVA, LARAVEL, PHP, POSTGRESQL, REACT, TAILWINDCSS.

Об’єкт дослідження – процес розробки вебзастосунку.

Мета роботи: розробка застосунку для онлайн-трансформації зображень засобами React та Laravel.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи проєктування, конструювання та тестування програмного забезпечення.

При розробці сервісу був проведений аналіз предметної області, обрано та спроєктовано архітектуру системи за допомогою UML у вигляді наборів діаграм прецедентів, класів, послідовностей, компонентів та розгортання. Реалізовано серверну частину з використанням Laravel, клієнтська частина була розроблена за допомогою бібліотеки React та фреймворку Konva.

У результаті роботи створено сервіс для онлайн-трансформації зображень.

SUMMARY

Bachelor’s qualifying paper “Development of an Online Service for Images Transformation”: 50 pages, 25 figures, 18 references.

BACK END, EDITING, FRONT END, IMAGE, JAVASCRIPT, KONVA, LARAVEL, PHP, POSTGRESQL, REACT, TAILWINDCSS, TRANSFORMATION, WEB APPLICATION.

The object of the study is the process of developing a web application.

The aim of the study is development of an application for online image transformation using React and Laravel.

The method of research are methods of collecting and analyzing software requirements, methods of designing, constructing and testing software.

During the development of the service, the subject area was analyzed, the system architecture was selected and designed using UML in the form of sets of diagrams of use-case, classes, sequences, components, and deployment. The server side was implemented using Laravel, the client side was developed using the React library and the Konva framework.

As a result of the work, a service for online image transformation was created.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Технічне завдання 9
1.1 Терміни та визначення 9
1.1.1 Загальні терміни 9
1.1.2 Технічні терміни 9
1.2 Функціональні вимоги 10
1.2.1 Призначення і цілі створення системи 10
1.2.2 Загальні функціональні можливості системи 10
1.3 Нефункціональні вимоги 11
1.3.1 Інтерфейс користувача 11
1.3.2 Підтримка браузерів 11
1.3.3 Вимоги до продуктивності 12
1.3.4 Вимоги до безпеки 12
1.4 Опис предметної області 12
1.5 Опис системи 13
1.6 Порівняння з подібними системами 14
2 Проєктування 19
2.1 Використання UML під час розробки системи 19
2.2 Діаграма прецедентів 20
2.2.1 Опис прецедентів 21
2.3 Діаграма класів 25
2.4 Діаграма послідовності 27
2.5 Діаграма компонентів 29
2.6 Діаграма розгортання 31
2.7 Проєктування інтерфейсу 32

3 Реалізація та тестування 35
3.1 Опис інструментів розробки 35
3.2 Серверна частина 36
3.3 Клієнтська частина 37
3.3.1 Компоненти автентифікації 37
3.3.2 Компоненти редактора 38
3.3.3 Компоненти трансформації 38
3.4 Керівництво користувача 40
3.4.1 Рівень підготовки користувача 40
3.4.2 Підготовка до роботи 40
3.4.3 Вхід до системи 40
3.4.4 Завантаження зображень 42
3.4.5 Перемикання режимів редагування 43
3.4.6 Виконання трансформацій 44
3.4.7 Збереження результатів 45
3.5 Тестування системи 47
Висновки 48
Перелік посилань 49

ВСТУП

У сучасному цифровому середовищі, коли візуальний контент відіграє ключову роль у комунікації, маркетингу та творчості, інструменти для обробки зображень набувають все більшого значення. Зростання популярності соціальних мереж, електронної комерції та цифрових медіа зумовлює високий попит на зручні, швидкі та функціональні засоби для трансформації зображень.

Розробка вебзастосунку для трансформації зображень визначається як актуальний напрямок у сфері веброзробки. Такий застосунок дозволяє редагувати фотографії без встановлення спеціального програмного забезпечення, а також реалізує можливість обробки як одного, так і кількох зображень одночасно. Враховуючи потреби користувачів у гнучкості, швидкості та якості інструментів редагування, створення подібної системи є вагомим технічним і прикладним завданням.

Метою даної кваліфікаційної роботи є розробка вебзастосунку для онлайн-трансформації зображень із використання Laravel, React та Konva. Для досягнення цієї мети передбачено вирішення таких задач:

– аналіз предметної області;
– проєктування архітектури системи;
– вибір та обґрунтування технологій розробки застосунку;
– реалізація клієнтської та серверної частини;
– тестування функціоналу та оцінка результату.

Об’єкт дослідження – процес розробки вебзастосунку.

Перший розділ присвячено аналізу предметної області та створенню технічного завдання. У другому розділі виконано проєктування архітектури розроблюваної системи. Третій розділ містить огляд використаних технологій, детальний опис реалізації програмного модуля. Робота завершується висновками та оцінкою перспектив розвитку системи.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Зображення – цифровий графічний файл, що є об’єктом обробки в межах системи.

Мегапіксель (МП) – один мільйон пікселей, що формують зображення [1].

Трансформація зображень – це процес зміни будь-яких властивостей зображення, включаючи його розмір, формат, візуальні та колірні характеристики [2].

Вебзастосунок – програмний продукт, який працює у веббраузері та забезпечує доступ до функціоналу системи через мережу Інтернет [3].

Онлайн-редактор зображень – функціональний модуль вебзастосунку, який дозволяє користувачу здійснювати трансформацію зображень.

Користувач – особа, яка взаємодіє із системою для редагування зображень через інтерфейс вебзастосунку.

Інтерфейс користувача – візуальна частина системи, з якою безпосередньо взаємодіє користувач.

1.1.2 Технічні терміни

Неавторизований користувач – гість системи, який не проходив процедуру реєстрації або входу.

Авторизований користувач – користувач, який має зареєстрований обліковий запис у системі та здійснив вхід до неї.

Одинична обробка – режим роботи, у якому здійснюється трансформація одного зображення.

Пакетна обробка – режим роботи, у якому одночасно здійснюється обробка кількох зображень з однаковими параметрами трансформації.

Сесія – період активної взаємодії користувача із системою від моменту входу до виходу або завершення роботи.

Демонстраційний режим – обмежений функціонал, доступний неавторизованому користувачу.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати можливість онлайн трансформації зображень.

Експлуатаційне призначення системи – система може використовуватися авторизованими та неавторизованими користувачами.

Мета створення системи – розробка системи онлайн трансформації зображень.

1.2.2 Загальні функціональні можливості системи

Система має надавати неавторизованим користувачам такі можливості:

– перегляд головної сторінки із загальною інформацією про застосунок;
– використання демонстраційного режиму;
– перехід до форм реєстрації або входу у систему.

Система має надавати авторизованим користувачам такі можливості:

– доступ до основного сервісу;

– завантаження одного або декількох зображень для редагування;
– виконання трансформації зображень;
– видалення зображень у власній сесії;
– збереження результатів редагування у локальне або хмарне сховище;
– вихід із системи.

1.3 Нефункціональні вимоги

1.3.1 Інтерфейс користувача

Система повинна коректно відображати інтерфейс користувача на пристроях з великим екраном: настільні комп’ютери, ноутбуки та планшети. Усі елементи управління повинні бути розташовані логічно, з урахуванням особливостей користування зазначеними пристроями. Усі кнопки, іконки, повідомлення та написи повинні мати чітке текстове або графічне пояснення свого призначення.

Розмітка інтерфейсу застосунку має забезпечувати чіткій поділ на функціональні зони: навігаційна панель, область відображення зображення та панель інструментів трансформації.

1.3.2 Підтримка браузерів

Система повинна коректно функціонувати для наступних браузерів останніх версій:

– Google Chrome;
– Mozilla Firefox;
– Microsoft Edge;
– Safari.

Підтримка застарілих версій браузерів не передбачається. У разі використання застарілого браузера, система повинна виводити відповідне попередження користувачу.

1.3.3 Вимоги до продуктивності

Система повинна забезпечувати швидке редагування на дії користувача без помітних затримок. Час відкриття основного інтерфейсу не повинен перевищувати 2 секунд при середньому інтернет-з’єднанні. Операції трансформації зображень мають виконуватись у межах до 500 мс для кожного зображення розміром 2 МП.

Пакетна обробка повинна підтримувати одночасну роботу щонайменше зі 100 зображеннями без критичного зниження швидкості роботи або збоїв у системі.

1.3.4 Вимоги до безпеки

Система повинна забезпечувати захист персональних даних користувачів. Доступ до основного функціоналу дозволений лише після успішної авторизації користувача. Аутентифікація користувача має здійснюватися за допомогою токенів доступу. Система повинна мати захист від типових вебатак.

1.4 Опис предметної області

Предметною областю є розробка системи для онлайн трансформації зображень. Дана система повинна надавати користувачам можливість

завантаження зображень, їх редагування за допомогою вбудованих інструментів трансформації та збереження результатів. Процес взаємодії з системою проходить наступним чином. Користувач повинен ввести адресу вебзастосунку у браузері, після чого потрапляє на головну сторінку, де має змогу ознайомитись із загальною інформацією про систему.

Якщо користувач ще немає облікового запису, йому необхідно пройти реєстрацію, після чого він отримає доступ до повного функціоналу системи. У разі, якщо користувач вже зареєстрований, він може здійснити вхід через відповідну форму. Без реєстрації користувачу доступний лише демонстраційний режим, у межах якого дозволяється редагування одного зображення без можливості збереження.

Після входу до системи користувач має можливість завантажити одне або кілька зображень. У системі реалізовано два основні режими роботи: одиночне та пакетне редагування. У першому випадку користувач працює з одним зображенням; у другому – має змогу обробляти відразу кілька зображень однаковим набором трансформацій. Доступні трансформації включають зміну розміру, обрізання, обертання та застосування фільтрів. Після редагування користувач може зберегти результат на свій пристрій або в хмарне сховище.

Процес взаємодії із зображенням проходить через інтерфейс, поділений на зони: навігаційна панель, область відображення зображення та панель інструментів трансформації. Користувач обирає потрібний інструмент, задає параметри трансформації, а після завершення редагування натискає кнопку збереження, після чого система генерує фінальний файл.

1.5 Опис системи

Обробка зображень є важливим інструментом у роботі користувачів, які нормалізують візуальний контент для публікації, презентацій чи архівації. Зазвичай така обробка виконується із використанням встановленого

програмного забезпечення. Це ускладнює робочий процес, робить його залежним від конкретного пристрою та знижує ефективність. У зв’язку з цим актуальною є розробка систем, які надають можливість онлайн трансформації зображень у пакетному режимі.

Системи з підтримкою пакетної обробки зображень стають незамінними інструментами для користувачів, яким потрібно швидко виконати однакові операції над багатьма файлами. Завдяки таким системам зменшується кількість рутинних дій, оптимізується час редагування та підвищується продуктивність. Користувачі отримують змогу звантажити декілька зображень одночасно, застосувати до них однаковий набір трансформацій та зберегти результати у потрібному форматі.

Однією з таких систем є «Сервіс онлайн трансформації зображень». Нова система, яка надає користувачу можливість швидко редагувати велику кількість зображень.

Можливості системи:

– завантаження одного або декількох зображень;
– виконання базових трансформацій;
– застосування трансформацій до групи у пакетному режимі;
– збереження результатів обробки у локальне та хмарне сховище.

1.6 Порівняння з подібними системами

Для порівняння подібних систем та визначення недоліків та переваг майбутнього сервісу було розглянуто три популярні онлайн-сервіси для трансформації зображень: Pixlr, Fotor та Picsart. Усі три системи підтримують одиночне редагування зображень, а також мають реалізовану функцію пакетної обробки.

Pixlr – було запущено наприкінці літа 2008 року, як онлайн-редактор зображень, розроблений Олою Севандерссоном. Спершу це була просто вебплатформа для базового редагування фото, яка швидко стала популярною

завдяки інтуїтивному інтерфейсу та доступному функціоналу. Сервіс має три різних види передплати: Plus, Premium та Teams [4].

Pixlr Batch Editor (див. рис. 1.1) надає можливість пакетної трансформації зображень. Він дозволяє користувачам редагувати до 100 зображень одночасно, застосовуючи однакові налаштування, такі як обрізання, зміна розміру та фільтри. Користувачі можуть зберігати налаштування як макроси для подальшого використання. Після завершення редагування користувачі можуть експортувати зображення у форматах JPEG, PNG та WebP.

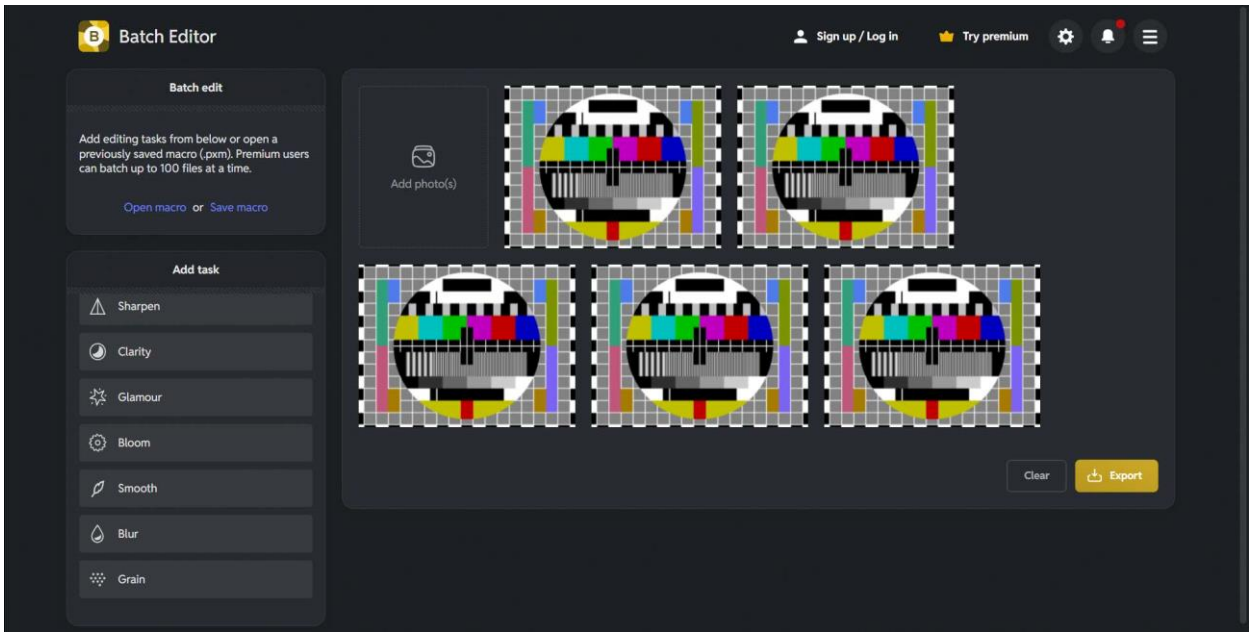


Рисунок 1.1 – Pixlr Batch

Редактор має низку суттєвих обмежень. При завантаженні великої кількості зображень були виявлені затримки, що вказує на погану оптимізацію обробки. У безкоштовній версії присутня настирлива реклама, яка знижує комфорт роботи. Користувачі без платної передплати обмежені у кількості збережень на день та доступі до деяких функцій.

Fotor – це комплексна творча платформа для трансформації зображень, запущений у 2012 році. Сервіс має безкоштовну версію, підписки Pro та Pro+ [5].

Fotor Batch Editor (див. рис. 1.2) надає можливість пакетного редагування зображень. Цей інструмент дозволяє користувачам редагувати до 50 зображень одночасно, застосовуючи налаштування, такі як обрізання, зміна розміру, видалення заднього фону, перетворення в інший формат та стиснення.

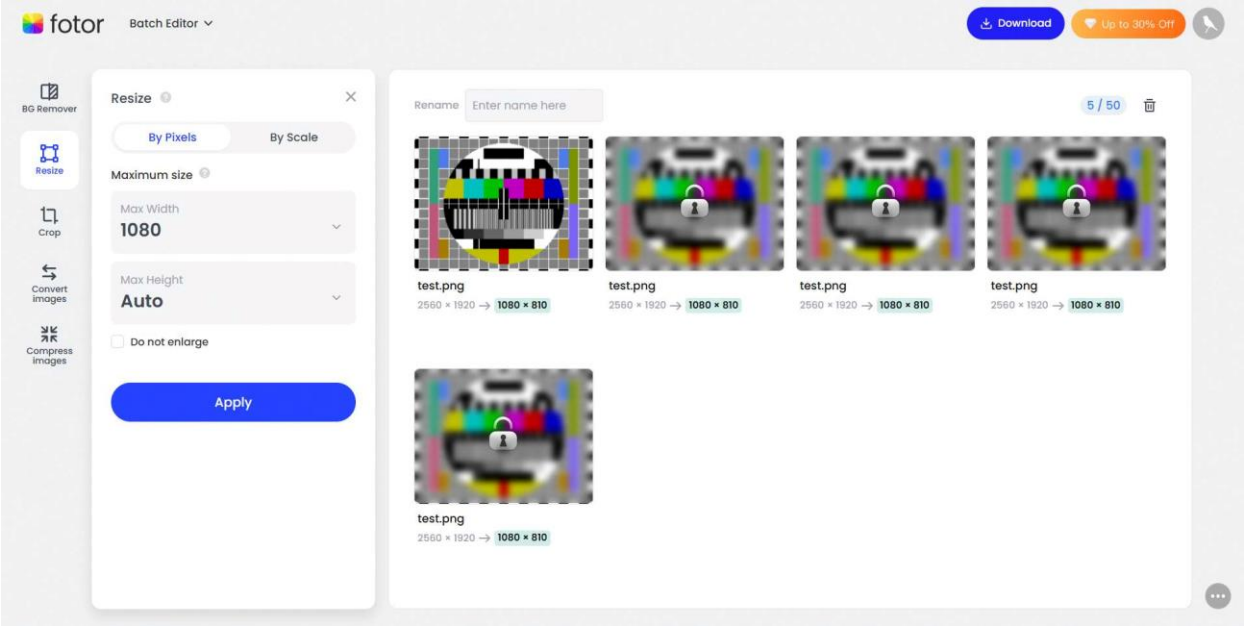


Рисунок 1.2 – Fotor Batch Editor

Незважаючи на широкий функціонал, сервіс має низку суттєвих недоліків, особливо для безкоштовних користувачів. Одним із найбільш критичних є велика кількість реклами, яка постійно з’являється під час редагування зображень. Вона не лише ускладнює інтерфейс, але й помітно знижує швидкість взаємодії з сервісом. Окрім того, функція пакетного редагування, фактично недоступна для безкоштовного використання – користувач без активної підписки може редагувати лише одне зображення за сесію.

Picsart – це універсальна платформа, де користувачі можуть створювати, налаштовувати та ділитись зображеннями та відео. Сервіс працює за моделлю “freemium”, де базові функції доступні безкоштовно, а розширені можливості відкриваються при підписці на Plus та Pro [6].

Picsart Batch Editor (див. рис. 1.3) дозволяє редагувати до 50 зображень одночасно, застосовуючи наступні налаштування: заміна заднього фону, покращення за допомоги штучного інтелекту, обрізання, обертання, масштабування та накладання мокрої печатки. Безкоштовна версія достатньо сильно обмежена у функціоналі, наприклад для експорту зображення у форматі PNG потрібно мати підписку, а експорт у взагалі неможливий.



Рисунок 1.3 – Picsart Batch Editor

Порівнюючи вищезгадані сервіси із системою, що розробляться, можна виокремити низку переваг для неї. Система фокусується на простоті та доступності для користувача без зайвих функцій і маркетингових обмежень. Ключова відмінність – можливість повноцінного використання пакетного режиму без необхідності придбання підписки. Відсутність реклами та фокус на ефективній роботі з кількома зображеннями одночасно дозволяють створити більш комфортний досвід взаємодії. Водночас система має і низку обмежень, які варто враховувати. Відсутність мобільної версії або адаптації для смартфонів. Редактор не підтримує роботу з векторною графікою та не має AI-функціоналу.

Отже, порівняння подібних систем показало, що попри наявні рішення у сфері онлайн трансформації зображень, розроблювана система вирізняється

завдяки чіткій орієнтації на пакетну обробку зображень. Система орієнтована на прості й зрозумілі сценарії використання без перевантаження додатковими можливостями.

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

Unified Modeling Language – це мова моделювання загального призначення. Головною метою UML є визначення стандартного способу візуалізації того, як система буде розроблена. Вона дуже схожа на креслення, що використовуються в інших галузях інженерії. UML – це не мова програмування, це радше мова візуального представлення [7].

Використання UML під час розробки системи є ключовим етапом у процесі проєктування програмного забезпечення.

UML забезпечує стандартизований візуальний інструментарій, який дозволяє формалізувати структуру системи, її компоненти, поведінку користувачів, взаємодії між модулями та середовище розгортання. Завдяки цьому стає можливим більш глибоке розуміння системи ще до початку її реалізації.

Уміння користуватись UML також сприяє покращенню комунікації між учасниками процесу розробки, а також між розробниками та замовником. Візуалізація складної структури системи у вигляді графічних діаграм допомагає узгодити бачення функціональності, виявити потенційні помилки або суперечності в логіці ще на етапі проєктування, що в результаті знижує втрати на виправлення помилок у пізніших фазах життєвого циклу розробки.

UML-діаграми дозволяють ефективно моделювати як статичні, так і динамічні аспекти системи.

У даній роботі UML було застосовано для створення діаграми прецедентів, діаграми класів, діаграми послідовності, діаграми компонентів та діаграми розгортання.

2.2 Діаграма прецедентів

Діаграми прецедентів описують високорівневі функції та області застосування системи. Ці діаграми також визначають взаємодію між системою та її акторами. Прецеденти та актори описують, що робить система та як актори її використовують, але не те, як система працює внутрішньо [7].

Діаграми прецедентів ілюструють та визначають контекст і вимоги або всієї системи, або важливих частин системи. Для моделювання складної системи можна використати одну діаграму або створити багато діаграми для моделювання компонентів системи. Зазвичай ці діаграми розробляються на ранніх етапах проєкту та звертаються до них протягом усього процесу розробки [7].

На рисунку 2.1 представлена діаграма прецедентів.

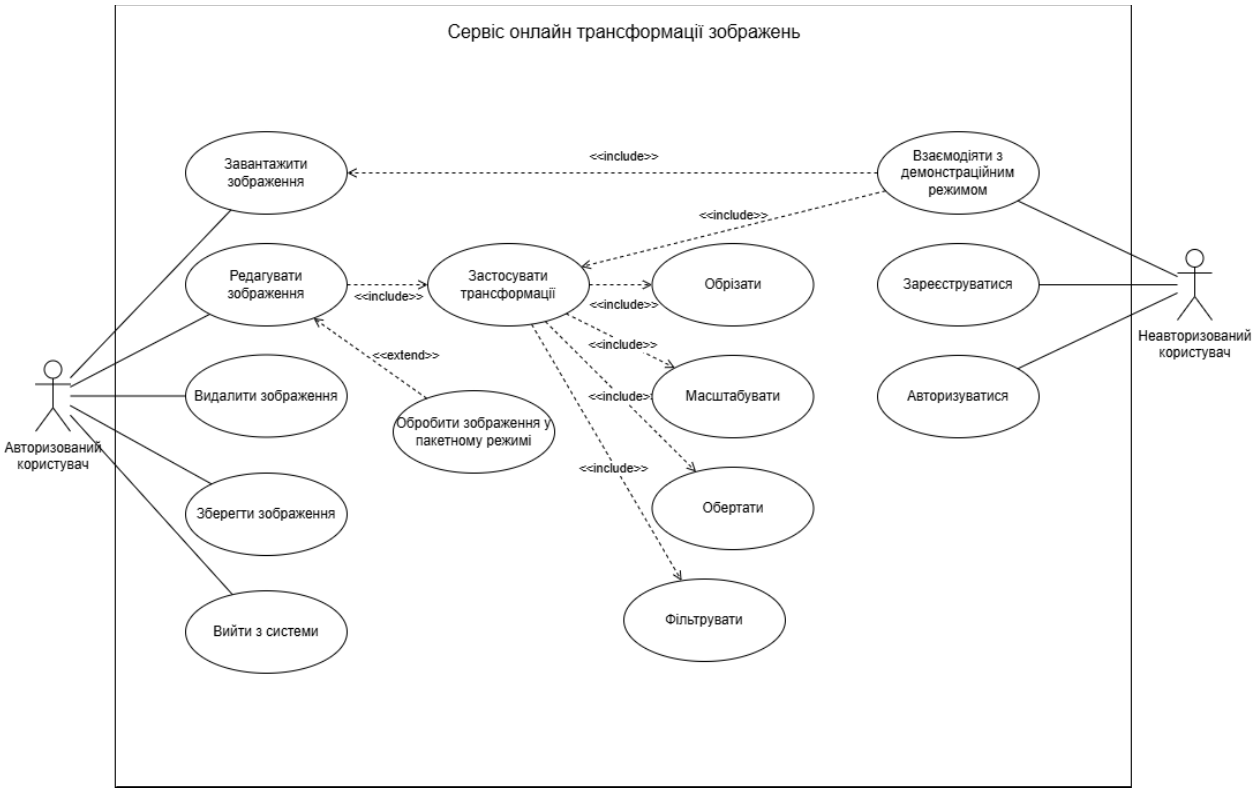


Рисунок 2.1 – Діаграма прецедентів

На діаграмі представлено двох акторів: «Неавторизований користувач» та «Авторизований користувач». Перший має доступ до базового функціоналу

системи, зокрема реєстрація та авторизація облікового запису, а також використання демонстраційного режиму. Другий актор отримує повний доступ до функцій системи трансформації зображень.

Основним прецедентом для актора «Авторизований користувач» є «Редагувати зображення», який включає в себе виконання трансформацій зображення. Прецедент «Редагування зображення» обов'язково включає «Застосувати трансформації», що, у свою чергу, включає такі базові дії як: «Обрізати», «Масштабувати», «Обертати» та «Фільтрувати».

У системі передбачено розширення у вигляді прецеденту «Обробити зображення у пакетному режимі», яке дозволяє застосовувати трансформації одночасно до кількох файлів. Окрім редагування, актор може «Завантажити зображення», «Зберегти зображення» або «Видалити зображення». Для завершення сесії є прецедент «Вийти з системи». Така структура дозволяє управляти як одиночним, так і пакетним редагування зображень у межах одного інтерфейсу.

2.2.1 Опис прецедентів

Прецедент «Взаємодіяти з демонстраційним режимом» (прецедент, доступний лише для неавторизованого користувача).

Призначення: даний прецедент надає неавторизованому користувачу можливість ознайомитися з базовим функціоналом системи через спрощений режим редагування зображення без авторизації.

Основний потік подій: прецедент починається автоматично при завантаженні головної сторінки. У нижній частині сторінки відображається вбудована демонстраційна зона, яка є спрощеною версією основного редактора. Користувач має можливість завантажувати одне зображення з локального пристрою, застосувати базові трансформації, але без можливості зберігати результат.

Передумова: користувач повинен бути неавторизованим та перебувати на головній сторінці системи.

Виняткова ситуація 1: користувач намагається завантажити друге зображення – система пропонує замінити попереднє або завершити дію.

Виняткова ситуація 2: користувач натискає кнопку «Зберегти» – система блокує операцію, виводячи повідомлення про те, що збереження недоступне у демонстраційному режимі, та пропонує пройти реєстрацію.

Прецедент «Завантажити зображення».

Призначення: даний прецедент надає користувачу можливість додавати одне або кілька зображень до системи для подальшого редагування.

Основний потік подій: прецедент починається, коли користувач натискає кнопку «Відкрити» в інтерфейсі системи. Система пропонує вибрати джерело: з локального пристрою або зі хмарного сховища. Після вибору та підтвердження, обрані файли додаються у список зображень користувача, відображаються у відповідному інтерфейсі та стають доступними для редагування.

Передумова: користувач повинен бути авторизований у системі.

Виняткова ситуація 1: користувач не вибрав жодного файлу або скасував дію – система повертається до попереднього стану без завантаження.

Виняткова ситуація 2: формат файлу не підтримується – система виводить повідомлення про помилку.

Виняткова ситуація 3: файл перевищує допустимий розмір – система не завантажує зображення та повідомляє користувача.

Прецедент «Редагувати зображення» (загальний прецедент для групування дій над трансформацією зображень).

Призначення: даний прецедент надає можливість авторизованому користувачу виконувати базові дії над зображеннями.

Основний потік подій: прецедент починає виконуватись після вибору користувачем одного із завантажених зображень у системі. Система відображає інтерфейс редактора з панеллю інструментів, що дозволяє застосовувати різні трансформації.

Альтернативний потік подій: якщо зображення не було обране, система повідомляє про необхідність попереднього завантаження зображення.

Передумова: перед початком виконання даного прецеденту користувач повинен завантажити зображення.

Прецедент «Застосувати трансформації» (прецедент, який має зв’язок “include” з батьківським прецедентом «Редагувати зображення»).

Призначення: даний прецедент надає користувачу можливість виконувати трансформації над вибраним зображенням.

Основний потік подій: прецедент починається з моменту вибору інструмента трансформації у редакторі. Користувач взаємодії з відповідними елементами керування. Після налаштування параметрів, трансформація застосовується до зображення в редакторі.

Альтернативний потік подій: якщо трансформація не може бути застосована через помилку в параметрах, система повідомляє про це повідомленням і не вносить зміни.

Передумова: користувач повинен перебувати в режимі редагування обраного зображення.

Прецедент «Обрізати» (прецедент, який має зв’язок “include” з батьківським прецедентом «Застосувати трансформації»).

Призначення: даний прецедент надає користувачу можливість обрізати зображення, тобто виділити його частину шляхом відсікання зайвих фрагментів.

Основний потік подій: прецедент починається з моменту вибору інструмента «Обрізати» в інтерфейсі редактора. Користувач виділяє бажану область зображення за допомогою рамки, за потреби змінює її розміри або переміщує, після чого натискає кнопку «Застосувати». Система обрізає зображення відповідно до вибраної області та оновлює його відображення.

Передумова: користувач активувати інструмент трансформації.

Виняткова ситуація 1: якщо користувач не виконав жодних дій з рамкою обрізання – система повертається до попереднього вигляду зображення без змін.

Виняткова ситуація 2: виділена область виходить за межі зображення – система автоматично обмежує область обрізання до допустимих розмірів.

Виняткова ситуація 3: користувач натиснув «Скасувати» – система не вносить жодних змін і повертає користувачу до загального інтерфейсу редагування.

Прецедент «Обробити зображення у пакетному режимі» (прецедент, який має зв’язок “extend” з батьківським прецедентом «Редагувати зображення»).

Призначення: даний прецедент надає авторизованому користувачу можливість редагувати одразу кілька зображень однаковими трансформаціями.

Основний потік подій: прецедент починається після того, як користувач переходить у режим «Пакетна обробка». Користувач завантажує декілька зображень, обирає трансформації та застосовує їх до всіх вибраних файлів. Після обробки система відображає оновлені версії зображень.

Передумова: користувач повинен завантажити зображення та перейти у режим пакетного редагування.

Виняткова ситуація 1: якщо виникла помилка при застосуванні трансформації до окремого файлу – система інформує користувача, але продовжує обробку інших файлів.

Виняткова ситуація 2: якщо користувач спробує застосувати трансформації без вибору жодного зображення – система повідомить про необхідність обрати хоча б одне зображення.

Прецедент «Зберегти зображення».

Призначення: прецедент надає користувачу можливість зберегти відредаговане зображення або серію зображень на локальний пристрій або у хмарне сховище.

Основний потік подій: прецедент починається з моменту натискання кнопки «Зберегти». Система пропонує вибрати місце збереження: локальне або хмарне сховище. Далі користувачу пропонується змінити назву та формат

зображення. Після підтвердження система виконує генерацію файлу та ініціює збереження. У разі пакетного режиму всі зображення зберігаються у вигляді архіву.

Передумова: зображення має бути готове до збереження.

Виняткова ситуація 1: користувач скасовує збереження або закриває вікно – система повертається до інтерфейсу редагування без змін.

Виняткова ситуація 2: неможливо підключитися до хмарного сховища – система пропонує зберегти на локальний пристрій.

2.3 Діаграма класів

Діаграми класів – це тип діаграм UML, що використовується для візуального представлення структури та зв’язків класів у системі. На цих діаграмах класи зображуються у вигляді блоків, кожен з яких містить три відсіки для назви класу, атрибутів та методів. Лінії, що з’єднують класи, ілюструють зв’язки. Діаграми класів надають загальний огляд дизайну системи, допомагаючи комунікувати та документувати структуру програмного забезпечення [8].

В об’єктно-орієнтованому програмуванні клас – це план або шаблон створення об’єктів. Об’єкти є екземплярами класів, і кожен клас визначає набір атрибутів та методів, які матимуть об’єкти, створені з цього класу. Атрибути представляють характеристики або властивості об’єкта, тоді як методи визначають поведінку або дії, які об’єкт може виконувати [8].

На рисунку 2.2 представлена діаграма класів. На діаграмі представлено структуру серверної частини системи. Вона відображає основні класи, їхні атрибути, методи, а також логічні зв’язки між ними відповідно до специфікації UML.

Центральними класами системи є User та Image, які моделюють користувача і його зображення відповідно. Клас User має атрибути, що

зберігають основну облікову інформацію: ім’я, електрону пошту, пароль, дати створення та оновлення. Він пов’язаний з класом Image асоціативним зв’язком типу «один до багатьох», тобто користувач може мати багато зображень, метод images() визначає цей зв’язок. Клас Image містить атрибути, що описують зображення: назва, шлях, ширину, висоту, формат, розмір, дата створення та оновлення, а також метод user(), який вказує на зворотний зв’язок із користувачем.

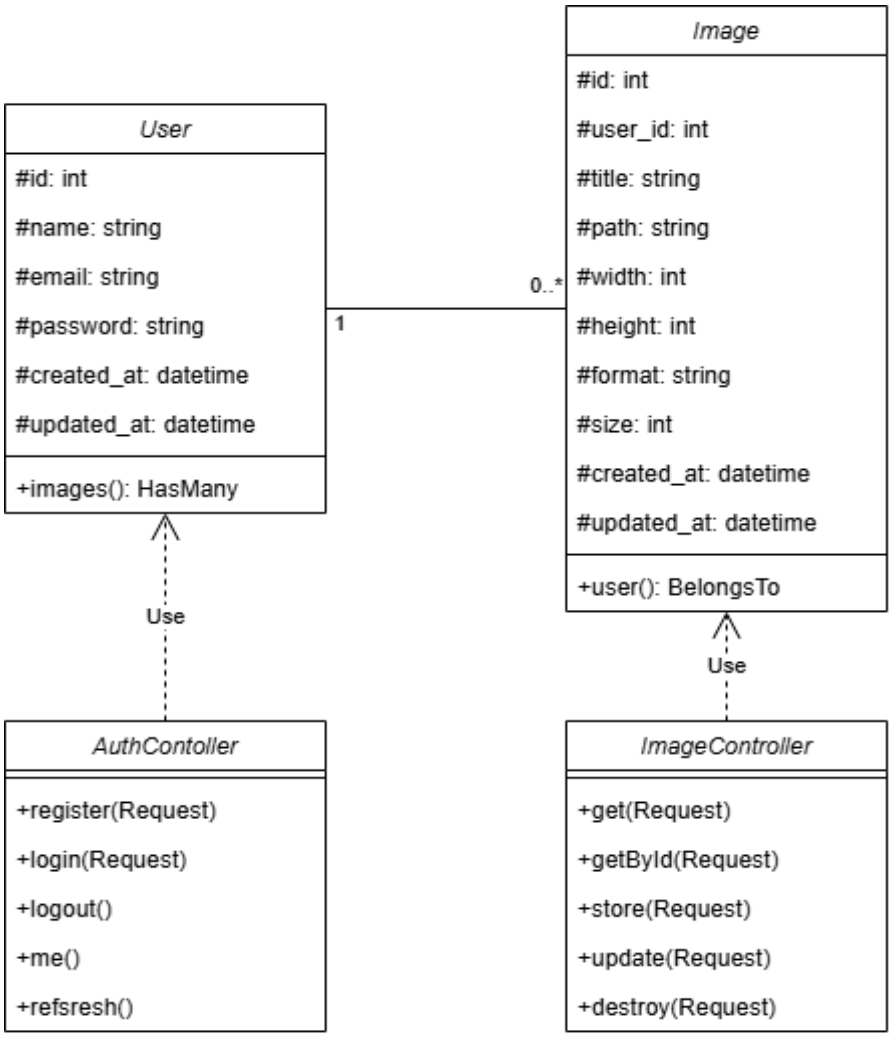


Рисунок 2.2 – Діаграма класів

Клас AuthController реалізує функціонал, пов’язаний з автентифікацію користувача. Його методи забезпечують створення облікового запису, вхід, вихід та оновлення сесії. Він використовує функціонал класу User для доступу до облікових даних.

Клас ImageController відповідає за роботу із зображеннями: завантаження, отримання, оновлення та видалення. Він використовує функціонал класу Image, що дозволяє взаємодіяти з базою даних зображень користувачів.

Діаграма чітко демонструє залежності між контролерами та відповідними моделями, а також внутрішні зв’язки між користувачем і його зображеннями. Така структура дозволяє масштабувати серверну логіку, зберігаючи її чітко розділеною та керованою.

2.4 Діаграма послідовності

Діаграма послідовності – це діаграм UML, яка демонструє послідовність повідомлень між об’єктами під час взаємодії. Ця діаграма складається з групи об’єктів, представлених лініями життя, та повідомлень, якими вони обмінюються з часом під час взаємодії [8].

Діаграма послідовності показує порядок викликів і взаємозв’язки між компонентами системи в процесі реалізації конкретного сценарію. Вона дозволяє моделювати високорівневу взаємодію між активними об’єктами, деталізувати механізми їхньої співпраці у межах певного прецеденту, а також відобразити як загальні шаблони взаємодії з усіма можливими шляхами виконання, так і окремі приклади з одним варіантом проходження сценарію.

На рисунку 2.3 представлена діаграма послідовності для вдалого сценарію збереження зображення у хмарне сховище Amazon S3.

Учасниками взаємодії на діаграмі виступають: користувач, frontend, backend, AWS S3 SDK та сервіс Amazon S3.

Процес розпочинається з того, що користувач обирає зображення у системі. Після цього frontend надсилає HTTP-запит методом POST на маршрут /api/images, передаючи файл. Backend приймає запит і виконує валідацію файлу.

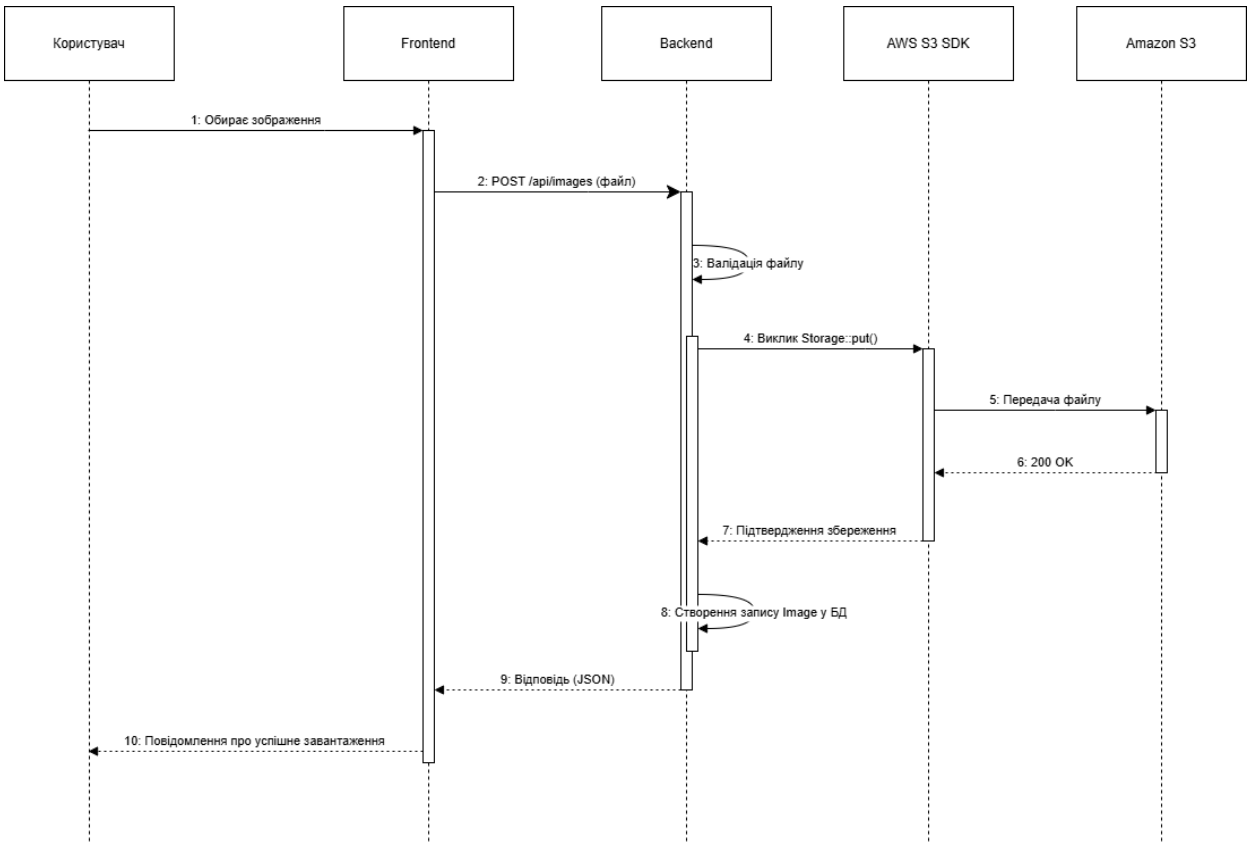


Рисунок 2.3 – Діаграма послідовності для збереження зображення у S3

У разі успішної перевірки backend викликає метод Storage::put() через AWS S3 SDK, який ініціює передачу файлу до хмарного сховища Amazon S3. Після завершення завантаження сервіс Amazon S3 повертає статус 200 OK, що свідчить про успішне збереження файлу.

Після цього backend фіксує успішне збереження, створює відповідний запис у базі даних та надсилає відповідь у форматі JSON до frontend, яка містить необхідну інформацію про збережене зображення, зокрема шлях або URL. Завершується процес тим, що користувач отримує повідомлення про успішне збереження зображення.

На рисунку 2.4 представлена діаграма послідовності для вдалого сценарію завантаження зображення з хмарного сховища Amazon S3.

Учасниками взаємодії на діаграмі виступають: користувач, frontend, backend та сервіс Amazon S3.Сценарій починається з того, що користувач відкриває галерею у вебзастосунку. У відповідь frontend ініціює запит до серверу GET /api/images з метою отримання списку збережених зображень.

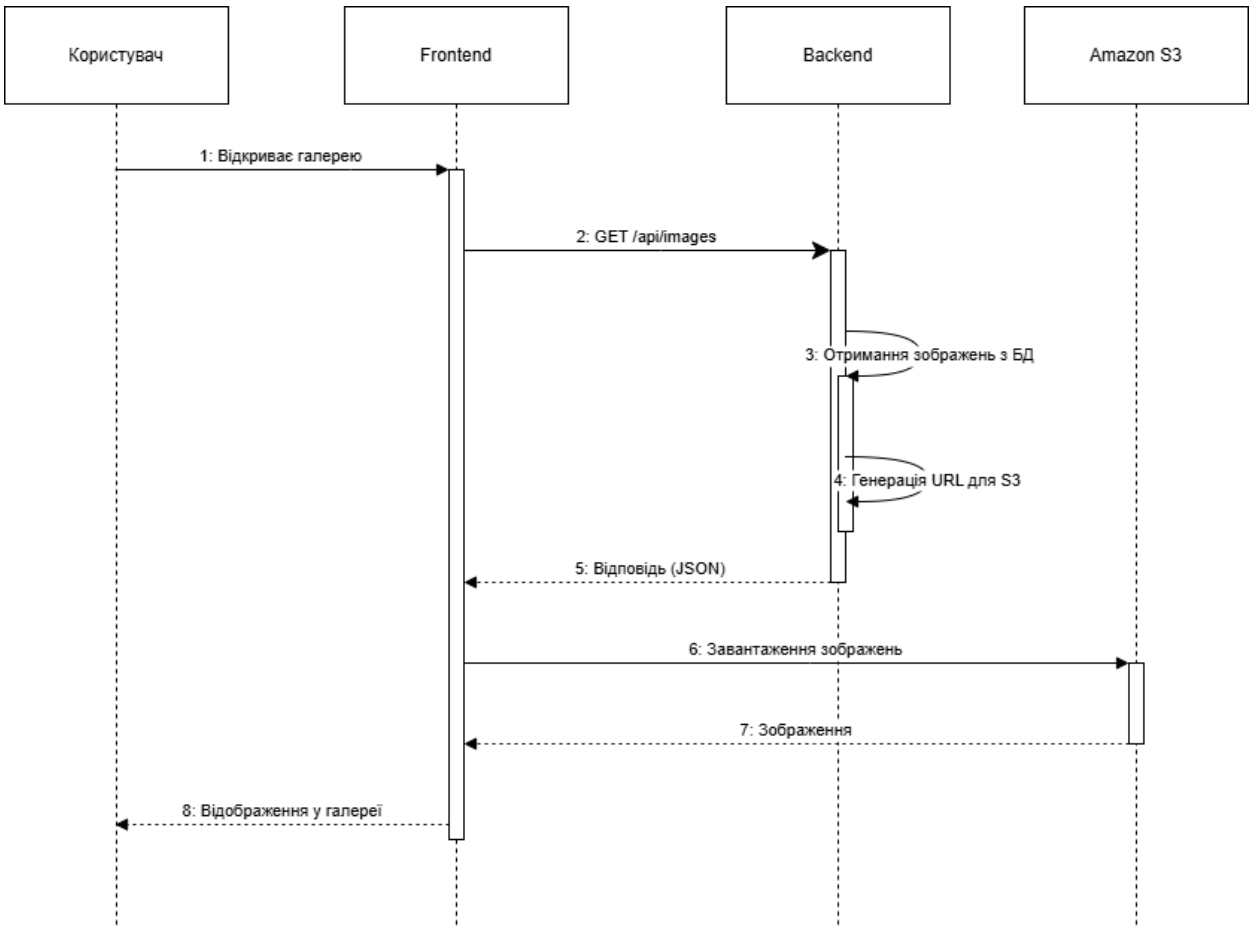


Рисунок 2.4 – Діаграма послідовності для завантаження зображення з S3

Після отримання запиту backend звертається до бази даних, щоб витягнути записи, пов’язані з зображеннями користувача. Далі сервер генерує посилання на об’єкти у сховищі S3, це можуть бути публічні або тимчасово підписані URL. Сформовані посилання надсилаються у відповідь frontend у форматі JSON.

Frontend, отримавши список URL, ініціює завантаження самих зображень безпосередньо з Amazon S3. Після завантаження, зображення відображаються у галереї користувача.

2.5 Діаграма компонентів

Діаграма компонентів – діаграма UML, яка демонструє як компоненти системи розташовані та пов’язані один з одним. Вона використовується для

моделювання програмних систем на високому рівні або для відображення компонентів на нижчому рівні [9].

Цей тип діаграми підтримує компонентну розробку, в якій система поділена на компоненти та інтерфейси, які можна використовувати повторно та замінювати.

На рисунку 2.5 представлена діаграма компонентів.

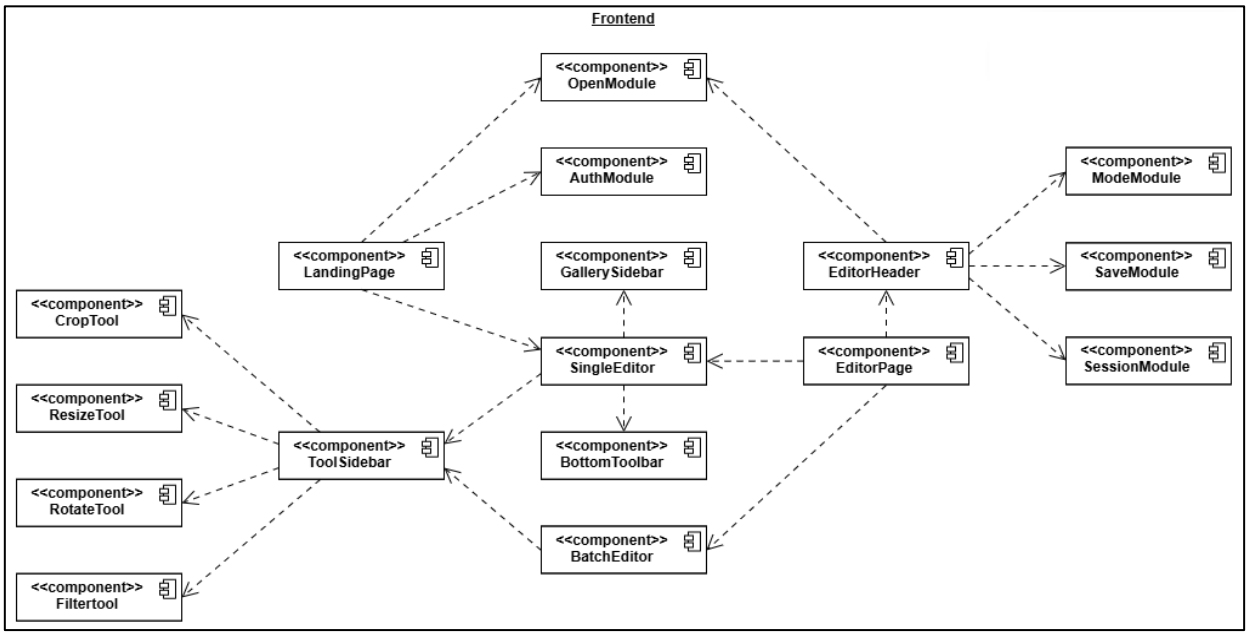


Рисунок 2.5 – Діаграма компонентів

На діаграмі компонентів зображено архітектуру клієнтської частини системи. Діаграма ілюструє фізичну структуру інтерфейсу користувача, поділену на логічні модулі, які відповідають за окремі функціональні частини застосунку. Штриховані стрілки між компонентами позначають залежність, яка вказує на те, що один компонент викликає або взаємодіє з іншим.

Компонент LandingPage відповідає за початкову взаємодію з неавторизованим користувачем. Він ініціює відкриття форм для реєстрації та авторизації, взаємодію з модулем для завантаження зображень, а також має зв’язок з редактором для одиночного режиму.

Компонент EditorPage – це головний контейнер для середовища редагування зображень. Він об’єднує у собі компоненти редакторів для обох режимів та панель управління.

Компонент EditorHeader містить основні елементи керування редактором: відкриття та збереження зображень, перемикач режимів редагування та модуль керування сесією.

Компоненти SingleEditor та BatchEditor реалізують редактори для одного або декількох файлів відповідно. Кожен із них безпосередньо взаємодію з панеллю інструментів. SingleEditor має додаткові два компонента які відповідають за галерею зображень, та панеллю керування зображення на сцені. Перемикання між цими редакторами координується через головний контейнер.

Компонент ToolSidebar – це контейнер для інструментів редагування. Він координує виклики трансформацій, які реалізуються окремими допоміжними модулями. Його завдання – передати параметри до відповідного інструмента та відобразити відповідний функціонал у межах основного редактора.

2.6 Діаграма розгортання

Діаграма розгортання – це модель фізичної архітектури системи, яка показує зв’язки між програмними та апаратними компонентами системи та фізичний розподіл обробки. Діаграми розгортання важливі для візуалізації, специфікації та документування вбудованих, клієнт-серверних та розподілених систем [9].

На рисунку 2.6 представлена діаграма розгортання. На діаграмі зображено фізичну архітектуру, на якій функціонує вебзастосунок. Система побудована за допомоги клієнт-серверної архітектури з інтеграцією зовнішнього хмарного сервісу. Всі взаємодії між вузлами відбуваються через захищені протоколи: HTTPS та TCP/IP.

Початковою точку взаємодії є користувач, який працює через веббраузер. Інтерфейс користувача реалізовано як React-застосунок, який розгорнутий на Frontend сервері.

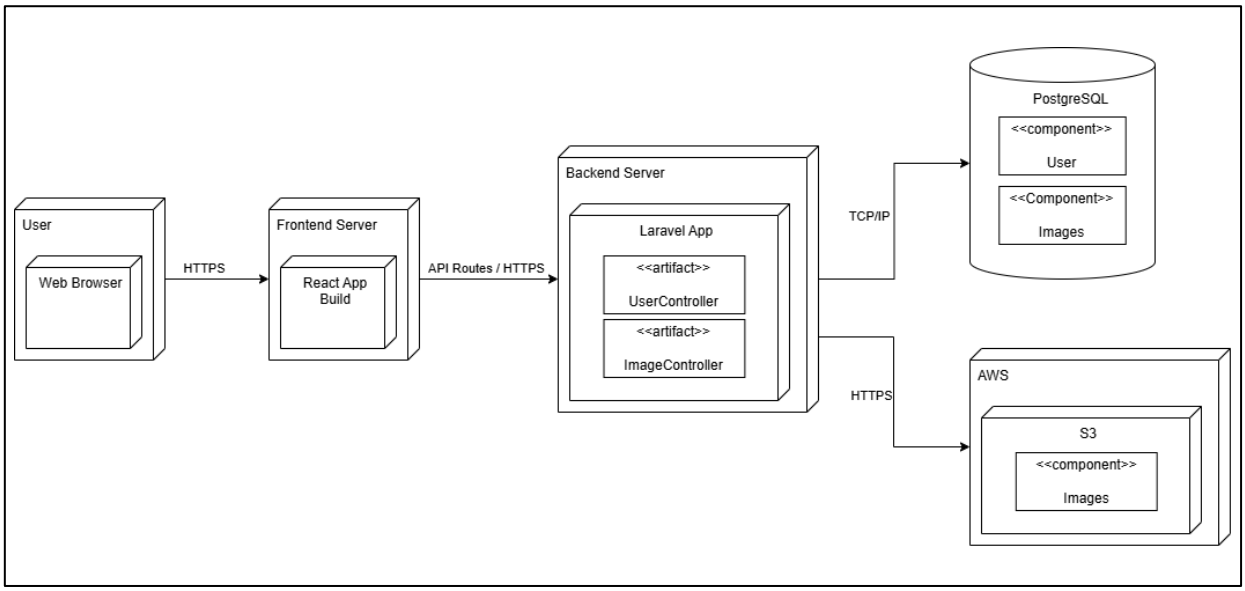


Рисунок 2.6 – Діаграма розгортання

Frontend надсилає запити до Backend, де розгорнуто Laravel-застосунок. Усередині нього визначено два основних контролери: UserController – для управління користувачами, та ImageController – для обробки зображень.

Backend виконує дві основні задачі: взаємодія з базою даних і роботу з хмарним сховищем. Для зберігання даних використовується PostgreSQL, яка зберігає таблиці User та Images.

Для зберігання самих зображень використовується Amazon S3. Контролер ImageController завантажує файли туди або створює тимчасові посилання для їх отримання клієнтом.

2.7 Проєктування інтерфейсу

Мокап – це графічне представлення майбутнього інтерфейсу, яке містить основні блоки, кнопки, поля введення та інші елементи взаємодії. Він дозволяє створити статичну візуальну модель сторінки, показати її структуру, визначити ієрархію та логіку переходів між елементами.

На рисунках 2.7-2.9 продемонстровані мокапи початкової сторінки, для редактора у одиночному та пакетному режимі відповідно.

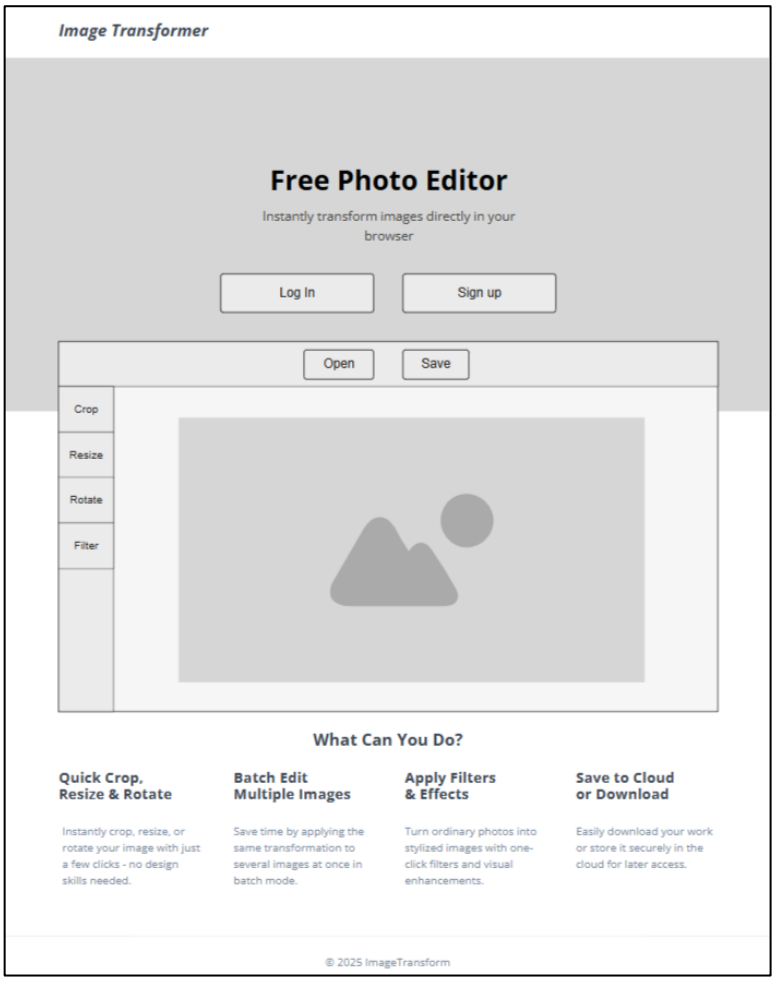


Рисунок 2.7 – Початкова сторінка застосунку

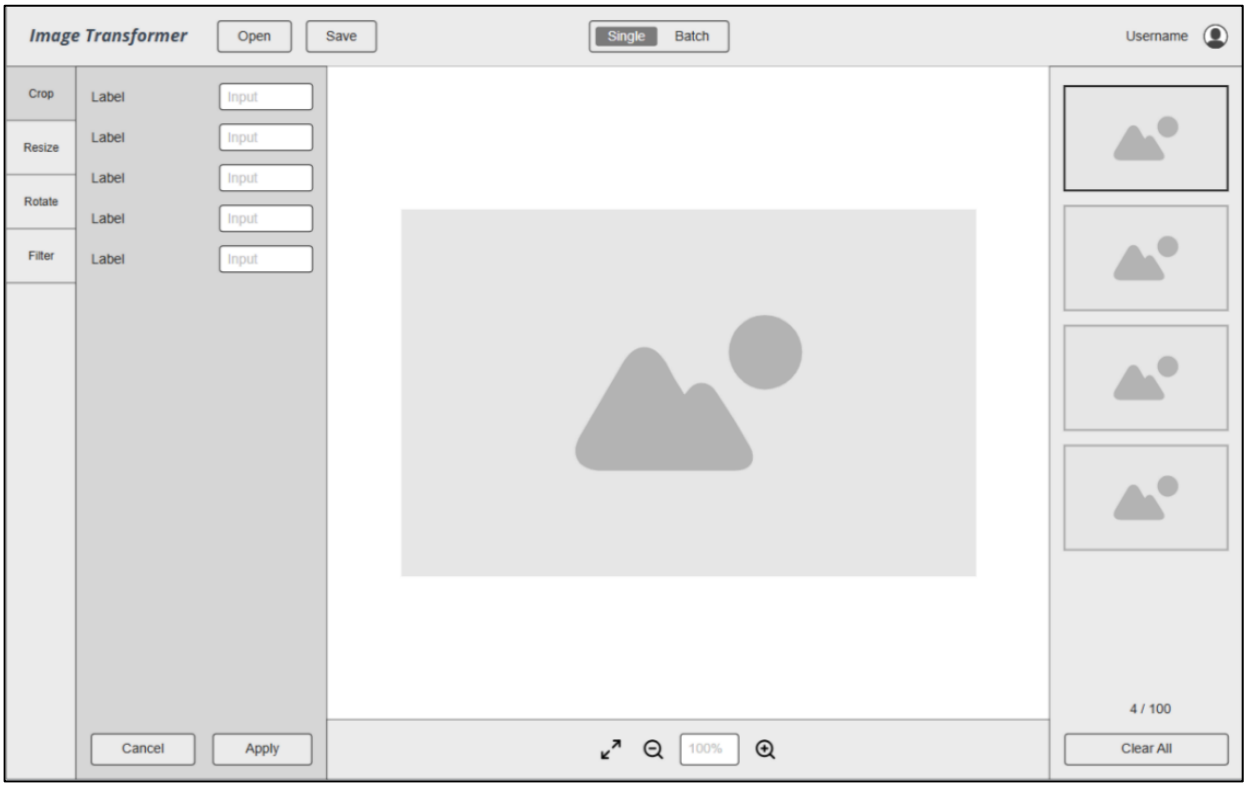


Рисунок 2.8 – Редактор зображень у одиночному режимі

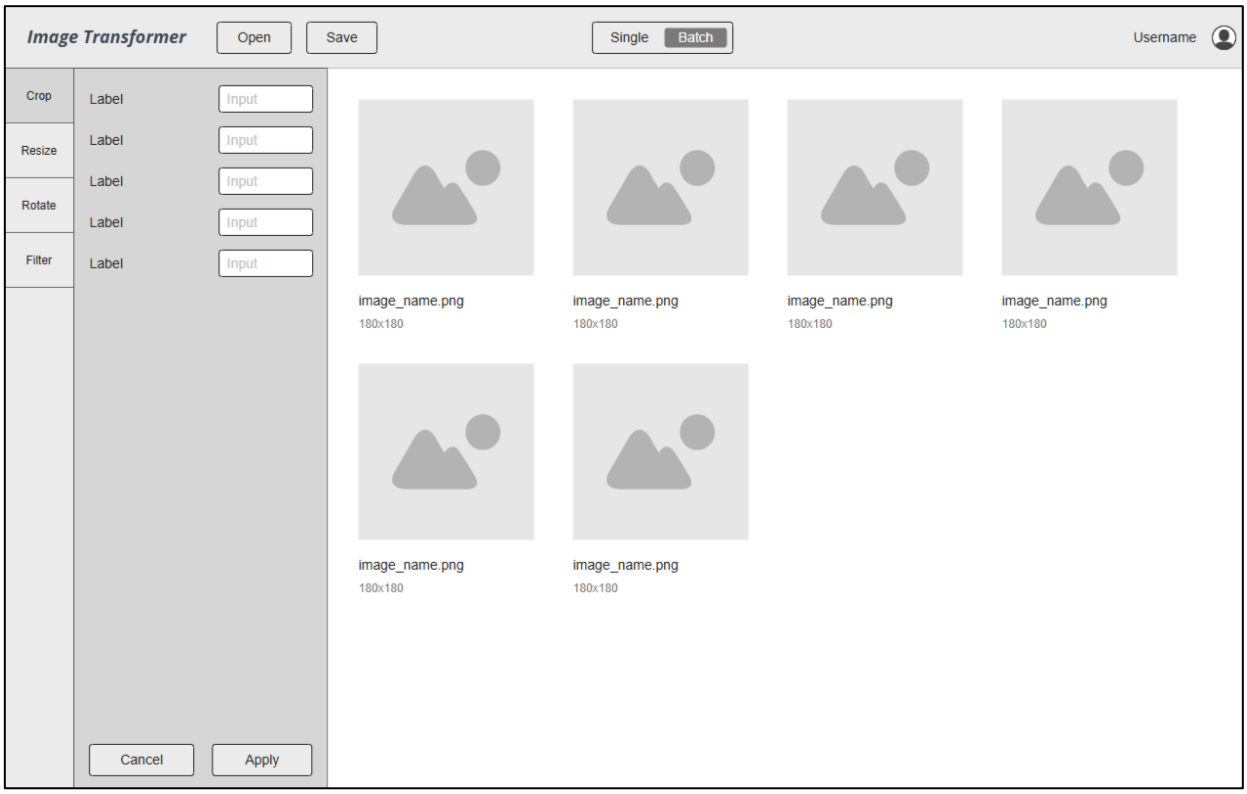


Рисунок 2.9 – Редактор зображень у пакетному режимі

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

React – це JavaScript-бібліотека для створення інтерфейсів користувача. Її основна філософія полягає в побудові користувацького інтерфейсу з незалежних, повторно використовуваних компонентів. React використовує концепцію віртуального DOM, завдяки цьому забезпечується швидкий та ефективний рендеринг змін у інтерфейсі [10].

Konva – це HTML5 Canvas JavaScript фреймворк, який розширює двовимірний контекст, забезпечуючи інтерактивність полотна для десктопних і мобільних застосунків. Він дозволяє створювати високопродуктивну анімацію, переходи, вкладеність вузлів, шарування, фільтрацію, кешування, обробку подій та багато іншого. Для інтеграції з React існує обгортка react-konva, яка забезпечує декларативний підхід до побудови Canvas-графіки, дозволяючи використовувати компоненти React для створення та управління графічними елементами [11].

Tailwind CSS – це утилітарний CSS фреймворк, який надає набір класів для розробки вебінтерфейсів без написання кастомного CSS. Він пропонує малі, утилітарні класи, які можна комбінувати для створення будь-якого дизайну безпосередньо в HTML розмітці [12].

FlyonUI – це бібліотека компонентів Tailwind CSS із відкритим кодом із семантичними класами та JavaScript плагінами [13].

PostCSS – це інструмент для трансформації CSS за допомогою JavaScript плагінів. Основний робочий процес PostCSS включає токенізацію CSS рядка, побудову абстрактного синтаксичного дерева (AST), обробку цього дерева плагінами та генерацію фінального CSS [14].

Lucide – це бібліотека SVG-іконок з відкритим кодом, яка надає понад 1000 векторних іконок. Lucide має інтеграцію з React [15].

Vite – це інструмент для збірки вебзастосунків. Основна мета Vite – забезпечити швидку збірку проєкту та швидке оновлення під час розробки завдяки використання нативних ES-модулів та попередньому бандлінгу залежностей через Rollup [16].

Laravel – це безкоштовний PHP-фреймворк з відкритим кодом, створений для розробки вебзастосунків з акцентом на шаблоні MVC (Model-View-Controller), що дозволяє чітко розділити логіку застосунку, інтерфейс користувача та взаємодію з базою даних [17].

PostgreSQL – це об’єктно-реляційна система керування базами даних з відкритим кодом, яка відзначається високою надійністю, масштабованістю та широкими функціональними можливостями. Вона підтримує розширені типи даних, складні запити, зовнішні ключі, тригери та представлення, а також процедурні мови для збереження процедур [18].

3.2 Серверна частина

У системі розроблені дві моделі: User та Image. Перша відповідає за облікові записи користувачів і включає JWT-автентифікацію. Друга зберігає метадані зображень – шлях до файлу, розмір, формат, а також зв’язується з користувачем через відношення «один до багатьох».

Кожна модель має власний контролер. AuthController обробляє реєстрацію, вхід, вихід та оновлення токенів користувача. ImageController відповідає за логіку збереження, оновлення, видалення та отримання зображення.

Хмарне сховище Amazon S3 інтегровано як основний механізм збереження зображень користувачів. Після завантаження файлів Laravel використовує драйвер S3 для збереження у відповідному бакеті. Якщо зберегти у хмару не вдається – зображення зберігається у публічній теці на сервері.

API реалізовано за принципами REST, з авторизацією через JWT-токени, які передаються у вигляді HTTP cookie.

На рисунку 3.1 продемонстровано шляхи API для автентифікації користувача.

```
Route::prefix('auth')->group(function () {
    Route::post('/register', [AuthController::class, 'register']);
    Route::post('/login', [AuthController::class, 'login']);
    Route::middleware([InjectJwtFromCookie::class, 'api'])->group(function () {
        Route::post('/me', [AuthController::class, 'me']);
        Route::post('/logout', [AuthController::class, 'logout']);
        Route::post('/refresh', [AuthController::class, 'refresh']);
    });
});
```

Рисунок 3.1 – Реалізація маршрутів для автентифікації користувача

3.3 Клієнтська частина

3.3.1 Компоненти автентифікації

У застосунку реалізовано два модальні компоненти для автентифікації користувача: SignUpModal та LogInModal. Вони працюють у зв’язці з контекстом UserContext, де міститься логіка запитів до серверу.

Компонент SignUpModal вимагає від користувача ввести ім’я, електронну пошту, пароль та його підтвердження.

Компонент LogInModal включає тільки поля для електронної пошти та пароля.

Обидва компоненти виводяться з UserPanel, який змінює свій вміст у залежності від того, чи користувач увійшов до акаунту.

3.3.2 Компоненти редактора

Логіка інтерфейсу для редактора реалізована у компоненті CanvasEditor, який виводить різні варіанти інтерфейсу в залежності від режиму редагування: SingleCanvas, для одиночної обробки, та BatchCanvas, для пакетного редагування. Дані про зображення у системі керуються контекстом ImagesManagerContext.

Компонент RightSidebar – це вертикальна панель з мініатюрами всіх завантажених зображень, яка дозволяє керувати зображеннями у поточній сесії. Компонент BottomToolbar містить елемент керування масштабуванням, який дозволяє користувачу змінювати візуальний розмір зображення на сцені.

Компонент OpenMenu надає користувачу можливість вибору джерела завантаження зображень. Для локального завантаження використовується прихований компонент FileInput, який читає файл за допомогою FileReader та додає його до контексту ImagesManagerContext.

Для завантаження з хмари використовується компонент CloudModal, який завантажує зображення через API з сервера. CloudModal надає можливість користувачу керувати його збереженими зображеннями: додавати до поточної сесії або видалити зі сховища.

Функціонал збереження реалізовано за допомоги компонента SaveMenu. Користувач може зберегти зображення на локальному диску або завантажити його у хмару. В залежності від режиму редагування відкривається компонент SaveModal або BatchSaveModal. У одиночному режимі можна обрати назву, формат, якість зображення і шлях збереження. У пакетному режимі реалізовано збереження архіву у форматі ZIP з усіма зображеннями.

3.3.3 Компоненти трансформації

Редагування зображень у застосунку реалізовано з використанням бібліотеки Konva.js. У режимах одиночного та пакетного редагування, кожне

зображення виводиться у вигляді Konva.Stage, всередині якого знаходяться шари (Konva.Layer) та зображення (Konva.Image).

Ресемплювання зображень – це процес перерахування пікселів згідно нового розміру. Цей процес є обов’язковим для збереження результатів трансформації, оскільки Konva лише візуально змінює зображення на екрані.

Для покращення швидкодії ресемплювання було використано OffscreenCanvas та Web Workers API. Worker обробляє операції в окремому потоці, що дозволяє уникнути блокування інтерфейсу користувача. OffscreenCanvas рендерить зображення без участі DOM, зменшуючи навантаження на потік і графічний процесор. ImageBitmap передається у Worker як transferable-об’єкт без копіювання, тобто об’єкт передається з однієї області пам’яті в іншу. Результат генерується асинхронно у форматі Blob.

Обрізання зображень реалізується через компонент CropTool, який підтримує як довільне, так і пропорційне обрізання.. Для візуалізації області обрізання використовується прямокутник (Konva.Rect), трансформер (Konva.Transformer) і напівпрозора маска (Konva.Shape). Область обрізки може масштабуватись, переміщуватись, і обмежується розмірами зображення.

Масштабування зображень реалізоване в компоненті ResizeTool, Користувач має можливість обрати один із режимів: за пікселями, за відсотками, пропорційно по ширині або висоті.

Компонент RotateTool реалізує обертання зображень за допомоги властивостей Konva.Image. Підтримуються повороти на 90°, а також віддзеркалення по горизонталі та вертикалі.

Накладання фільтрів реалізовано через гнучку систему компонентів FilterTool і внутрішню архітектуру filterMap. Для кожного зображення створюється екземпляр логіки, який керує накладанням фільтрів через Konva.Filters.

Операції накладання фільтрів у Konva є важкими операціями по продуктивності. Тому перед накладанням фільтрів зображення кешується за допомоги вбудованого метода Konva.

3.4 Керівництво користувача

3.4.1 Рівень підготовки користувача

Вміння працювати з персональним комп’ютером і веббраузером.

Ознайомлення з керівництвом користувача.

3.4.2 Підготовка до роботи

Доступ до сайту здійснюється через Інтернет з використанням веббраузера. Адреса сайту: https://imagetransformer.loc. Для стабільної роботи клієнтської частини рекомендовано використовувати Google Chrome, Mozilla Firefox, Microsoft Edge або Safari.

Після переходу на сайт відкривається лендінг сторінка, де доступні авторизація та демонстраційний редактор.

На сайті визначено такі типи користувачів:

– неавторизований користувач;
– авторизований користувач.

3.4.3 Вхід до системи

Після переходу на сайт, система відображає лендінг сторінку системи (див. рис. 3.2).

Якщо Ви не зареєстровані, необхідно натиснути кнопку “Sign Up”, заповнити поля з іменем, електронною поштою, паролем і його підтвердженням, після чого натиснути кнопку “Sign Up” у модальному вікні (див. рис. 3.3).

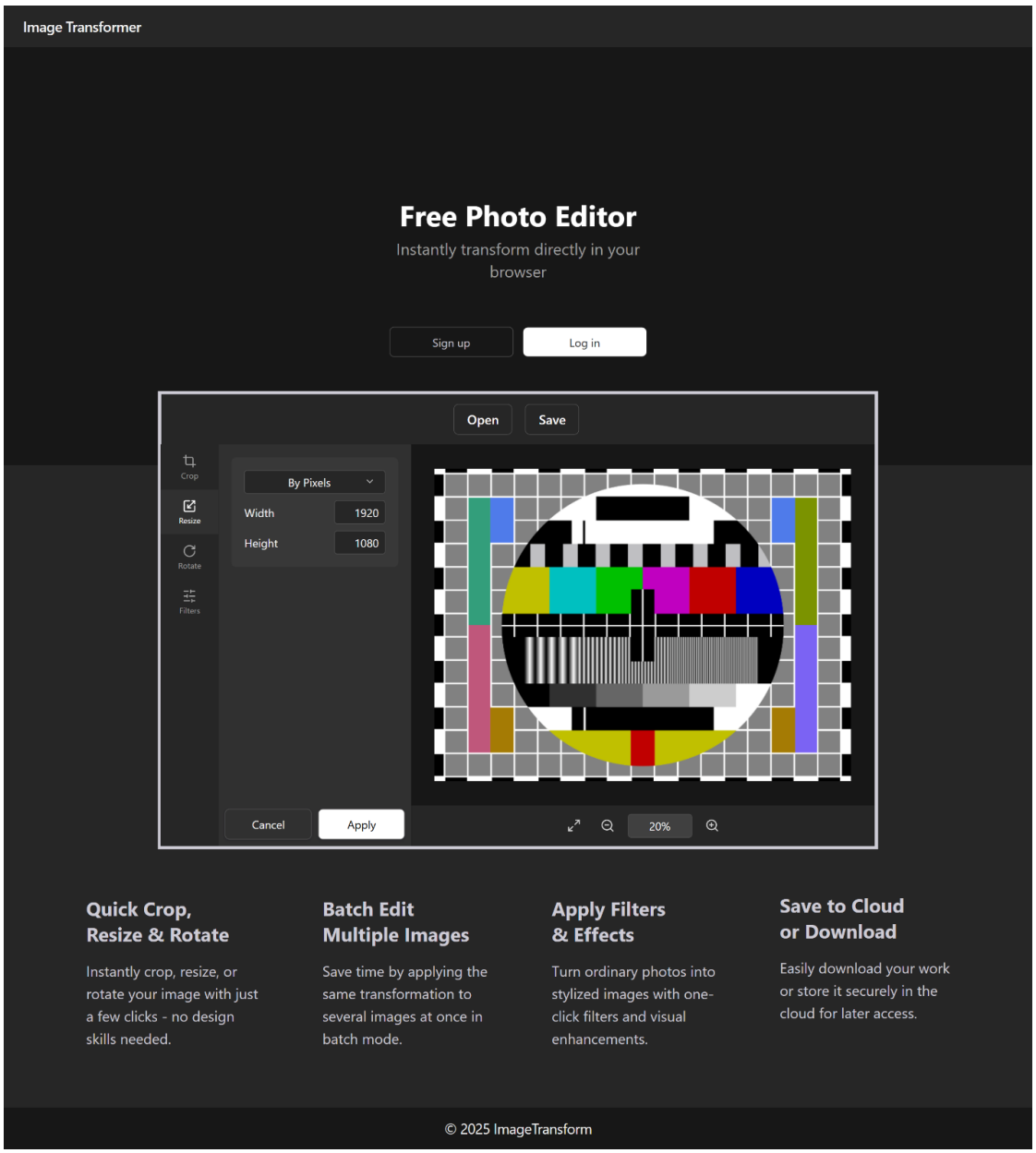


Рисунок 3.2 – Лендінг сторінка системи

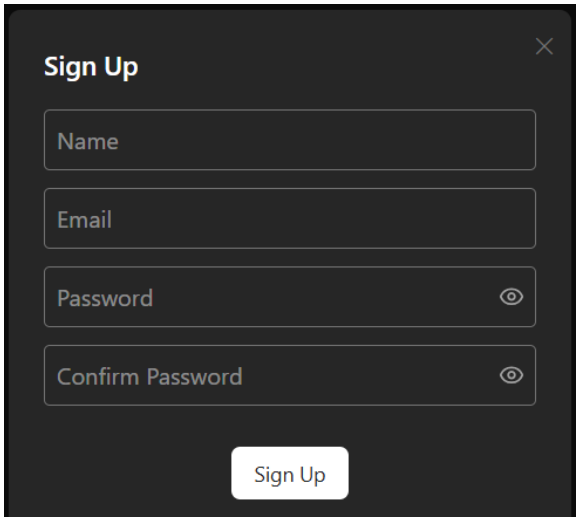


Рисунок 3.3 – Модальне вікно реєстрації

Якщо Ви вже зареєстровані, слід натиснути кнопку “Log In”, ввести електрону пошту та пароль у відповідні поля та натиснути кнопку “Log In” у модальному вікні (див. рис. 3.4). Після авторизації, система відобразить сторінку редактора зображень (див. рис. 3.5).

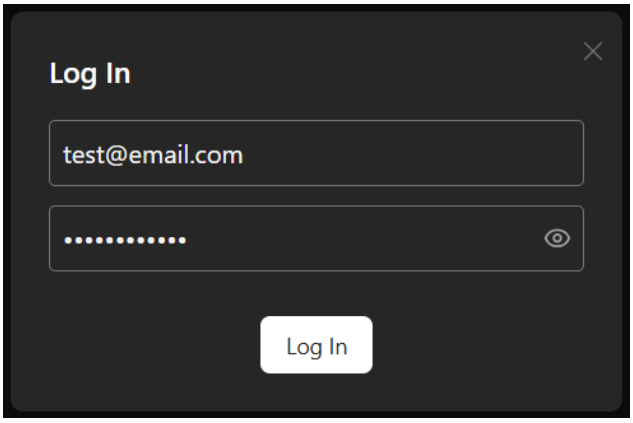


Рисунок 3.4 – Модальне вікно авторизації

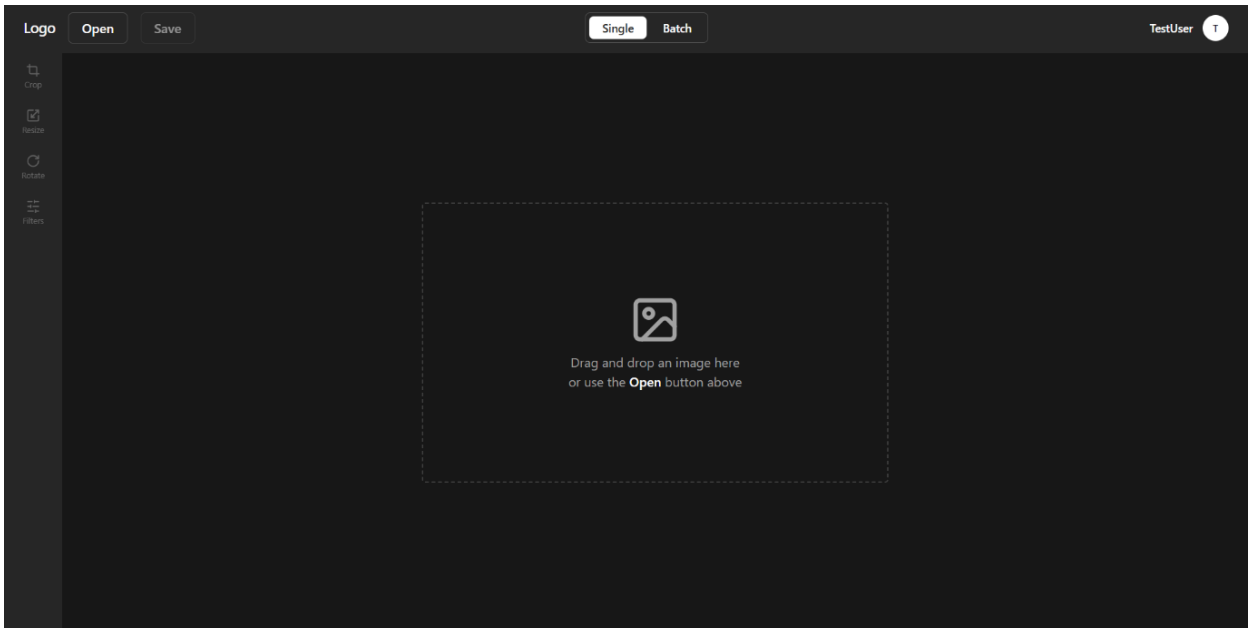


Рисунок 3.5 – Сторінка редактора зображень

3.4.4 Завантаження зображень

Для початку роботи із зображенням його потрібно завантажити до системи. Потрібно натиснути на кнопку “Open” (див. рис. 3.6), та вибрати

режим завантаження: з комп’ютера (“From Computer”) або з хмарного сховища (“From Cloud”).

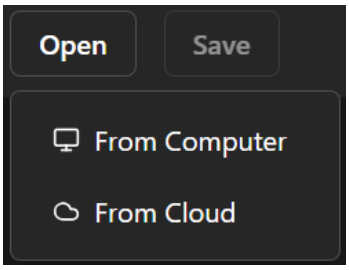


Рисунок 3.6 – Меню вибору режиму завантаження зображення

Якщо Ви обрали режим “From Computer”, то відкриється стандартне вікно вибору файлів.

Якщо Ви обрали режим “From Cloud”, то відкриється модальне вікно (див. рис. 3.7), у якому потрібно вибрати зображення, та в кінці натиснути кнопку “Select”.



Рисунок 3.7 – Модальне вікно хмарного сховища

3.4.5 Перемикання режимів редагування

Перемикання режимів редагування відбувається за допомоги перемикача у верхні панелі редактора. Після завантаження зображення

система автоматично активує режим одиночного редагування (див. рис. 3.8), що позначається як “Single”.

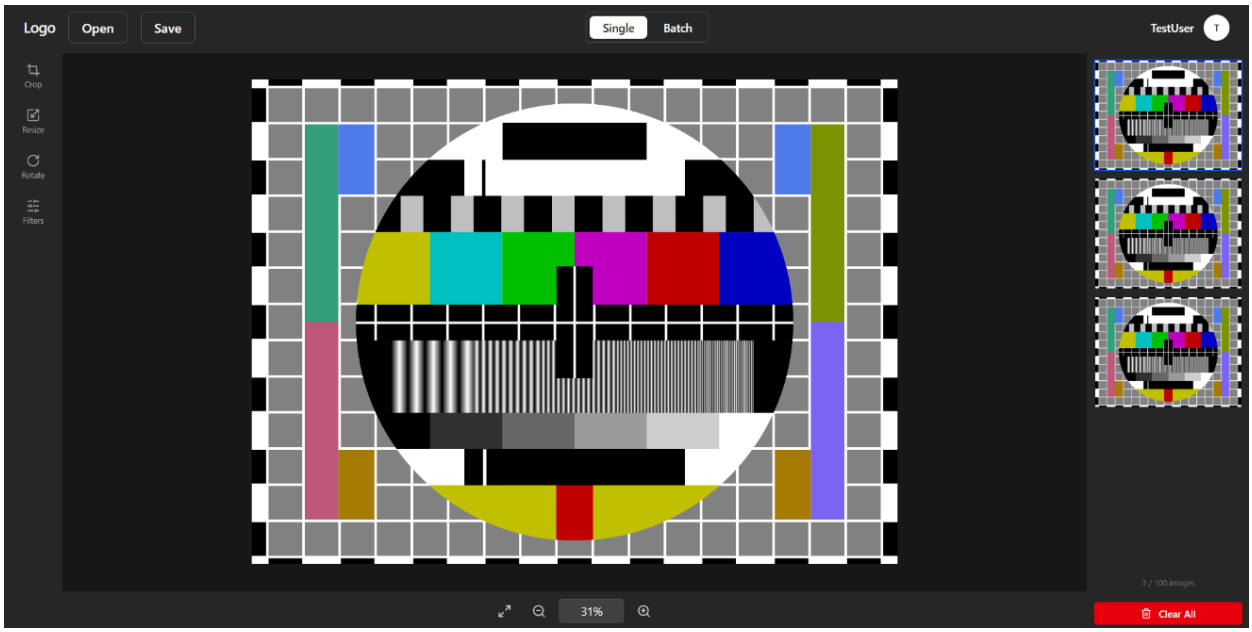


Рисунок 3.8 – Редактор у одиночному режимі редагування

Для перемикання у режим пакетного редагування (див. рис. 3.9) потрібно натиснути на кнопку “Batch”.



Рисунок 3.9 – Редактор у пакетному режимі редагування

3.4.6 Виконання трансформацій

Для застосування трансформацій потрібно натиснути на інструмент редагування у лівій панелі. Далі необхідно ввести налаштування

трансформації та натиснути кнопку “Apply”. Для того щоб відмінити редагування, потрібно натиснути кнопку “Cancel”. На рисунку 3.10 продемонстрована робота інструмента для обрізання зображення.

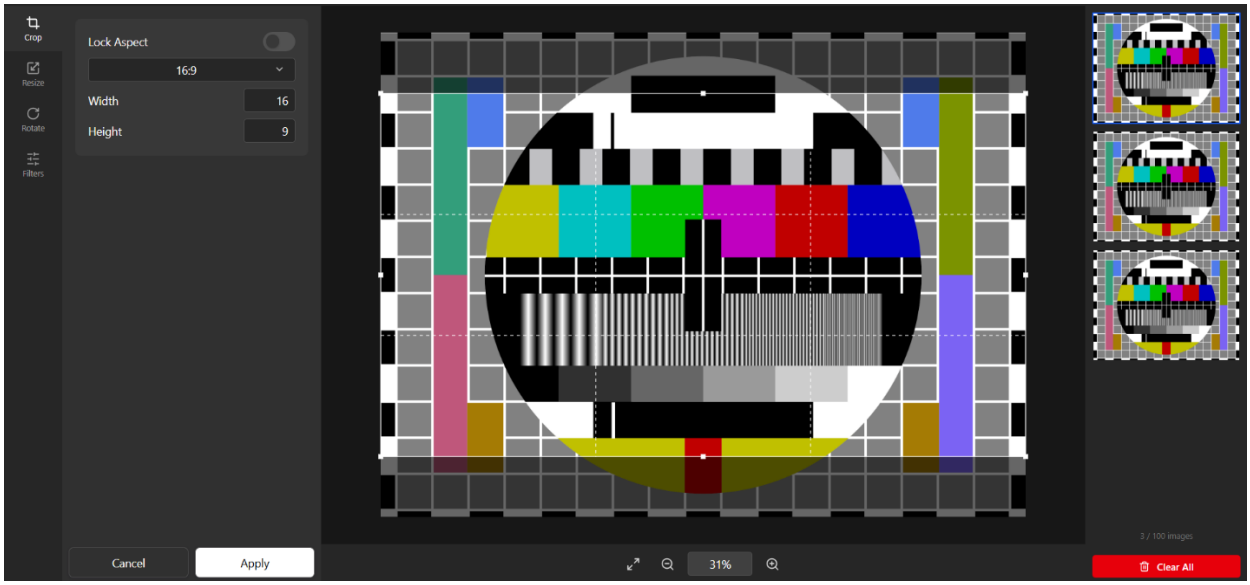


Рисунок 3.10 – Інструмент обрізання

3.4.7 Збереження результатів

Щоб зберегти результат роботи потрібно натиснути кнопку “Save” (див. рис. 3.11), яка знаходиться у верхній панелі редактора.

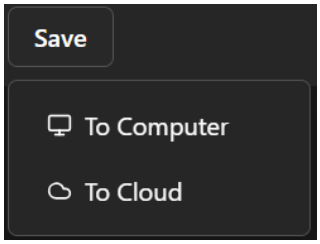


Рисунок 3.11 – Меню вибору режиму збереження зображення

Доступно два режими збереження: до комп’ютера (“To Computer”) або до хмарного сховища (“To Cloud”).

Якщо Ви зберігаєте у одиночному режимі редагування, то відкриється модальне вікно (див. рис. 3.12), у якому потрібно вказати назву, обрати формат

та якість. Після цього потрібно натиснути кнопку “Download”. Далі відкриється стандартне вікно для збереження файлів, де можна вказати місце збереження.

Якщо Ви зберігаєте у пакетному режимі редагування, то відкриється модальне вікно, у якому потрібно обрати формат та якість. Після цього потрібно натиснути кнопку “Download”. Далі відкриється стандартне вікно для збереження файлів, де можна вказати місце збереження.

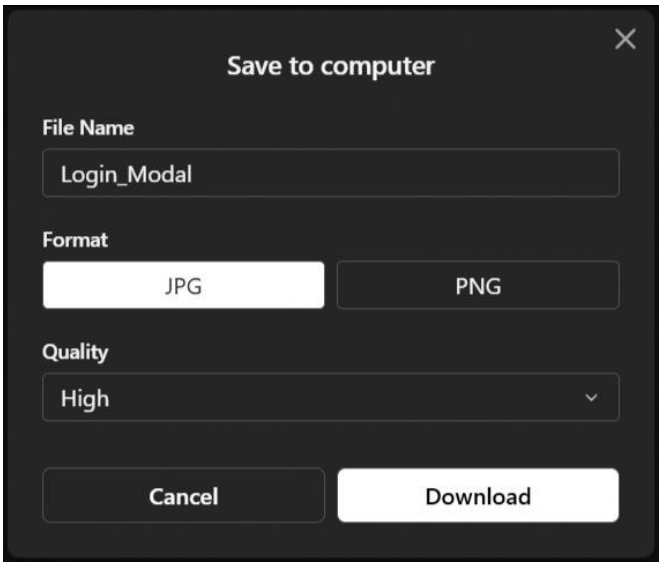


Рисунок 3.12 – Меню збереження до комп’ютера

У випадку збереження у хмару (див. рис. 3.13) стандартне вікно для збереження не відкривається, після натискання кнопки “Download”. Замість нього, якщо збереження успішне, буде виведено відповідне повідомлення.

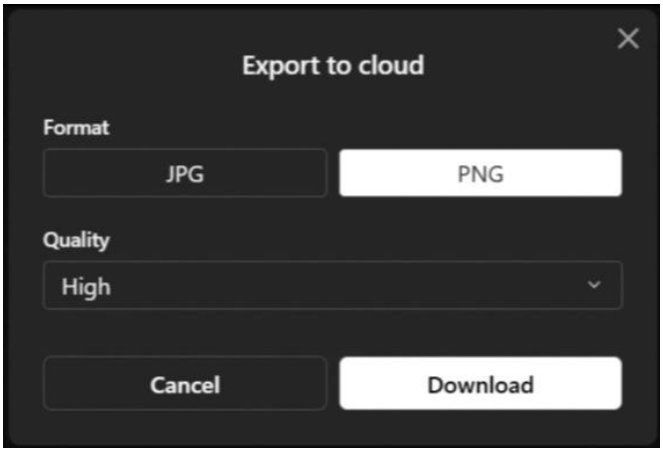


Рисунок 3.13 – Меню збереження у хмарне сховище

3.5 Тестування системи

Для розробленою системи було проведено тестування. Встановлено, що система коректно виконує функції через вебінтерфейс. Вона підтримує трансформацію як у одиночному, так і пакетному режимі, забезпечуючи стабільну взаємодію між клієнтською та серверною частинами системи.

Результати тестування підтверджують відповідність реалізованого функціонала технічними вимогам, які ставились на початку розробки. Система ефективно виконує онлайн трансформацію зображень.

ВИСНОВКИ

У ході кваліфікаційної роботи було розроблено сервіс онлайн трансформації зображень. Поставлена задача передбачала аналіз предметної області проєктування архітектури, реалізацію серверної та клієнтської частини системи.

У першому розділі роботи проведено аналіз предметної області. На основі цього аналізу сформовано технічні вимоги до майбутнього застосунку. Другий розділ присвячено проєктування архітектури застосунку за допомоги UML діаграм. У третьому розділі описано технологічний стек, детально розглянуто реалізовану функціональність.

Поставлені завдання було реалізовано в повному обсязі. Вебзастосунок повністю функціонує, відповідає сучасним вимогам та включає всі основні функції для одиночної та пакетної трансформації зображень. Під час реалізації вдалося не лише досягти цілей, але й забезпечити технічний фундамент для майбутнього розширення функціоналу без повної переробки архітектури.

У майбутньому функціональність застосунку можна значно розширити. Серед перспектив – реалізація історії змін, додавання шарів, підтримка векторної графіки, інтеграція генеративного штучного інтелекту для автоматичної фільтрації або покращення зображень. Розширення на мобільні платформи та PWA-версія.

Під час розробки даного сервісу було отримано цінні практичні навички роботи з повним стеком сучасних вебтехнологій: Laravel, React та Konva. Це дозволило поглибити знання як у серверній, так і в клієнтській частині веброзробки, а також навчитися ефективно організовувати командну логіку та графічні трансформації у вебзастосунках.

ПЕРЕЛІК ПОСИЛАНЬ

1. Megapixel. *Cloudinary*. URL: https://cloudinary.com/glossary/megapixel (дата звернення: 01.04.2025).

2. Solomon C., Breckon T. Fundamentals of digital image processing. Hoboken : Wiley-Blackwell, 2011. 352 p.

3. What is a Web Application? *Amazon Web Services, Inc*. URL: https://aws.amazon.com/what-is/web-application/ (дата звернення: 01.04.2025).

4. About us. *Pixlr*. URL: https://pixlr.com/about-us/ (дата звернення: 07.04.2025).

5. About us. *Fotor*. URL: https://www.fotor.com/aboutus.html/ (дата звернення: 03.04.2025).

6. About us. *Picsart*. URL: https://picsart.com/about-us/ (дата звернення: 03.04.2025).

7. Fowler M. UML distilled : 3rd ed. Boston : Addison-Wesley Professional, 2003. 208 p.

8. Hamilton K., Miles R. Learning UML 2.0. Sebastopol : O'Reilly, 2006. 286 p.

9. Larman C. Applying UML and patterns : 3rd ed. Hoboken, NJ : Prentice Hall, 2004. 736 p.

10. Banks A., Porcello E. Learning react: modern patterns for developing react apps : 2nd ed. Sebastopol : O'Reilly, 2020. 310 p.

11. Starting with Konva. *Konva*. URL: https://konvajs.org/docs/index.html (дата звернення: 28.04.2025).

12. Tailwind CSS. *Tailwind CSS*. URL: https://tailwindcss.com/ (дата звернення: 28.04.2025).

13. FlyonUI. *FlyonUI*. URL: https://flyonui.com/ (дата звернення: 28.04.2025).

14. PostCSS. *PostCSS*. URL: https://postcss.org/ (дата звернення: 28.04.2025).

15. Lucide Icons. *Lucide*. URL: https://lucide.dev/ (дата звернення: 28.04.2025).

16. Getting Started. *Vite*. URL: https://vite.dev/guide/ (дата звернення: 28.04.2025).

17. Installation. *Laravel*. URL: https://laravel.com/docs/12.x (дата звернення: 28.04.2025).

18. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (дата звернення: 28.04.2025).