

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

**«Вебплатформа робочого місця фотографа з
інтеграцією ШІ-функцій для обробки контенту»**

Виконала студентка групи ВТ-22-2
ПІВОВАРОВА Дар'я Валеріївна

Керівник роботи:
ЧИЖМОТРИЯ Олексій Володимирович

Рецензент:
ВЛАСЕНКО Олег Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«27» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Пивоварова Дар'я Валеріївна**

Керівник роботи: **Чижмотря Олексій Володимирович**

Тема роботи: **«Вебплатформа робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **Об'єктом дослідження є процеси організації та автоматизації професійної діяльності фотографа під час роботи з цифровим контентом.**

Предметом дослідження є використання технологій розробки вебплатформ та методи інтеграції штучного інтелекту для автоматизації обробки, аналізу та керування фотографічним контентом.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	27.02.2025р.	27.02.2025р.
2	Чижмотря О. В.	06.03.2025р.	06.03.2025р.
3	Чижмотря О. В.	24.04.2025р.	24.04.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка до випускної кваліфікаційної роботи містить вступ, 3 розділи, висновки, список використаних джерел із 30 найменувань (3 стор.), один додаток (37 стор.), 12 таблиць та 24 ілюстрацій. Повний обсяг роботи становить 113 сторінок, з них 75 сторінок основного тексту.

Метою роботи є створення вебплатформи для автоматизації роботи фотографа, яка забезпечує зручну взаємодію з клієнтами, централізоване керування контентом та інтеграцію сучасних AI-технологій для обробки зображень і генерації матеріалів. Система реалізована у вигляді клієнт-серверного вебзастосунку з використанням фреймворку Laravel 12, бібліотеки React та технології Inertia.js. Для зберігання даних використовується PostgreSQL, а для організації файлового сховища реалізовано інтеграцію з Google Drive API. Також система підтримує інтеграцію з AI-сервісами Groq та Pollinations для генерації текстового та графічного контенту.

У роботі проведено аналіз предметної області та існуючих аналогів, визначено функціональні та нефункціональні вимоги до системи, обґрунтовано обрані архітектурні рішення та технологічний стек. Спроектовано структуру бази даних, розроблено алгоритми роботи ключових модулів, зокрема оформлення замовлень, автоматичної синхронізації файлів із хмарним сховищем, AI-редагування зображень та генерації контенту. Реалізовано клієнтську та серверну частини застосунку, а також проведено тестування основних функціональних можливостей системи.

Ключові слова: LARAVEL, REACT, INERTIA.JS, TYPESCRIPT, PHP, POSTGRESQL, GOOGLE DRIVE API, GROQ, POLLINATIONS, ПОРТФОЛІО.

					ІПЗ.КР.Б – 121 – 26 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа робочого місця фотографа з інтеграцією ШІ- функцій для обробки контенту Пояснювальна записка	Літ.	Арк.	Аркушів	
Розроб.		Пивоварова Д.В.							
Керівник.		Чижмoтря О.В.					3	113	
						Житомирська політехніка, група ВТ-22-2			
Рецензент		Власенко О.В.							
Зав. каф.		Вакалюк Т.А							

ABSTRACT

The bachelor's qualification thesis consists of a software system and an explanatory note. The explanatory note includes an introduction, 3 chapters, conclusions, a list of 30 references (3 pages), one appendix (37 pages), 12 tables, and 24 figures. The total volume of the thesis is 113 pages, including 75 pages of main text. The developed system is designed to automate a photographer's activities and covers client interaction processes, portfolio management, blog administration, photo session order processing, photo retouching, as well as the use of artificial intelligence tools.

The aim of the thesis is to develop a web platform for photographer workflow automation that provides convenient client interaction, centralized content management, and integration of modern AI technologies for image processing and content generation. The system is implemented as a client-server web application using the Laravel 12 framework, the React library, and Inertia.js technology. PostgreSQL is used as the database management system, while Google Drive API integration is implemented for file storage management. The platform also supports integration with Groq and Pollinations AI services for generating textual and graphical content.

The thesis includes an analysis of the problem domain and existing solutions, defines the functional and non-functional requirements of the system, and justifies the selected architectural approaches and technology stack. A database structure was designed, and algorithms for the key system modules were developed, including order processing, automatic synchronization of files with cloud storage, AI-powered image editing, and content generation. Both the client-side and server-side components of the application were implemented, and the main functional capabilities of the system were tested.

Keywords: LARAVEL, REACT, INERTIA.JS, TYPESCRIPT, PHP, POSTGRESQL, GOOGLE DRIVE API, GROQ, POLLINATIONS, PORTFOLIO.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	12
1.3. Архітектура системи.....	20
1.4. Обґрунтування вибору засобів реалізації.....	24
Висновки до першого розділу.....	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	30
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	30
2.2. Розробка бази даних системи.....	38
2.3. Проєктування та реалізація алгоритмів роботи системи	47
Висновки до другого розділу	54
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ	56
3.1. Порядок встановлення та налаштування параметрів системи	56
3.2. Структура інтерфейсу.....	59
3.3. Тестування роботи програмного комплексу	68
Висновки до третього розділу	70
ВИСНОВКИ.....	72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	73
ДОДАТКИ.....	76

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API - Application Programming Interface, Інтерфейс прикладного програмування

PHP - Hypertext Preprocessor, мова програмування для веброзробки

SEO - Search Engine Optimization, пошукова оптимізація

JSON - JavaScript Object Notation, формат обміну даними

БД - База даних

AI - штучний інтелект

СУБД - Система управління базами даних

UI - користувацький інтерфейс.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми

Станом на 2026 рік цифрова присутність стала невід’ємною складовою для професійної діяльності фотографа та важливим чинником його успішної конкурентноздатності на ринку. В умовах стрімкого розвитку цифрових технологій, більшість процесів взаємодії з клієнтами переходить у мережу Інтернет. Водночас рівень конкуренції у сфері фотографії постійно зростає, що зумовлює необхідність пошуку ефективних способів самопрезентації та вирізнення серед інших фахівців.

Потенційні клієнти дедалі частіше здійснюють пошук фотографів саме за допомогою онлайн-ресурсів, орієнтуючись на візуальне наповнення, стиль виконання робіт, відгуки та загальну репутацію спеціаліста. За таких умов наявність власної вебплатформи у вигляді портфоліо-блогу перестає бути лише додатковою перевагою та перетворюється на необхідний інструмент здійснення професійної діяльності. Така платформа забезпечує можливість систематизованої презентації робіт, сприяє формуванню та просуванню особистого бренду фотографа, а також створює ефективний канал комунікації з цільовою аудиторією.

Сучасна вебплатформа для фотографа виконує функцію не лише онлайн-галереї, а й повноцінного інструменту керування контентом та клієнтами. Вона дозволяє структуровано розміщувати фотопроєкти за категоріями та жанрами, публікувати авторські статті, ділитися досвідом і новинами, а також забезпечувати зручну навігацію для відвідувачів. Наявність блогу сприяє підвищенню впізнаваності автора, покращує пошукову оптимізацію та створює додаткові можливості для залучення нової аудиторії.

Крім презентаційної функції, вебплатформа може містити інструменти зворотного зв’язку, форми для замовлення фотосесій, інтеграцію із соціальними мережами та засоби базової аналітики відвідуваності. Це дозволяє фотографу оцінювати зацікавленість користувачів, аналізувати популярність окремих проєктів і приймати обґрунтовані рішення щодо подальшого розвитку творчої діяльності.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Такий підхід забезпечує системність у веденні професійної діяльності та сприяє підвищенню конкурентоспроможності на ринку послуг.

Метою випускної кваліфікаційної роботи бакалавра є визначення особливостей проєктування та функціонування вебплатформи робочого місця для фотографа з інтеграцією ШІ-функцій для обробки контенту.

Для реалізації поставленої мети потрібно виконати наступні **завдання**:

- Аналіз теоретичних підходів до розробки вебзастосунків і систем керування контентом.
- Дослідження предметної області та формування вимог до функціоналу платформи.
- Моделювання основних функцій і користувацьких сценаріїв.
- Проєктування інформаційної структури та бази даних системи.

Об'єктом дослідження є процеси організації та автоматизації професійної діяльності фотографа під час роботи з цифровим контентом.

Предметом дослідження є технології розробки вебплатформ та методи інтеграції штучного інтелекту для автоматизації обробки, аналізу та керування фотографічним контентом.

Методом дослідження є комплексне використання методів системного аналізу, об'єктно-орієнтованого проєктування, вебпрограмування та UI/UX дизайну, а також методів інтеграції алгоритмів штучного інтелекту для обробки зображень.

Практичним значенням отриманих результатів є можливість безпосереднього впровадження розробленої вебплатформи у професійну діяльність фотографів, фотостудій та креативних агентств. Реалізація ШІ-функцій дозволить автоматизувати рутинні процеси обробки та сортування цифрового контенту, що значно підвищить ефективність роботи користувачів, скоротить час на виконання замовлень та оптимізує керування фотоархівами.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1. Постановка задачі

Програмна система «Вебплатформа робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту» розробляється як комплексне рішення для представлення творчих робіт, публікації тематичних матеріалів, просування особистого бренду та організації ефективної взаємодії з клієнтами. Основною метою системи є створення єдиного цифрового середовища, яке поєднує функції портфоліо, блогу та комунікаційної платформи з клієнтами, з акцентом на SEO-просування та залучення нової аудиторії через пошукові системи.

Система повинна забезпечувати зручну демонстрацію портфоліо з можливістю структурування робіт за категоріями (жанр, тип зйомки, тематика проєкту), детального перегляду фотосерій та опису виконаних робіт. Особлива увага приділяється візуальній привабливості, швидкості завантаження сторінок, адаптивності інтерфейсу та технічній оптимізації для досягнення високих показників у пошуковій видачі.

Важливим компонентом платформи є блог на фотографську тематику, який виконує не лише інформаційну, а й маркетингову функцію. Публікація статей, оглядів техніки, порад зі зйомки та обробки фото сприяє формуванню експертного іміджу автора та покращує органічний трафік завдяки SEO-оптимізації контенту. В адміністративній панелі передбачається система створення статей із використанням різних API побудованих на AI моделях, що дозволяє генерувати ідеї, структуру матеріалів та чернетки текстів із подальшим редагуванням[2]. Це значно пришвидшує процес наповнення сайту та підтримку регулярності публікацій.

Окремим функціональним елементом є контактна форма із клієнтами, яка забезпечує оперативний зворотній зв'язок. Через форму користувачі можуть узгоджувати проведення фотосесій, обговорювати деталі співпраці, а також

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

надсилати фотографії для професійної обробки (ретуші, кольорокорекції, художньої обробки у графічних редакторах). Таким чином платформа підтримує як послуги зйомки, так і дистанційні послуги з редагування фото.

Для досягнення поставленої мети система повинна забезпечувати ефективну взаємодію з користувачами та сприяти просуванню вебресурсу в мережі Інтернет. Одним із ключових завдань є створення адаптивного дизайну, який забезпечує коректне відображення сайту на різних типах пристроїв. При цьому основний акцент робиться на візуальному контенті, оскільки саме він є визначальним для сприйняття робіт фотографа та формування першого враження користувача.

Важливим аспектом є побудова логічної структури сайту та реалізація зручної навігації. Чітка організація розділів і зрозумілий інтерфейс дозволяють користувачам швидко знаходити необхідну інформацію, що позитивно впливає на тривалість перебування на сайті та рівень конверсії.

Окрім цього, система повинна передбачати впровадження інструментів SEO-оптимізації. До них належать налаштування мета-тегів, використання Open Graph для коректного відображення сторінок у соціальних мережах, застосування мікророзмітки, оптимізація швидкості завантаження сторінок, створення sitemap, а також формування зрозумілих і зручних URL-адрес. Використання цих підходів забезпечує кращу індексацію ресурсу пошуковими системами та підвищує його видимість у результатах пошуку.

В умовах сучасного ринку особливо важливо орієнтувати систему на залучення органічного трафіку, що досягається за рахунок покращення позицій сайту у пошукових видачах. Для контролю та оцінки ефективності реалізованих рішень доцільно використовувати інструменти аналітики, які дозволяють відстежувати поведінку користувачів, аналізувати джерела трафіку та приймати обґрунтовані рішення щодо подальшого вдосконалення сайту.

Звичайно система повинна забезпечувати зручне адміністрування та керування контентом. Зокрема, необхідно реалізувати можливість управління портфоліо, що включає додавання, редагування та групування робіт для їх

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

структурованого представлення. Це дозволяє підтримувати актуальність інформації та забезпечує зручну демонстрацію виконаних проєктів. Окрім цього, передбачається функціонал створення та редагування статей у блозі, що сприяє наповненню сайту контентом та покращує його інформаційну цінність.

Для підвищення ефективності роботи з текстовим наповненням використаємо API сервіси з лінгвістичними моделями штучного інтелекту, що дозволяє автоматизувати процеси генерації та оптимізації текстів. Це спрощує та значно прискорює підготовку матеріалів для блогу, що в свою чергу сприяє кращій адаптації до вимог пошукових систем. Важливою складовою є також можливість запитами клієнтів, що забезпечує контроль комунікації та своєчасну обробку звернень.

У межах взаємодії з клієнтами система повинна передбачати реалізацію форми двостороннього зв'язку у вигляді брифу. Користувач заповнює необхідні дані, описує запит або замовлення, після чого інформація надсилається адміністратору сайту для подальшої обробки. Адміністратор, у свою чергу, дасть відповідь через вказані користувачем контактні дані. Додатково буде передбачена можливість надсилання файлів. Система буде підтримувати як проведення фотосесій із фізичною присутністю клієнта, так і надання дистанційних послуг з обробки фотографій, що дозволяє розширити перелік доступних послуг і охопити ширшу аудиторію.

Крім функціональних можливостей, важливою вимогою до системи є її масштабованість, висока швидкість обробки запитів та надійність. Це дозволяє забезпечити стабільну роботу вебсайту навіть при зростанні кількості користувачів і обсягів даних, а також створює умови для подальшого розвитку без необхідності повної заміни програмного забезпечення.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

Оскільки обрана ніша передбачає високий рівень конкуренції, критично важливим етапом розробки є проведення детального аналізу існуючих рішень та потенційних конкурентів на ринку. Це дозволить виявити галузеві стандарти, а також визначити напрями для впровадження унікального функціоналу нашого продукту.

Як перший об'єкт дослідження було обрано веб-ресурс, що займає перші позиції у пошуковій видачі за відповідними запитами - персональний сайт-портфоліо фотографа (рис. 1.1).

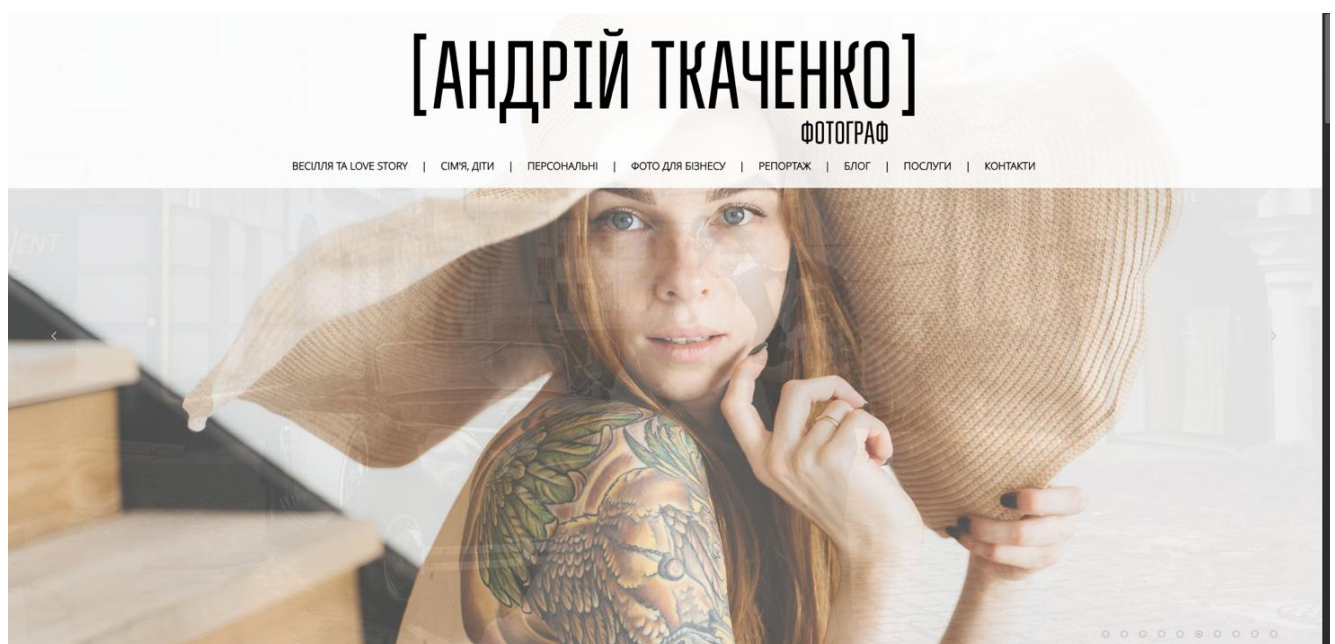


Рис. 1.1. Головна сторінка сайту фотографа

На головній сторінці реалізовано класичну архітектуру, характерну для вебресурсів такого типу, що забезпечує зручність використання та швидке орієнтування користувача. Навігація сайту є інтуїтивно зрозумілою: меню структуроване за типами послуг, зокрема весільна зйомка, Love Story, сімейні фотосесії та інші, що дозволяє користувачам оперативно знаходити потрібний розділ і отримувати доступ до відповідного контенту.

Візуальна складова сайту виконана у сучасному мінімалістичному стилі, що сприяє концентрації уваги на основному — фотографіях. Такий підхід дозволяє ефективно презентувати роботи автора без зайвих відволікаючих елементів. Кожна

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

категорія наповнена значною кількістю прикладів, що дає змогу потенційним клієнтам оцінити рівень та стиль виконання.

Окрему увагу приділено доступності контактної інформації: відповідний блок розміщений у зоні прямої видимості, що спрощує процес зв'язку з фотографом і підвищує зручність взаємодії для користувачів (рис. 1.2).

ВІДПРАВИТИ ЗАПИТ НА ФОТОЗЙОМКУ

Взнати вартість, умови та вільні дати

Ім'я *

Тип зйомки

Viber, telegram *

Місто проведення зйомки

ВІДПРАВИТИ

Рис. 1.2. Контактна форма

Загалом сайт справляє позитивне враження, має сучасний та стильний вигляд, а також містить велику кількість фотографій у різних тематичних категоріях, що дозволяє повною мірою оцінити рівень і різноманітність робіт фотографа.

Водночас більш детальний технічний аналіз виявив ряд недоліків, які можуть бути використані як точки росту при розробці власного проєкту. Зокрема, спостерігаються прогалини в SEO-стратегії. На сайті наявний розділ «Блог», однак на даний момент він фактично не наповнений тематичними матеріалами. Відсутність текстового контенту негативно впливає на довгострокове ранжування ресурсу в пошукових системах, а також знижує рівень залученості потенційних клієнтів, які здійснюють пошук за інформаційними запитам.

Крім того, на сайті відсутня сторінка з орієнтовними цінами на послуги. Незважаючи на те, що вартість може змінюватися залежно від індивідуальних вимог замовника, відсутність навіть базової цінової інформації може стримувати

частину потенційних клієнтів, зокрема тих, які орієнтуються на певний бюджет при виборі послуг.

На головній сторінці розташована контактна форма. На мій погляд це дуже зручний спосіб для комунікації з клієнтом, адже просто контакт телефону зараз вже не актуальний. Клієнт може подзвонити в не дуже зручний час, і це може вплинути на якісь обслуговування та перше враження.

Але я хочу піти далі та розширити функціонал такої форми до вигляду брифу. Це допомогло б краще підготуватись до комунікації з клієнтом та зрозуміти що він хоче від конкретної фотосесії ще до особистого спілкування.

Наступним об'єктом мого дослідження став веб-сайт, який демонструє високі показники релевантності та посідає лідируючі позиції у пошуковій видачі за широким спектром цільових запитів. Даний ресурс представляє інтерес з точки зору впровадження інтерактивних інструментів взаємодії з користувачем (рис. 1.3).

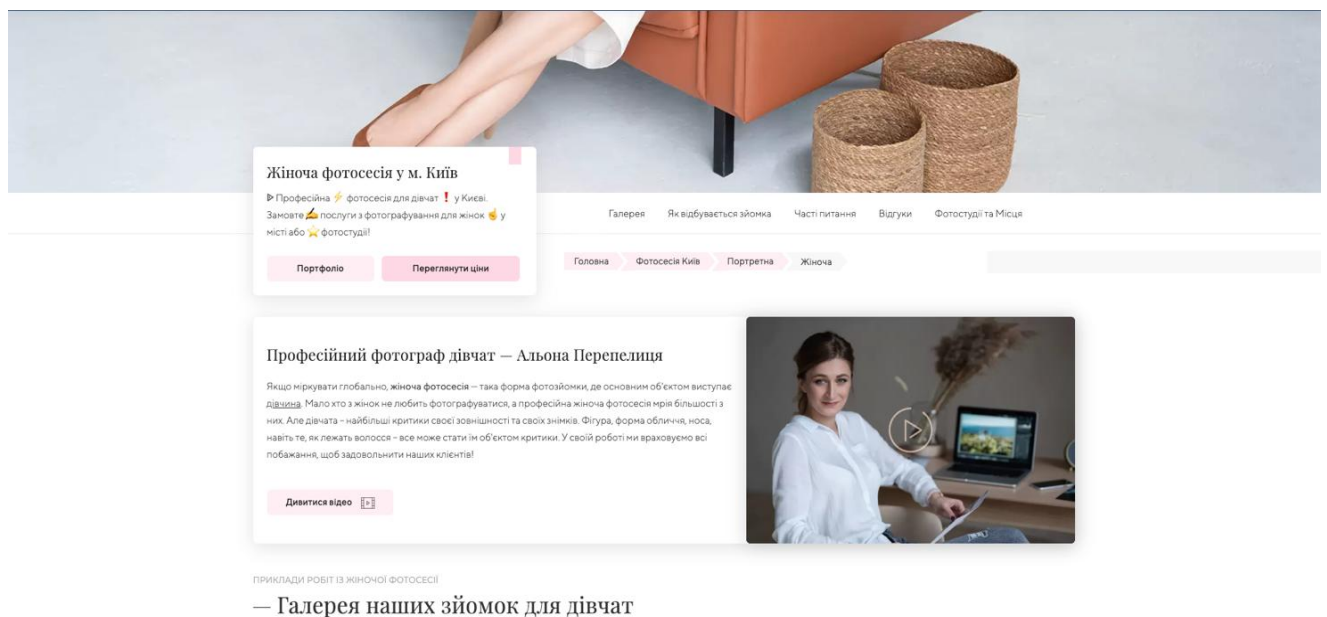


Рис. 1.3. Головна сторінка сайту

Під час аналізу було виявлено наступні ключові особливості та функціональні переваги даного рішення:

Інтерактивний калькулятор вартості: На відміну від попереднього аналога, цей ресурс інтегрував модуль автоматичного розрахунку вартості послуг. Це є вагомою конкурентною перевагою, оскільки дозволяє потенційному клієнту

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

миттєво зорієнтуватися в ціновій політиці, персоналізувати пакет послуг під власні потреби та отримати фінальний кошторис без прямого контакту з менеджером. Такий підхід значно підвищує рівень довіри та автоматизує первинну воронку продажів. На мою думку це дуже цікава ідея та реалізація, що дозволить клієнту краще орієнтуватися в цінах та послугах, а також автоматизовує цей процес.

Важливою функціональною особливістю ресурсу є інтеграція інтерактивного калькулятора вартості послуг. Такий підхід виступає вагомою конкурентною перевагою, оскільки дозволяє потенційному клієнту швидко зорієнтуватися в ціновій політиці, самостійно сформувати пакет послуг відповідно до власних потреб і отримати орієнтовний кошторис без необхідності прямого контакту з менеджером. Це не лише підвищує рівень довіри до сервісу, але й автоматизує первинний етап взаємодії з клієнтом.

Подібне рішення можна вважати доцільним і перспективним для впровадження, оскільки воно спрощує процес прийняття рішення та покращує користувацький досвід.

Архітектура сайту побудована за принципом мультилендінгу, де кожен розділ, присвячений окремому виду послуг, реалізований у вигляді самостійної цільової сторінки. Такий підхід дозволяє більш детально розкрити особливості конкретної послуги, продемонструвати релевантне портфоліо та розмістити тематичні відгуки, що підвищує рівень сприйняття інформації та впливає на прийняття рішення клієнтом.

Для забезпечення ефективної лідогенерації на сайті реалізовано інтуїтивно зрозумілу контактну форму. Мінімальна кількість обов'язкових полів для заповнення знижує бар'єр для користувача та підвищує ймовірність конверсії відвідувача у потенційного клієнта. Такий підхід дозволяє спростити процес подання заявки та зробити його максимально швидким і зручним.

Загальна логіка побудови сайту та його юзабіліті є продуманими та обґрунтованими. Поділ контенту за тематичними категоріями забезпечує структурованість інформації та дозволяє користувачу зосередитися на потрібному

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

типі послуг без відволікання на зайві елементи. Це позитивно впливає на якість користувацького досвіду та підвищує ефективність взаємодії з вебсайтом.

Розділення сторінок на сайті залежить від тематики фотосесії, і на мою думку це правильна та логічна концепція. Сторінка кожного формату зйомки представлена у вигляді лендінгу, що виглядає цікаво. Для замовлення фотосесії також представлена контактна форма, що як ми розглянули раніше дуже зручно.

Даний конкурент демонструє високий рівень технічної складності сайту та маркетингового орієнтування. Найбільш успішним рішенням, яке доцільно врахувати при розробці нашого програмного продукту, є впровадження інструментів автоматизації (калькуляторів) та використання лендінгової структури для окремих сервісів (рис. 1.4).

— Ціни на зйомку для жінок

Калькулятор

4900 грн

1 година зйомки

1 фото локація

60 фото у кольорі (6 у ретуші)

Передача через Google Drive

Готовність 10 днів

☐ Прискорена обробка

☐ Оформити на подарунок

Замовити

Готові пакети:

- ☐ Калькулятор
- ☐ Light
- ☐ Standart
- ☐ Street
- ☐ Стилист

Додаткові послуги:

- ☒ Фотозйомка, година: 1
- ☐ Оренда фотостудії, година: 1
- ☐ Макіяж/Чоловічий тон: 1
- ☐ Укладання від стиліста: 1
- ☐ Дод. ретуш, шт: 1
- ☐ Передача на флешці: 1
- ☐ Фотокнига "Light": 1
- ☐ Фотокнига "Standart": 1

Як користуватися калькулятором?

У цьому інструменті ви можете самостійно створити власний пакет для ідеальної фотозйомки. Кожен пункт буде додавати та змінювати загальне ваше замовлення.

Рис. 1.4. Калькулятор цін

Третім об'єктом дослідження став персональний сайт фотографа "PORIVAEV PHOTOGRAPHY". Цей веб-ресурс є чудовим прикладом поєднання лаконічного дизайну, грамотної структуризації контенту та прозорості цінової політики, що робить його сильним конкурентом на ринку фотопослуг.

На головній сторінці (рис. 1.5) реалізовано класичний, але дуже ефективний підхід до оформлення першого екрану. Відразу привертає увагу розгорнута навігаційна панель, яка дозволяє швидко перейти до потрібного типу послуг або супутньої інформації (відгуки, портфоліо, ціни).

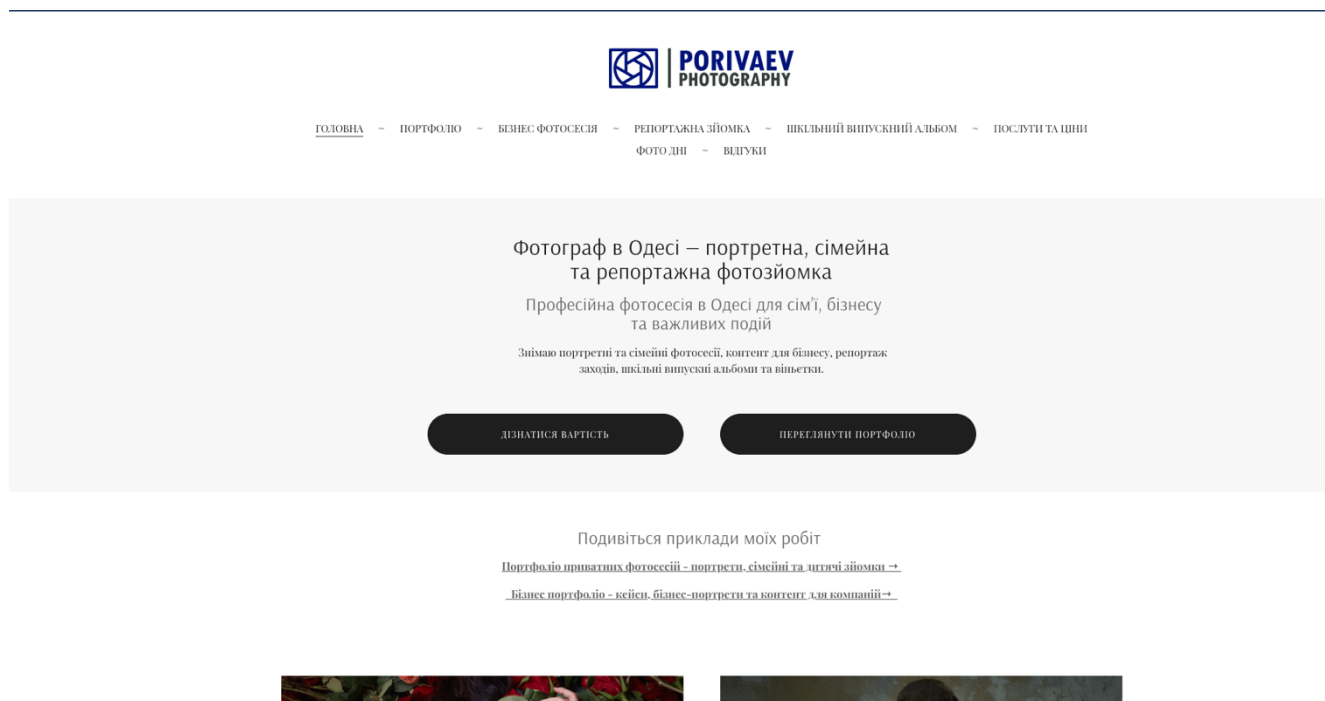


Рис. 1.5. Головна сторінка сайту "PORIVAEV PHOTOGRAPHY"

Головний екран містить чітку унікальну торгову пропозицію з описом спеціалізації фотографа та дві акцентовані кнопки заклику до дії Дізнатися вартість та Переглянути портфоліо. Це надзвичайно вдале рішення з точки зору юзабіліті, оскільки воно відразу пропонує користувачу два найпопулярніші сценарії взаємодії з сайтом: оцінити візуальну якість робіт або перейти до фінансових умов. Дизайн виконаний у світлих тонах, що створює відчуття простору та дозволяє сфокусуватися на інформації.

Перейшовши до розділу портфоліо (рис. 1.6), можна відзначити грамотну категоризацію робіт. Замість суцільного масиву фотографій, який часто перевантажує відвідувача, користувачу пропонується вибрати конкретний напрямок: жіноча, чоловіча або дитяча фотосесія. Кожна категорія супроводжується якісним прев'ю-зображенням. Такий підхід значно полегшує

навігацію і дозволяє потенційному клієнту відразу переглядати саме ті приклади робіт, які відповідають його запиту, заощаджуючи час.



Рис. 1.6. Розділ загального портфоліо з категоріями

При переході в конкретну категорію, наприклад Чоловіча фотосесія (рис. 1.7), дизайн сторінки залишається мінімалістичним, концентруючи всю увагу відвідувача виключно на візуальному контенті. Фотографії розміщені у вигляді сітки з дотриманням єдиної стилістики. Відсутність масивних текстових блоків на цьому етапі є цілком виправданою, оскільки головне завдання цієї сторінки - наочно продемонструвати рівень майстерності та стиль автора у конкретному жанрі.

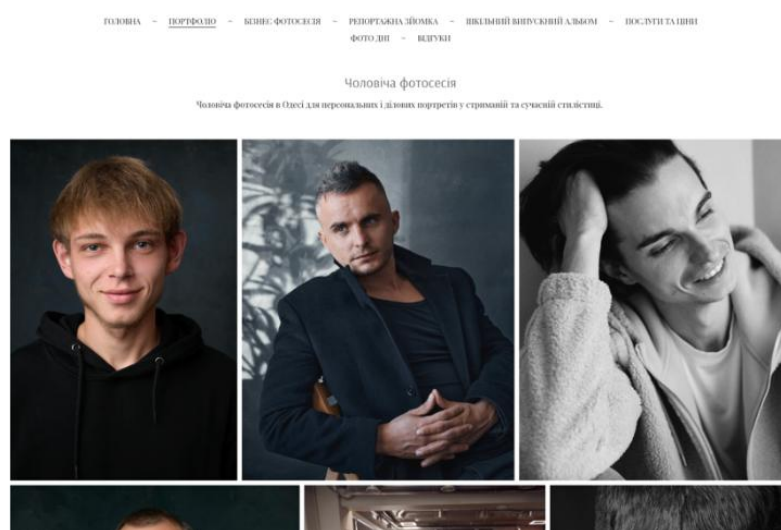


Рис. 1.7. Внутрішня сторінка категорії портфоліо

Окремої уваги та аналізу заслуговує сторінка з вартістю послуг (рис. 1.8), якої суттєво бракувало першому проаналізованому аналогу. Тут цінова політика представлена у вигляді чітких та деталізованих тарифних пакетів: Фото день Стандарт, Індивідуальний Експрес, Індивідуальний Оптимальний та Репортаж.

Вартість фотосесій в Одесі

Нижче наведені актуальні формати фотозйомки та орієнтовна вартість послуг. Фінальна ціна залежить від формату зйомки, тривалості та ваших побажань.



Фото день Стандарт

Тривалість зйомки- 45хв

Студія включена

У підсумку: не менше 70 фото
плюс 5 фото в ретуші

Особиста онлайн-галерея для
перегляду і завантаження
зображень терміном на 3 місяці

Вартість 2500грн

Тариф діє тільки у фото дні!
Дізнайтеся, коли наступний або
подайте заявку на участь.



Індивідуальний Експрес

Тривалість зйомки- 60хв

У підсумку: не менше 70 фото
плюс 10 фото в ретуші

Особиста онлайн-галерея для
перегляду і завантаження
зображень терміном на 3 місяці

Вартість 3500 грн

Студія оплачується окремо



Індивідуальний Оптимальний

Тривалість зйомки- 90хв

У підсумку: не менше 100 фото
плюс 15 фото в ретуші

Особиста онлайн-галерея для
перегляду і завантаження
зображень терміном на 3 місяці

Вартість 4500 грн

Студія оплачується окремо



Репортаж, зйомка заходів

Тривалість зйомки- від 60хв

У підсумку: не менше 100 фото/
год

Готовність фото наступного дня

Особиста онлайн-галерея для
перегляду і завантаження
зображень терміном на 3 місяці

Вартість 3500грн

кожна наступна година зйомки
2000грн

Рис. 1.8. Сторінка "Послуги та ціни" з пакетними пропозиціями

Такий формат подачі інформації є максимально зручним для клієнта. Прозора цінова політика підвищує довіру до спеціаліста та автоматично відфільтровує нецільову аудиторію, зменшуючи навантаження на комунікацію.

Цікавою особливістю, яку варто виділити, є згадка про надання клієнту особистої онлайн-галереї для перегляду і завантаження зображень терміном на 3 місяці. Це вагома додаткова цінність послуги. З технічної точки зору, при розробці нашого програмного продукту, реалізація подібного модуля із можливістю завантаження файлів та обмеженням доступу за часом або паролем може стати суттєвою конкурентною перевагою нашої системи.

Таблиця 1.1.

**Порівняння функціональних можливостей аналогів та розроблюваної
платформи**

Характеристика	Сайт-портфоліо	Сайт з калькулятором	Porivnaev	Проектowana вебплатформа
Структуроване портфоліо за категоріями	+	+	+	+
Відкрита інформація про ціни та пакети послуг	-	+	+	+
Форма зворотного зв'язку / Деталізований бриф клієнта	+/-	+/-	+/-	+
Інтерактивний калькулятор вартості	-	+	-	+
Персональна клієнтська онлайн-галерея	-	-	+	+
Наявність наповненого блогу (SEO-оптимізація)	-	-	-	+
Мультилендінгова структура для окремих послуг	-	+	-	+
Інтеграція ШІ-функцій для обробки та аналізу контенту	-	-	-	+

1.3. Архітектура системи

Під час проектування вебплатформи робочого місця фотографа ключовим етапом є вибір архітектури програмної системи. Від обраного архітектурного підходу залежать швидкість завантаження сторінок, коректність індексації

пошуковими системами, зручність управління контентом, а також можливість подальшого розширення функціоналу без суттєвої перебудови структури проєкту.

З урахуванням поставлених завдань, масштабу проєкту та цільової аудиторії було обрано монолітну архітектуру на основі сучасного РНР-фреймворку Laravel [15]. На відміну від мікросервісної архітектури, яка вимагає значних ресурсів на підтримку інфраструктури та налаштування зв'язків між сервісами, монолітний підхід передбачає реалізацію всіх функціональних модулів (публічна частина сайту, блог, портфоліо, адміністративна панель, підсистема клієнтських галерей, інтеграції з зовнішніми сервісами) в межах єдиного застосунку. Така архітектура є оптимальною для персонального бренду фотографа або невеликої студії, оскільки забезпечує простоту розгортання, централізоване управління кодовою базою та суттєво зменшує витрати на хостинг.

В основі побудови системи лежить класичний архітектурний патерн MVC, який забезпечує чіткий поділ логіки застосунку на три взаємопов'язані компоненти. Першим з них є Модель, яка відповідає за взаємодію з базою даних та містить основну бізнес-логіку системи, описуючи такі ключові сутності, як користувач або клієнт, галерея, зображення та бронювання або бриф. За візуальне відображення цих даних кінцевому користувачеві відповідає Представлення. Зокрема, у публічній частині платформи для цього використовується шаблонізатор Blade, що гарантує швидкий серверний рендеринг, тоді як в адміністративній панелі застосовується технологія Inertia.js у зв'язці з сучасними фронтенд-фреймворками, такими як Vue.js або React. Зв'язуючою ланкою між ними виступає Контролер, завдання якого полягає у прийнятті HTTP-запитів від користувача, їх обробці, зверненні до відповідних Моделей для отримання необхідної інформації та подальшій передачі цих даних у Представлення для фінального виводу на екран.

Основний акцент у розробці публічної частини робиться на SEO-адаптивності та швидкодії. Використання Laravel дозволяє формувати серверний рендеринг сторінок, що забезпечує коректну та швидку індексацію портфоліо, статей блогу, окремих фотопроектів та посадкових сторінок пошуковими

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

роботами. Це критично важливо для просування персонального бренду фотографа, залучення органічного трафіку та підвищення видимості в конкурентному середовищі. Структура URL-адрес, метатеги, Open Graph-розмітка для соціальних мереж, автоматична генерація sitemap.xml та інші інструменти пошукової оптимізації інтегруються безпосередньо на рівні серверної логіки.

Оскільки вебплатформа фотографа передбачає роботу з великим обсягом візуального контенту (фотографії високої роздільної здатності), особливу увагу приділено архітектурі зберігання та доставки файлів. Замість зберігання всіх медіафайлів безпосередньо на локальному сервері застосунку, архітектура передбачає використання хмарних об'єктних сховищ (наприклад, Amazon S3 або його аналогів). Для мінімізації часу завантаження зображень клієнтами з різних географічних точок передбачена інтеграція з мережею доставки контенту. Це дозволить кешувати статичний контент на серверах, фізично наближених до кінцевого користувача, що позитивно вплине на показники Core Web Vitals.

Адміністративна частина системи орієнтована на максимально зручне управління контентом: створення та редагування статей, додавання фотосерій, управління категоріями, тегами, ціновими пакетами, а також обробку вхідних брифів від клієнтів. Для реалізації високої інтерактивності в адмін-панелі використовується пакет Laravel Inertia.js [14]. Ця технологія дозволяє створювати динамічні користувацькі інтерфейси без необхідності розробки окремого REST API та підтримки двох розрізнених кодових баз (фронтенду та бекенду). Inertia.js значно спрощує роботу з формами, валідацією даних на льоту, Drag-and-Drop завантаженням фотографій та управлінням станом інтерфейсу, зберігаючи при цьому всю бізнес-логіку та маршрутизацію на захищеному сервері.

Інтеграція модулів штучного інтелекту є ключовою функціональною особливістю платформи, тому під неї виділено окремий архітектурний прошарок. Оскільки процеси обробки зображень алгоритмами машинного навчання (наприклад, автоматичне тегування фотографій, розумний пошук за змістом зображення, генерація SEO-описів, або покращення якості) є ресурсомісткими і

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

можуть тривати від кількох секунд до хвилин, їх виконання у синхронному режимі призвело б до зависання інтерфейсу користувача. Тому архітектура системи використовує асинхронну обробку завдань.

Процес взаємодії з модулем штучного інтелекту побудований за принципом асинхронної обробки і розгортається у кілька етапів. Спочатку адміністратор завантажує фотографію в систему через користувацький інтерфейс, реалізований за допомогою Inertia.js. Далі системний контролер приймає цей файл, надійно зберігає його у хмарному S3-сховищі та одночасно створює відповідний запис у базі даних із початковим статусом «В обробці». Після цього завдання на аналіз завантаженого зображення автоматично додається у чергу за допомогою інструменту Laravel Queues, де в ролі брокера повідомлень може використовуватися Redis або сама реляційна база даних.

На наступному етапі спеціальний фоновий обробник перехоплює це завдання з черги та ініціює запит до зовнішнього AI API, яким може виступати OpenAI API, Google Cloud Vision або інша інтегрована модель. Зрештою, після успішного отримання результатів аналізу від штучного інтелекту, воркер оновлює метадані фотографії безпосередньо в базі даних, автоматично додаючи згенеровані теги, текстові описи чи інші параметри, та змінює статус файлу на «Оброблено». Завершується цей цикл тим, що актуалізована інформація миттєво відображається в інтерфейсі користувача завдяки використанню технології WebSockets, що дозволяє уникнути необхідності ручного оновлення сторінки. Для зберігання текстових даних, налаштувань платформи та зв'язків між сутностями використовується реляційна система управління базами даних, наприклад PostgreSQL або MySQL. Реляційна модель є оптимальною для організації строгих ієрархічних даних: інформації про фотопроекти, статті блогу, категорії, теги, профілі користувачів та деталізовані брифи клієнтів. Використання SQL дозволяє ефективно формувати складні вибірки для відображення динамічного портфоліо, фільтрації послуг та аналітики звернень.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Важливим компонентом архітектури є підсистема безпеки. Платформа використовує вбудовані механізми захисту Laravel для протидії поширеним веб-вразливостям, таким як SQL-ін'єкції, міжсайтовий скриптинг та підробка міжсайтових запитів. Доступ до адміністративної панелі та клієнтських онлайн-галерей регулюється системою аутентифікації та рольового доступу, що гарантує конфіденційність фотоматеріалів клієнтів.

Таким чином, обрана монолітна архітектура на базі фреймворку Laravel у поєднанні з Inertia.js для адміністративної частини, хмарним зберіганням для медіафайлів та асинхронними чергами для обробки ШІ-завдань, забезпечує ідеальний баланс між високою продуктивністю, SEO-ефективністю, надійністю та зручністю управління контентом. Вона закриває всі поточні потреби робочого місця професійного фотографа і залишає можливості для подальшого масштабування продукту [14, 15].

1.4. Обґрунтування вибору засобів реалізації

На етапі проєктування та вибору технологічного стеку для реалізації вебплатформи портфоліо-блогу фотографа першочергова увага приділялася забезпеченню високої SEO-ефективності, стабільності обробки великих обсягів великоформатного медіаконтенту та створенню максимально ергономічного інтерфейсу адміністрування.

Специфіка діяльності фотографа вимагає, щоб вебплатформа поєднувала в собі два протилежні архітектурні підходи. Публічна частина сайту повинна завантажуватися миттєво, мати статичну структуру для ефективної індексації пошуковими роботами та відповідати суворим критеріям Core Web Vitals.

Водночас внутрішня адміністративна панель має функціонувати як насичений інтерактивний додаток, що забезпечує швидке керування медіафайлами, динамічне налаштування параметрів зйомок і безшовну взаємодію із зовнішніми сервісами штучного інтелекту. Обраний інструментарій повинен гарантувати цілісність даних, високу швидкість обробки черг асинхронних завдань та безпеку від поширених вразливостей без надмірного ускладнення інфраструктури.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Серверна архітектура вебплатформи реалізується на основі сучасного об'єктно-орієнтованого фреймворку Laravel, який надає розробнику повноцінний інструментарій для побудови надійного монолітного застосунку. Використання цього фреймворку забезпечує зручну роботу з базою даних, гнучку систему маршрутизації, механізми кешування та керування чергами завдань, що є особливо актуальним при обробці зображень або взаємодії з AI API.

Організація архітектурних рішень у межах фреймворку спирається на патерни проєктування корпоративних застосунків, описані у головних працях Мартіна Фаулера [8]. Побудова системи на базі монолітного архітектурного підходу є оптимальною для створення єдиного цифрового середовища фотографа, оскільки забезпечує простоту розгортання та централізоване управління кодовою базою. Екосистема фреймворку володіє потужними вбудованими інструментами для реалізації складної фонові логіки, що є критично важливим у контексті інтеграції з хмарними сховищами через Google Drive API та взаємодії із зовнішніми сервісами штучного інтелекту.

Завдяки компоненту керування чергами завдань, система отримує можливість виконувати ресурсомісткі операції, такі як первинна обробка зображень, валідація важких медіафайлів та очікування відповідей від віддалених AI-моделей, в асинхронному режимі. Це дозволяє розвантажити основний потік вебсервера, запобігти виникненню таймаутів з'єднання та забезпечити миттєвий відгук інтерфейсу для кінцевого користувача. Додатково Laravel вирішує комплекс завдань, пов'язаних із безпекою, надаючи автоматичний захист від міжсайтового скриптингу, підробки запитів, SQL-ін'єкцій та вбудовані механізми гнучкого розмежування прав доступу на рівні посередників.

Побудова публічного інтерфейсу вебплатформи, який охоплює сторінки портфоліо, тематичний блог, інформаційні блоки та модулі зворотного зв'язку, базується на концепції серверного рендерингу за допомогою вбудованого шаблонізатора Blade. Використання Blade дозволяє формувати фінальний HTML-код сторінки безпосередньо на стороні сервера перед його відправкою в браузер

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

клієнта. Такий підхід є фундаментальним для забезпечення високої видимості сайту в пошукових системах, оскільки роботам індексації не потрібно виконувати складні клієнтські JavaScript-сценарії для зчитування контенту. Сторінки, згенеровані сервером, містять повний набір метаданих, семантично коректну структуру заголовків та інтегровану мікророзмітку, що позитивно впливає на позиції платформи в пошуковій видачі. Відмова від використання важких клієнт-орієнтованих фреймворків у публічній зоні мінімізує обсяг завантажуваного коду, що безпосередньо знижує час до першого відмальовування контенту та забезпечує відповідність сучасним стандартам швидкодії Core Web Vitals на мобільних пристроях.

Для реалізації адміністративної панелі платформи застосовано комбінований підхід, що базується на інтеграції Inertia.js та бібліотеки React. Ця зв'язка дозволяє уникнути класичного розділення проєкту на два ізольовані застосунки (backend та frontend) і усуває потребу в розробці та документуванні окремого RESTful API.

Інструментарій Inertia.js виступає в ролі архітектурного місту, який дозволяє використовувати реактивні компоненти React як звичайні серверні шаблони. Під час взаємодії менеджера чи фотографа з адміністративною частиною, система не перезавантажує сторінку повністю, а здійснює асинхронні запити, отримуючи від сервера лише чисті дані у форматі JSON і динамічно оновлюючи інтерфейс користувача. Такий підхід сприяє збереженню основної бізнес-логіки на сервері, забезпечуючи при цьому високий рівень зручності та продуктивності користувацького інтерфейсу [20]. Це забезпечує плавність роботи на рівні Single Page Application, що є надзвичайно зручним при завантаженні серій фотографій, налаштуванні параметрів інтерактивного брифу або редагуванні текстових матеріалів за допомогою штучного інтелекту. Вся ключова бізнес-логіка, валідація, авторизація та маршрутизація залишаються під контролем сервера, тоді як React відповідає виключно за високу швидкість відгуку та інтерактивність клієнтського інтерфейсу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Компонент стилізації інтерфейсу побудований на використанні утилітарного CSS-фреймворку Tailwind CSS. Відхід від традиційних монолітних стильових таблиць на користь низькорівневих класів-утиліт дозволяє конструювати адаптивний дизайн безпосередньо в коді розмітки Blade або компонентів React [25].

Tailwind CSS забезпечує автоматичну оптимізацію фінальних файлів стилів за допомогою інструментів збірки, які видаляють весь невикористаний код під час фінальної генерації проєкту. Це дозволяє отримати надзвичайно легкі стильові файли, які завантажуються та обробляються браузером за лічені секунди, що вкрай важливо для оптимізації швидкодії вебзастосунків. Використання фіксованої конфігураційної сітки фреймворку гарантує візуальну цілісність усіх елементів інтерфейсу, кросбраузерність та швидку адаптацію відображення під екрани будь-яких портативних чи десктопних пристроїв.

Рівень персистентності та збереження структурованих даних базується на використанні потужної об'єктно-реляційної системи керування базами даних PostgreSQL. Рівень управління даними реалізовано на базі PostgreSQL — сучасної СУБД, відомої своєю надійністю, архітектурною стабільністю та гнучкістю при роботі зі складними структурами даних [19]. Реляційна структура є оптимальним вибором для вебплатформи фотографа, оскільки інформаційна модель проєкту містить велику кількість чітко типізованих сутностей із вираженими зв'язками між ними, таких як користувачі, замовлення, фотопроекти, категорії, статті блогу та параметри брифів. Наявність вбудованої підтримки типу даних JSONB у PostgreSQL створює додаткову гнучкість, дозволяючи зберігати неструктуровані або динамічні відповіді від зовнішніх AI-сервісів без необхідності постійної зміни схеми бази даних. Використання ефективного індексування за зовнішніми ключами та полями статусів забезпечує високу швидкість виконання запитів в адміністративній панелі.

Аналіз апаратних та системних вимог для успішного розгортання та експлуатації розробленої системи вказує на оптимізоване споживання ресурсів серверної інфраструктури. Для стабільного функціонування платформи в режимі

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

мінімального навантаження достатньо базової конфігурації віртуального виділеного сервера, що містить двоядерний процесор із частотою від двох гігагерц та обсяг оперативної пам'яті від трьох гігабайт. Дискова підсистема сервера повинна мати місткість від шістдесяти гігабайт, причому критично важливим є використання накопичувачів типу SSD, які суттєво підвищують швидкість запуску PHP-скриптів та обробку запитів до бази даних PostgreSQL. Оскільки основна частина важких медіафайлів фотографа завантажується асинхронно у хмарне сховище Google Drive, локальний диск сервера використовується переважно для зберігання ядра системи, кешу, текстового контенту бази даних та тимчасових файлів під час сесій обробки.

В ролі системного оточення вебсервера може виступати середовище Nginx або Apache з обов'язковим встановленням менеджера процесів PHP-FPM актуальної версії. Для забезпечення безперебійної роботи фонових черг завдань на сервері налаштовується системний монітор процесів Supervisor, який автоматично перезапускає консольні воркери Laravel у разі їх аварійного завершення.

Для локальної контейнеризації інфраструктури бази даних та ізоляції СУБД від основного операційного середовища розробника застосовується інструментарій Docker, що дозволяє швидко розгортати та масштабувати необхідні сервіси. Клієнтська частина платформи повністю відповідає стандартам сучасного вебконструювання та не висуває жодних специфічних вимог до кінцевого обладнання користувачів. Завдяки адаптивності Tailwind CSS та відсутності блокуючих скриптів у публічній зоні, інтерфейс коректно, швидко та інформативно відображається у будь-кому сучасному веббраузері на персональних комп'ютерах, планшетах чи смартфонах незалежно від встановленої операційної системи.

Висновки до першого розділу

У першому розділі було розглянуто предметну область розробки вебплатформи робочого місця фотографа, а також визначено основні вимоги до майбутньої системи. Проведено аналіз ролі цифрової присутності у сфері фотографії та особливостей просування послуг у мережі Інтернет, що дозволило

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

обґрунтувати необхідність створення сучасного вебресурсу для представлення робіт, ведення блогу та взаємодії з клієнтами. Окрему увагу приділено конкурентному середовищу, що підтвердило важливість якісної презентації контенту, SEO-оптимізації та зручних механізмів комунікації з потенційними замовниками.

На основі проведеного аналізу було сформовано функціональні та нефункціональні вимоги до системи. Зокрема, визначено необхідність реалізації портфоліо з можливістю структурованого представлення робіт, блогу для публікації статей, а також механізмів взаємодії з користувачами у вигляді форми зворотного зв'язку та обробки запитів. Додатково обґрунтовано потребу у впровадженні SEO-інструментів, інтеграції з AI-сервісами для генерації текстового контенту та забезпеченні високої продуктивності й масштабованості системи.

Також у розділі було обрано та обґрунтовано технологічний стек для реалізації проєкту. В якості серверної основи використано фреймворк Laravel, який забезпечує структуровану архітектуру застосунку, зручну роботу з базою даних та вбудовані механізми безпеки. Для реалізації адміністративної панелі застосовано Inertia з react.js, що дозволяє створювати інтерактивний інтерфейс. Публічна частина системи реалізується із використанням Blade-шаблонізатора, що забезпечує SEO-оптимізований серверний рендеринг сторінок. Для стилізації інтерфейсу обрано Tailwind CSS, а для зберігання даних PostgreSQL, що гарантує надійність та ефективну роботу з реляційною структурою даних.

У результаті сформовано бачення цілісного технологічного рішення, яке відповідає вимогам до сучасних вебплатформ та забезпечує основу для подальшої реалізації функціональної, продуктивної та масштабованої системи.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

У даному розділі здійснено визначення варіантів використання та описано об'єктно-орієнтовану структуру вебплатформи робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту. Основною метою системи є забезпечення ефективного представлення портфоліо, автоматизації взаємодії з клієнтами, обробки медіаконтенту із застосуванням інструментів штучного інтелекту, а також організації процесів замовлення фото.

Розроблена система орієнтована на вирішення комплексу прикладних задач, що охоплюють демонстрацію портфоліо, збір технічних вимог через систему брифів, управління замовленнями на монтаж та інтеграцію зі сторонніми сервісами для обробки та зберігання файлів. Окреме значення має використання ШІ-функцій, які застосовуються для автоматизації підготовки контенту, зокрема генерації текстових описів, допоміжного редагування матеріалів та покращення процесів роботи з інформаційним наповненням платформи[4].

Архітектура системи побудована за модульним принципом і включає кілька функціональних підсистем. Клієнтський модуль реалізований як інтерактивний вебзастосунок на основі React та Inertia.js, що забезпечує зручний перегляд портфоліо, взаємодію з формами замовлень та заповнення брифів. Модуль управління замовленнями відповідає за повний життєвий цикл заявок, включаючи їх створення, обробку, зміну статусів та контроль виконання. Адміністративний модуль забезпечує керування контентом платформи, налаштуванням послуг, редагуванням матеріалів блогу та загальною конфігурацією системи. Модуль інтеграції зі сторонніми сервісами відповідає за взаємодію з хмарними сховищами та AI API[11], що використовуються для обробки та передачі файлів.

У системі передбачено дві ролі користувачів, це клієнт та адміністратор. Клієнт має можливість переглядати портфоліо, створювати замовлення,

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

заповнювати брифи, відстежувати статус виконання робіт та отримувати результати через інтегровані сервіси зберігання даних. Адміністратор володіє повним доступом до функціоналу системи, включаючи управління користувачами, модерацію контенту, налаштування параметрів послуг, обробку замовлень та контроль роботи AI-інструментів.

Об'єктно-орієнтована структура системи базується на наборі взаємопов'язаних сутностей предметної області, що відображають процеси роботи фотографа. До основних об'єктів належать користувачі, проекти портфолію, категорії зйомок, брифи, замовлення на монтаж, медіафайли, записи блогу та налаштування системи. Кожен об'єкт має власні атрибути та методи, які забезпечують обробку відповідних бізнес-процесів, а взаємодія між ними реалізується через сервісний шар і механізми об'єктно-реляційного відображення.

Таким чином, визначені варіанти використання та сформована об'єктно-орієнтована структура системи забезпечують основу для реалізації вебплатформи, яка поєднує функції керування портфолію, автоматизації замовлень та використання ШІ-технологій для обробки контенту, формуючи цілісне цифрове робоче середовище фотографа.

Функціональні вимоги визначають перелік можливостей, які повинна забезпечувати вебплатформа робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту. Система орієнтована на повний цикл роботи з клієнтами та медіаконтентом: від демонстрації портфолію та збору технічних вимог до управління замовленнями на постпродакшн і автоматизованої взаємодії з хмарними сервісами зберігання даних. Окрему роль відіграє підтримка розмежування доступу між різними типами користувачів, що забезпечує безпеку та коректну організацію бізнес-процесів.

Важливим компонентом функціоналу є керування портфолію, яке включає створення, редагування та видалення робіт, завантаження фото- та відеоматеріалів, а також їх структурування за категоріями та типами зйомок. Це дозволяє забезпечити зручну навігацію та ефективне представлення контенту для кінцевих

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів. Паралельно реалізується система брифів, яка дає змогу збирати детальні технічні вимоги до зйомок залежно від обраного типу послуги, що значно спрощує комунікацію між клієнтом і виконавцем.

Окремим функціональним блоком виступає модуль управління замовленнями на монтаж, який охоплює створення заявок, вибір пакетів послуг, визначення технічних параметрів та контроль статусу виконання робіт. Важливою складовою є інтеграція з Google Drive, що забезпечує автоматизований обмін медіафайлами, створення структурованих папок для проєктів та передачу результатів обробки[12]. Додатково система передбачає ведення блогу, який використовується для публікації матеріалів, кейсів та новин з можливістю керування статусами публікацій.

Технічна реалізація функціоналу включає використання фонових завдань для обробки тривалих операцій, таких як виконання замовлень на монтаж, що дозволяє уникнути блокування інтерфейсу користувача. Також система забезпечує глобальні налаштування платформи, які охоплюють SEO-параметри, контактну інформацію та інші конфігураційні дані. Додатково реалізовано підсистему сповіщень, яка інформує користувачів про зміни статусів замовлень, нові брифи та інші події, а також механізми безпеки, що включають валідацію даних, контроль доступу та логування помилок.

У таблиці 2.1 наведено узагальнену інформацію щодо функціональних вимог системи.

Таблиця 2.1

Функціональні вимоги

Етап	Опис
Ролі (Client/Admin)	Відокремлення функціоналу: клієнт має доступ до своїх замовлень та брифів, адміністратор керує контентом, користувачами та налаштуваннями системи.

Керування портфоліо	Створення, редагування та видалення робіт; завантаження фото- та відеофайлів; групування за категоріями та типами зйомок (PhotoshootType)
Система брифів	Формування технічних завдань для різних типів зйомок; збереження та перегляд заповнених форм адміністратором
Замовлення на графічне редагування фото	Створення заявок на редагування, вибірв послуг, визначення вимог та відстеження статусів виконання
Інтеграція з Google Drive та ai сервісами для обробки фото	Автоматичне створення проєктних папок; завантаження та передача медіафайлів; обмін результатами між клієнтом і адміністратором.
Керування блогом	Створення та редагування публікацій і керування їх статусами
Глобальні налаштування	Керування SEO даними, контактами, соціальними мережами та іншими системними параметрами та контентом
Сповіщення	Інформування про зміни статусів замовлень, нові брифи та інші системні події
Безпека та валідація	Перевірка даних форм, контроль доступу, захист маршрутів та логування помилок системи

Робота вебплатформи робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту починається з авторизації користувача та застосування механізмів розмежування доступу відповідно до визначених ролей. Після входу в систему клієнт отримує доступ до портфоліо, можливості створення замовлень, заповнення брифів та перегляду статусів виконання робіт, тоді як адміністратор має розширені права для керування контентом, замовленнями, користувачами та системними налаштуваннями..

У межах бізнес-логіки платформи користувач взаємодіє з основними модулями системи, що включають роботу з портфоліо, формування технічних брифів, створення замовлень на постпродакшн та управління медіафайлами. Значна частина процесів автоматизується за рахунок інтеграції зі сторонніми сервісами, зокрема хмарними сховищами для передачі великих файлів, а також AI-сервісами,

які використовуються для допоміжної генерації текстів, описів та оптимізації контенту.

Нефункціональні вимоги визначають якісні характеристики системи, що забезпечують її стабільність, продуктивність, безпеку та зручність використання. Вони не описують безпосередній функціонал, однак є критично важливими для коректної роботи платформи в реальних умовах експлуатації та впливають на загальний користувацький досвід.

З точки зору зручності використання, система повинна мати інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам швидко освоїти основні функції без тривалого навчання. Інтерфейс має бути адаптивним і коректно працювати як на настільних комп'ютерах, так і на мобільних пристроях, забезпечуючи гнучкий доступ до функціоналу платформи.

Надійність системи повинна гарантувати стабільну роботу протягом тривалого часу без критичних збоїв, а також коректну обробку помилок під час взаємодії з базою даних, файловими сховищами та зовнішніми API. Продуктивність платформи має забезпечувати одночасну роботу значної кількості користувачів без суттєвого зниження швидкості відгуку інтерфейсу, а також підтримувати асинхронну обробку ресурсоємних операцій, таких як завантаження та обробка медіафайлів або запити до AI-сервісів.

Масштабованість системи передбачає можливість розширення серверних ресурсів та окремих компонентів, зокрема черг обробки завдань і фонових процесів. Це дозволяє ефективно працювати з ростом кількості користувачів, проєктів та обсягу медіаконтенту без втрати продуктивності.

Безпека платформи реалізується через систему автентифікації, рольовий доступ та контроль дозволів. Особлива увага приділяється захисту медіафайлів та персональних даних користувачів, а також безпечній взаємодії з зовнішніми сервісами. Передача даних здійснюється із застосуванням захищених протоколів, що мінімізує ризик несанкціонованого доступу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

Системні вимоги передбачають використання сучасних веббраузерів, які підтримують актуальні стандарти JavaScript та HTML. Платформа є кросплатформною і може використовуватися на операційних системах Windows, macOS та Linux. Для коректної роботи клієнтської частини необхідне стабільне інтернет-з'єднання та достатні ресурси пристрою для обробки мультимедійного контенту.

Серверна частина системи базується на сучасному вебфреймворку, що забезпечує високу продуктивність та гнучкість архітектури. Для зберігання структурованих даних використовується реляційна база даних PostgreSQL, яка гарантує надійність та цілісність інформації. Зберігання файлів реалізується через інтеграцію з хмарними сховищами, що забезпечує масштабованість та швидкий доступ до контенту..

Із зовнішніми сервісами налаштована інтеграція, зокрема AI-інструментами, дозволяє автоматизувати частину процесів обробки контенту, включаючи генерацію текстових описів, допоміжне редагування матеріалів та покращення загальної якості інформаційного наповнення платформи. Це підвищує ефективність роботи фотографа та зменшує час на виконання рутинних операцій.

Отже, дотримання нефункціональних вимог і системних характеристик забезпечує стабільну, безпечну та масштабовану роботу вебплатформи, яка поєднує функції керування портфоліо, обробки замовлень та використання штучного інтелекту для автоматизації роботи з контентом.

Діаграму варіантів використання наведено нижче, вона відображає основні сценарії взаємодії користувачів із вебплатформною та дозволяє наочно представити логіку роботи системи на концептуальному рівні (рис. 2.1). Вона демонструє, яким чином різні категорії користувачів взаємодіють із функціональними модулями системи, а також які дії доступні кожній ролі в межах визначених ролей[30].

Діаграма ілюструє ключові процеси роботи з портфоліо, контактними брифами, замовленнями, а також взаємодію з адміністративною частиною платформи.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

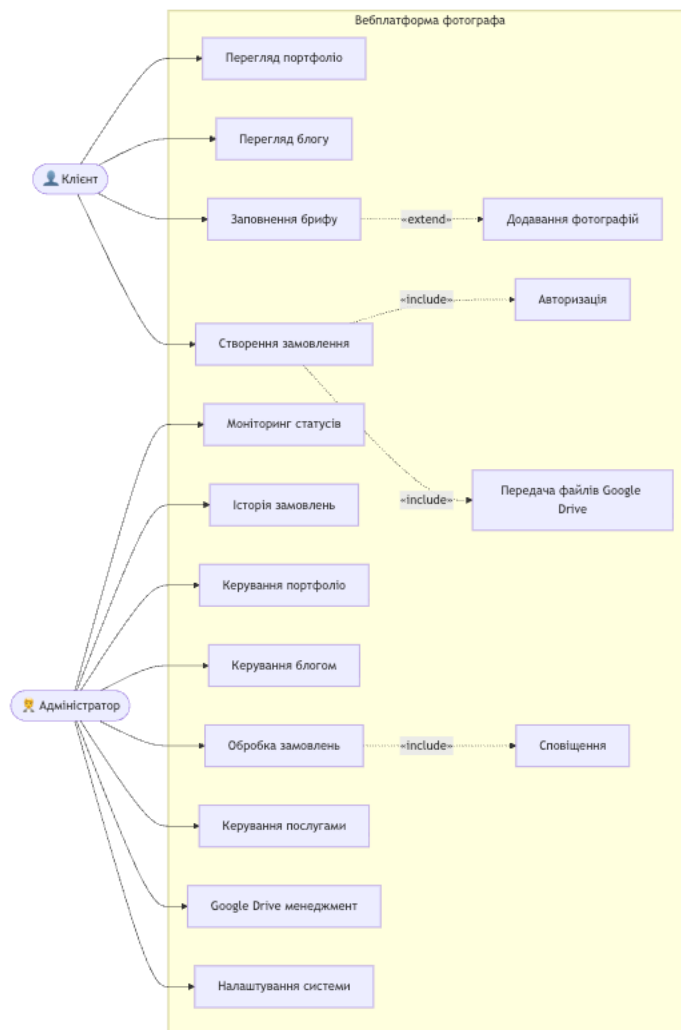


Рис. 2.1. Діаграма варіантів використання

Процес взаємодії клієнта з вебсайтом починається з перегляду портфоліо, що є одним із ключових сценаріїв використання системи. У межах цього функціоналу користувач отримує доступ до галереї виконаних робіт, представлених у вигляді структурованих елементів портфоліо. Додатково реалізовано можливість фільтрації за типами зйомок, що дозволяє швидко знаходити релевантні приклади відповідно до потреб клієнта. Це сприяє кращому розумінню стилю фотографа та формуванню початкового уявлення про майбутній результат.

Окремим сценарієм є перегляд блогу, який містить інформаційні матеріали, кейси та тематичні статті. Цей функціонал дозволяє клієнту ознайомитися з додатковим контентом, що описує процеси роботи, підходи до зйомки та приклади

реалізованих проєктів. Таким чином підвищується рівень довіри до виконавця, а також формується більш усвідомлене бачення майбутньої співпраці.

Процес заповнення брифу є одним із найбільш важливих сценаріїв взаємодії клієнта з системою. У цьому випадку користувач формує технічне завдання на зйомку, вказуючи детальні параметри майбутнього проєкту, такі як локація, стилістика, кількість учасників, додаткові вимоги до обладнання та інші специфічні побажання. Система динамічно адаптує форму брифу відповідно до обраного типу зйомки, що забезпечує зручність та точність збору даних.

Створення замовлення на ретуш фото є наступним етапом взаємодії. Клієнт обирає відповідну послугу з каталогу та формує заявку, у якій зазначає технічні вимоги, бажані терміни виконання та особливості фінального результату. Під час цього процесу також відбувається передача файлів через інтеграцію з хмарним сховищем Google Drive, що забезпечує надійний обмін великими обсягами даних.

Моніторинг статусів замовлення дозволяє клієнту відстежувати поточний стан виконання робіт у режимі реального часу. У системі відображаються етапи життєвого циклу замовлення, такі як очікування, виконання або завершення. Це зменшує потребу у додаткових комунікаціях між клієнтом та виконавцем і підвищує прозорість процесу.

Адміністративна частина системи починається з керування портфоліо, що включає повний цикл створення, редагування та видалення робіт. Адміністратор має можливість завантажувати медіаконтент, додавати описи, встановлювати метадані та SEO-параметри, а також прив'язувати роботи до відповідних категорій і типів зйомок. Це забезпечує структуроване та якісне представлення контенту на платформі.

Наступним елементом є керування блогом, яке передбачає створення та публікацію статей, новин і кейсів. Адміністратор також здійснює управління глобальними налаштуваннями системи, зокрема контактною інформацією, посиланнями на соціальні мережі та SEO-описами, що впливає на загальну видимість платформи в пошукових системах.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

Обробка замовлень є ключовим операційним процесом адміністратора. У цьому модулі здійснюється аналіз вхідних заявок і брифів, перевірка вимог клієнта та визначення можливості виконання замовлення. Після цього замовлення переводиться між етапами виконання, що дозволяє контролювати весь процес постпродакшну.

Контроль виконання та здача робіт тісно пов'язані з інтеграцією Google Drive. Адміністратор отримує доступ до медіафайлів клієнта, виконує необхідну обробку та завантажує готовий результат у відповідну структуру сховища. Зміна статусу замовлення автоматично ініціює сповіщення клієнта про оновлення.

Окремим напрямом є системне адміністрування, яке включає модерацію довідників платформи. Адміністратор керує переліком послуг монтажу та типів зйомок, змінює їх описи, ціни та додає нові напрямки без необхідності втручання в програмний код системи. Додатково здійснюється моніторинг фонових процесів, зокрема черг завдань, які відповідають за синхронізацію даних та відправку системних сповіщень.

2.2. Розробка бази даних системи

Для зберігання та структурованої обробки даних у межах вебплатформи робочого місця фотографа з інтеграцією ШІ-функцій для обробки контенту було спроектовано реляційну базу даних на основі PostgreSQL. Вибір даної системи керування базами даних обумовлений її високою надійністю, стабільністю, підтримкою складних запитів, а також можливістю ефективної роботи з великими обсягами структурованих даних, що особливо важливо для медіаорієнтованих вебплатформ[10].

Проектування моделі даних здійснювалося з урахуванням принципів нормалізації, що дозволило мінімізувати надлишковість інформації та забезпечити логічну цілісність між сутностями системи. Особлива увага приділялася розмежуванню відповідальностей між таблицями, що сприяє зрозумілій структурі зберігання даних і полегшує подальше масштабування системи. Крім того, архітектура бази даних враховує необхідність підтримки історичності змін, що є

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

важливим для відстеження життєвого циклу замовлень, взаємодії з клієнтами та аналізу виконаних проєктів.

Структура бази даних охоплює всі ключові бізнес-процеси платформи. Вона включає сутності, що відповідають за користувачів системи та їхні ролі, управління портфоліо фотографа, категорії та типи зйомок, а також зберігання окремих елементів портфоліо. Окремий блок таблиць відповідає за роботу з брифами, які використовуються для збору технічних вимог від клієнтів перед проведенням зйомок. Це дозволяє формалізувати процес комунікації та зменшити кількість уточнень під час реалізації проєктів.

Окремо реалізовано структури для керування блогом, який використовується як інструмент контент-маркетингу та SEO-просування платформи, а також таблиці системних налаштувань, що дозволяють гнучко конфігурувати параметри роботи вебзастосунку без необхідності внесення змін у код.

Таблиця 2.2

Структура бази даних

Назва таблиці	Призначення
users	Зберігає інформацію про користувачів системи, включаючи їх облікові дані та роль у платформі.
portfolio_items	Містить дані про роботи портфоліо фотографа, зокрема назви проєктів, описи, медіафайли та супровідні метадані.
photoshoot_types	Використовується для зберігання типів фотозйомок, які застосовуються для класифікації робіт і структурування брифів.
briefs	Зберігає заповнені клієнтами технічні завдання на зйомку з усіма необхідними деталями та побажаннями до проєкту.
editing_services	Містить перелік доступних послуг постпродакшну з їх описами, характеристиками та умовами виконання.
editing_orders	Фіксує замовлення на монтаж, включаючи зв'язок із клієнтом, вибраною послугою, технічними вимогами та статусом виконання.
posts	Використовується для зберігання публікацій блогу, новин та описів виконаних проєктів і кейсів.

settings	Містить глобальні налаштування платформи, такі як контактна інформація, SEO-параметри та соціальні посилання.
jobs	Служить для управління фоновими процесами системи, зокрема задачами інтеграції з зовнішніми сервісами.
failed_jobs	Використовується для збереження інформації про помилки, що виникли під час виконання фонових завдань.

Нижче детально розглянемо кожен таблицю і її ключові поля, які відображають основні сутності та бізнес-логіку вебплатформи робочого місця фотографа. Такий підхід дозволяє краще зрозуміти структуру даних, взаємозв'язки між таблицями та їх роль у забезпеченні функціонування всієї системи..

Спершу розглянемо таблицю users (табл. 2.3). Вона є базовою сутністю системи, яка відповідає за зберігання всіх облікових записів платформи. Вона об'єднує як клієнтів, так і адміністратора, забезпечуючи механізм автентифікації та розмежування прав доступу.

Таблиця 2.3

Опис таблиці users

Поле	Тип	Призначення
id	bigint	Первинний ключ, унікальний ідентифікатор користувача
name	VARCHAR(255)	Повне ім'я користувача
email	VARCHAR(255)	електронна адреса
email_verified_at	TIMESTAMP	Час підтвердження email
password	VARCHAR(255)	Хешований пароль
remember_token	VARCHAR(100)	Токен на remember me
role	VARCHAR(255)	Перевірка чи адміністратор
created_at	TIMESTAMP	Дата створення
updated_at	TIMESTAMP	Дата останнього оновлення

Таблиця portfolio_items (табл.2.4) використовується для зберігання всіх робіт фотографа, які відображаються на публічній частині платформи. Вона дозволяє

структурувати портфоліо за типами зйомок та забезпечує зручну навігацію для клієнтів.

Таблиця 2.4

Опис таблиці portfolio_items

Поле	Тип	Призначення
id	BIGINT	Первинний ключ
photoshoot_type_id	BIGINT	Тип зйомки (вторинний ключ)
title	VARCHAR(255)	Назва проєкту
image_path	VARCHAR(255)	Шлях до картинки
order	int	Порядковий номер
is_featured	bool	Показник доступності
created_at	timestamp	Дата створення
updated_at	timestamp	Дата оновлення

Таблиця photoshoot_types (табл.2.5) виконує роль довідника типів зйомок і використовується для систематизації портфоліо та брифів. Вона дозволяє стандартизувати категорії робіт і забезпечує єдину структуру класифікації контенту.

Завдяки цій таблиці система отримує можливість масштабувати перелік напрямків зйомки без змін у коді, що підвищує гнучкість платформи..

Таблиця 2.5

Опис таблиці photoshoot_types

Поле	Тип	Призначення
id	BIGINT	Первинний ключ
title	VARCHAR(255)	Назва типу зйомки
slug	VARCHAR(255)	SEO-ідентифікатор для url
description	TEXT	Роль користувача в організації
Image_path	VARCHAR(255)	Шлях до картинки на сервері
order	int	Порядковий номер замовлення
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата оновлення

Таблиця briefs (табл. 2.6) призначена для зберігання технічних завдань, які клієнти заповнюють перед замовленням зйомки. Вона є одним із ключових елементів взаємодії між користувачем і фотографом. Використання JSON-поля дозволяє зберігати гнучку структуру відповідей, що особливо важливо для різних типів зйомок із унікальними наборами питань.

Таблиця 2.6

Опис таблиці briefs

Поле	Тип	Призначення
id	BIGINT	Ідентифікатор ролі
name	varchar(255)	Користувач
phone	varchar(255)	Номер телефону користувача
message_platform	varchar(255)	Платформа для контакту з користувачем
message_handle	varchar(255)	Ідентифікатор користувача на обраній платформі для зв'язку
photoshoot_type_id	BIGINT	Тип обраної фотосесії
photoshoot_type_other	varchar(255)	Дод. інформація, якщо є інший вид фотосесії
desired_date	date	Дата проведення фотосесії
additional_services	json	Додаткові послуги
references_link	text	Посилання на референс
details	text	Деталі до замовлення
read_at	TIMESTAMP	Час, коли бриф отриманий
created_at	TIMESTAMP	Дата створення
updated_at	TIMESTAMP	Оновлення

Таблиця editing_services (табл. 2.7) містить перелік доступних послуг постпродакшну. Вона дозволяє адміністратору керувати пропозицією сервісів, які доступні клієнтам.

Таблиця 2.7

Опис таблиці editing_services

Поле	Тип	Призначення
id	BIGINT UNSIGNED	Первинний ключ
title	VARCHAR(255)	Назва послуги
description	TEXT	Опис послуги
price_info	VARCHAR(255)	Інформація про вартість
order	INT	Замовлення
created_at	TIMESTAMP	Створення
updated_at	TIMESTAMP	Оновлення

Таблиця editing_orders (табл. 2.8) фіксує всі замовлення на постпродакшн, які створюють клієнти. Вона є центральною частиною процесу взаємодії між користувачем і виконавцем.

Через цю таблицю реалізується відстеження життєвого циклу замовлення, включаючи статуси виконання та прив'язку до файлів.

Таблиця 2.8

Опис таблиці editing_orders

Поле	Тип	Призначення
id	BIGINT UNSIGNED	Первинний ключ
client_name	VARCHAR(255)	Ім'я користувача
email	VARCHAR(255)	Електронна адреса клієнта
phone	VARCHAR(255)	Номер телефону клієнта
description	TEXT	Опис замовлення
client_drive_link	TEXT	Посилання на гугл диск клієнта
admin_drive_link	TEXT	Посилання на гугл диск в системі адміністратора
status	VARCHAR(255)	Статус замовлення
created_at	TIMESTAMP	Дата створення
updated_at	TIMESTAMP	Дата редагування

Таблиця posts (табл. 2.9) використовується для ведення блогу фотографа, який виконує роль маркетингового та інформаційного інструменту. Вона дозволяє публікувати статті, кейси та новини. Завдяки структурі з підтримкою slug реалізується SEO-оптимізація сторінок блогу.

Таблиця 2.9

Опис таблиці posts

Поле	Тип	Призначення
id	int	Ідентифікатор поста
title	varchar(255)	Заголовок статті
slug	varchar(255)	Унікальний url сторінки
content	text	Текст статті
image_path	varchar(255)	Шлях головної
category	varchar(255)	Категорія до якої належить стаття
published_at	timestamp	Дата публікації
created_at	TIMESTAMP	Дата створення
updated_at	TIMESTAMP	Дата зміни

Таблиця settings (табл. 2.10) є універсальним сховищем конфігураційних параметрів системи та використовується для централізованого керування вмістом вебсайту. Завдяки зберіганню налаштувань у базі даних адміністратор може змінювати текстову інформацію, контактні дані, SEO-параметри та інші елементи інтерфейсу без необхідності внесення змін до програмного коду або повторного розгортання застосунку. Такий підхід підвищує гнучкість системи та спрощує її підтримку.

У таблиці зберігаються дані для різних функціональних розділів вебсайту. Зокрема, для головної сторінки містяться параметри блоку привітання (заголовок та підзаголовок), для сторінки «Про мене» інформація про фотографа та опис професійної діяльності, а для сторінки контактів електронна пошта, номер телефону, місцезнаходження та посилання на соціальні мережі. Крім того, таблиця використовується для зберігання SEO-опису сайту, що впливає на його індексацію пошуковими системами та покращує представлення ресурсу в результатах пошуку.

Окрему групу записів становлять системні налаштування інтеграцій із зовнішніми сервісами. До них належать параметри взаємодії зі службами штучного інтелекту, зокрема вибір моделі та інструкцій для обробки тексту, а також реквізити інтеграції з Google Drive для автоматизованого зберігання та керування файлами. Використання універсальної структури «ключ–значення» дозволяє легко додавати нові параметри без зміни структури бази даних, що забезпечує масштабованість і подальший розвиток системи.

Таблиця 2.10

Опис таблиці settings

Поле	Тип	Призначення
id	uuid	Ідентифікатор параметра
key	varchar	Ключ параметра для зберігання
value	varchar	Значення
type	text	Тип параметра
group	text	До якої групи (сторінки) належить
created_at	TIMESTAMP	Дата створення
updated_at	TIMESTAMP	Дата зміни

На рис. 2.2 наведено загальну діаграму бази даних інформаційної системи керування портфоліо фотографа. Діаграма відображає структуру основних сутностей системи, їх атрибути та взаємозв'язки, що забезпечують зберігання контенту сайту, обробку замовлень на фотозйомку, управління послугами та взаємодію з користувачами. Побудова бази даних виконана відповідно до принципів нормалізації, що дозволяє уникнути дублювання даних, забезпечити їх цілісність та спростити подальший супровід системи.



Рис. 2.2. Діаграма бази даних

Центральне місце у структурі займають таблиці, пов'язані з формуванням контенту вебсайту. До них належать таблиці posts, portfolio_items, photoshoot_types та settings. Таблиця posts містить інформацію для ведення блогу, включаючи текстові матеріали, категорії та зображення. Таблиця portfolio_items використовується для зберігання фотографій портфоліо та їх групування за типами фотозйомок через зв'язок із таблицею photoshoot_types. Таблиця settings забезпечує

централізоване зберігання налаштувань і текстового наповнення сайту, що дозволяє змінювати контент без внесення змін до програмного коду.

Окремий блок сутностей реалізує функціонал взаємодії з клієнтами. Таблиця `briefs` призначена для зберігання анкет замовників, які містять інформацію про бажаний тип фотозйомки, контактні дані, додаткові послуги та інші побажання. Для обліку замовлень на ретушування фотографій використовується таблиця `editing_orders`, у якій зберігаються дані клієнта, посилання на вихідні матеріали, адміністративні примітки та статус виконання замовлення. Перелік доступних послуг ретушування формується за допомогою таблиці `editing_services`, що дозволяє гнучко керувати переліком сервісів через адміністративну панель.

Для забезпечення роботи системи автентифікації та керування доступом використовується таблиця `users`, яка містить дані зареєстрованих користувачів, їх ролі, облікові записи та параметри двофакторної автентифікації. Додатково функціонують службові таблиці `sessions` та `password_reset_tokens`, що відповідають за підтримку активних сеансів користувачів і механізм відновлення доступу до облікового запису.

Крім основних бізнес-сутностей, база даних містить низку системних таблиць, необхідних для стабільної роботи застосунку. До них належать `jobs`, `job_batches` та `failed_jobs`, які забезпечують виконання фонових задач, `cache` і `cache_locks` для кешування даних та підвищення продуктивності системи, а також таблиця `migrations`, що використовується для контролю версій структури бази даних. Така архітектура дозволяє ефективно поєднати функціональність вебсайту-портфоліо, систему прийому замовлень та адміністративні інструменти в межах єдиної інформаційної системи.

2.3. Проектування та реалізація алгоритмів роботи системи

У системі керування портфоліо фотографа реалізовано алгоритм автоматизованої інтеграції з сервісом Google Drive, призначений для організації процесу приймання та обробки файлів клієнтів. Основною метою алгоритму є автоматичне створення структури каталогів для нового замовлення, передача

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

вихідних матеріалів ретушери та забезпечення централізованого зберігання результатів роботи. Використання хмарного сховища дозволяє мінімізувати навантаження на сервер застосунку та забезпечити зручний доступ до матеріалів з будь-якого пристрою.

Робота алгоритму починається в момент створення клієнтом нового замовлення на ретушування фотографій. Після надсилання форми система не виконує завантаження файлів безпосередньо під час HTTP-запиту, оскільки великі обсяги графічних матеріалів можуть суттєво збільшити час очікування відповіді сервера. Замість цього завантажені файли тимчасово зберігаються у локальному приватному сховищі сервера, а для подальшої обробки створюється спеціальна фонові задача. Такий підхід дозволяє миттєво повернути користувачу повідомлення про успішне створення замовлення та продовжити обробку даних незалежно від активності клієнта[9].

Під час створення нового замовлення контролер формує запис у таблиці `editing_orders`, після чого всі завантажені файли зберігаються у каталозі тимчасового сховища за допомогою файлової абстракції `Laravel Storage`. Для кожного файлу генерується унікальне ім'я, що унеможливорює конфлікти між файлами різних користувачів. Після успішного збереження формується масив метаданих, який містить шлях до файлу, його початкову назву, та службову інформацію, необхідну для подальшої передачі в чергу обробки.

Для виконання тривалих операцій використовується механізм `Laravel Queues`. Задача `ProcessEditingOrder` реалізує інтерфейс `ShouldQueue`, завдяки чому автоматично передається до системи черг. У робочому середовищі як драйвер черги може використовуватися `Redis` або база даних. Виконання задач здійснюється окремими воркерами, які працюють незалежно від вебсерверу та постійно відстежують появу нових записів у черзі. Це дозволяє масштабувати систему шляхом запуску декількох паралельних воркерів без змін у програмному коді.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

Важливою складовою алгоритму є механізм автоматичного оновлення токенів доступу до Google API. Після створення об'єкта сервісу виконується перевірка валідності поточного access token шляхом аналізу часової мітки його завершення. Якщо термін дії токена минув, система використовує збережений refresh token для отримання нового access token через OAuth 2.0 Authorization Server. Після отримання нових облікових даних JSON-структура токена серіалізується та оновлюється в таблиці settings, що дозволяє уникнути повторної авторизації адміністратора навіть після тривалої роботи системи.

Створення структури каталогів виконується через послідовність викликів методу files.create Google Drive API. Для кожної папки формується об'єкт метаданих із MIME-типом application/vnd.google-apps.folder. Після отримання ідентифікатора головної папки система створює підпапки Input та Result, передаючи ідентифікатор батьківської директорії через параметр parents. Таким чином формується ієрархічна структура, яка повністю відповідає логіці бізнес-процесу обробки замовлень.

Особливу увагу приділено обробці посилань на сторонні каталоги Google Drive. Для виділення ідентифікатора папки використовується регулярний вираз, що дозволяє коректно обробляти різні формати посилань, які генеруються сервісом Google Drive. Після отримання ідентифікатора система створює об'єкт типу application/vnd.google-apps.shortcut, у властивості shortcutDetails.targetId якого записується знайдений ідентифікатор каталогу клієнта. В результаті створюється не копія папки, а спеціальний ярлик, який займає мінімальний обсяг дискового простору та забезпечує миттєвий доступ до оригінальних матеріалів.

Завантаження файлів до Google Drive реалізоване через режим multipart upload, який дозволяє передавати в одному HTTP-запиті як метадані об'єкта, так і його вміст. Перед початком завантаження система визначає MIME-тип файлу та формує відповідні параметри запиту. Отримані від Google Drive ідентифікатори файлів можуть бути використані для подальшого керування ресурсами, зміни прав доступу або автоматизації додаткових процесів обробки.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Для забезпечення надійності алгоритму реалізовано комплексну систему обробки винятків. Усі критичні операції взаємодії з Google API виконуються в конструкціях try-catch, що дозволяє перехоплювати помилки мережевого з'єднання, проблеми авторизації або перевищення квот сервісу. У разі виникнення винятку інформація записується до журналу логування Laravel, а задача автоматично повертається до черги для повторного виконання відповідно до налаштованої політики повторних спроб[6].

Після завершення всіх операцій виконується фіналізація процесу. Задача оновлює запис замовлення в таблиці editing_orders, записуючи URL створеної папки Google Drive у поле admin_drive_link. Далі у блоці finally здійснюється гарантоване очищення тимчасового сховища за допомогою файлової системи Laravel. Незалежно від результату виконання задачі всі локальні копії файлів видаляються, що запобігає перевитраті дискового простору та забезпечує стабільну роботу серверної інфраструктури навіть при обробці великої кількості замовлень.

Розглянемо наступний алгоритм це мульти-модельний AI Pipeline, який реалізує концепцію єдиного інтелектуального шлюзу між клієнтським інтерфейсом і зовнішніми AI-сервісами. У цій архітектурі Laravel-сервер виступає як центральний координатор (Gateway), що приймає запити від React-клієнта, виконує їхню попередню обробку, забезпечує безпеку та маршрутизацію, а також передає дані до відповідних GPU-орієнтованих моделей. Такий підхід дозволяє ізолювати клієнтську частину від складнощів роботи з різними API та уніфікує взаємодію з усіма AI-сервісами.

Першим ключовим етапом є нормалізація вхідних даних, зокрема зображень у форматі Base64. Оскільки браузер передає файли разом із MIME-префіксом (наприклад, data:image/png;base64,...), система виконує очищення цього рядка через алгоритм розділення, залишаючи лише чисте Base64-представлення. Це гарантує сумісність із зовнішніми моделями, які очікують строго структуровані вхідні дані та можуть повертати помилки при наявності зайвих символів або метаданих.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Окремим підпроцесом реалізовано інтелектуальне маскування, яке використовується в задачах інпейнтингу та видалення об'єктів. У цьому випадку фронтенд формує два синхронізовані шари даних: оригінальне зображення та маску, де користувач вручну позначає області для редагування. Бекенд перевіряє наявність маски та формує відповідний payload, передаючи оригінал як основне зображення, а маску як окремий параметр. Після цього AI-модель аналізує навколишні пікселі та генерує правдоподібне заповнення вибраної області, відновлюючи текстури сцени.

Наступним важливим компонентом є адаптивна маршрутизація параметрів, яка забезпечує сумісність із різними AI-моделями, що мають відмінні API-специфікації. Наприклад, для моделей апскейлу використовується інший ключ передавання зображення, ніж для стандартних запитів, тоді як моделі релайтингу підтримують додаткові параметри освітлення, що впливають на результат рендерингу. Таким чином, контролер виконує роль “перекладача” між єдиним внутрішнім форматом даних і різними зовнішніми API.

Завершальним етапом є обробка бінарних потоків, які повертаються від AI-сервісів. Отримані дані у сирому вигляді конвертуються в Base64, після чого формуються у Data URI для зручної передачі на фронтенд. Це дозволяє реалізувати безперервний цикл редагування без обов'язкового збереження кожного проміжного результату в базі даних. Така модель роботи підтримує принцип non-destructive editing, коли користувач може послідовно застосовувати різні AI-операції, не втрачаючи якості вихідного зображення до моменту фінального збереження.

У результаті система об'єднує кілька спеціалізованих AI-інструментів а саме видалення фону, апскейлінг, релайтинг, інпейнтинг та розмиття заднього плану в єдиний керований пайплайн. Це забезпечує гнучкість, масштабованість і можливість легко додавати нові моделі без зміни основної архітектури.

Третій алгоритм це контекстне переписування тексту, який інтегрований безпосередньо в редактор контенту та дозволяє виконувати миттєву

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

трансформацію тексту без переходу між інструментами. Його основна ідея полягає в тому, що користувач може виділити фрагмент у редакторі TinyMCE та одразу отримати покращену версію тексту, змінену за стилем, тоном або граматично виправлену[26]. Це створює ефект вбудованого AI-редактора, який працює як частина інтерфейсу, а не як окремий сервіс.

З технічної точки зору, алгоритм базується на використанні високошвидкісного движка Groq у поєднанні з моделлю Llama 3[13]. Завдяки апаратній оптимізації на рівні Language Processing Units, система забезпечує наднизьку затримку відповіді - у межах 0.1–0.5 секунди. Це дозволяє реалізувати майже реального часу взаємодію, де користувач отримує результат переписування тексту без відчутної затримки та без необхідності ручного копіювання чи вставки.

Важливою частиною алгоритму є використання системних інструкцій, які зберігаються у базі даних і визначають поведінку моделі. Зокрема, використовується жорсткий system prompt. Такий підхід гарантує детерміновану поведінку моделі, оскільки вона повертає виключно фінальний текст без додаткових пояснень або коментарів. Це критично для інтеграції з редактором, оскільки результат одразу підміняє виділений фрагмент у DOM.

Окремим компонентом реалізовано систему історії змін та можливість відкату. У фронтенд-частині використовується локальний стан groqHistory, який зберігає всі версії тексту після кожної ітерації переписування. Це дозволяє користувачу переглядати попередні варіанти тексту та повертатися до них у разі, якщо нова версія виявилась менш вдалою. Таким чином, AI-редагування стає контрольованим і безпечним процесом, а не одноразовою дією без можливості відновлення.

Також у системі реалізовано динамічний вибір моделі, який дозволяє адаптувати поведінку AI під складність задачі. Користувач може перемикатися між більш потужними моделями, такими як Llama 3.3 70B для складних стилістичних або смислових трансформацій, і легшими варіантами на кшталт Llama 3.1 8B для

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

швидких та простих правок. Це забезпечує баланс між якістю результату та швидкодією системи залежно від сценарію використання.

Четвертий алгоритм це візуальне оновлення статті, який відповідає за автоматичне створення зображень безпосередньо в процесі написання контенту. Його ключова мета полягає в тому, щоб користувач міг миттєво генерувати унікальні ілюстрації або обкладинки для статей на основі текстового опису, не виходячи з редактора. Таким чином, система об'єднує процес написання тексту та створення візуального контенту в єдиний креативну послідовність.

З технічної точки зору алгоритм реалізований через інтеграцію з API сервісу Pollinations AI у межах PollinationsController. Система формує динамічний HTTP-запит у вигляді URL, куди передаються ключові параметри генерації: текстовий prompt, розміри зображення. Такий підхід дозволяє генерувати зображення без складної серверної обробки, делегуючи основні обчислення зовнішньому AI-сервісу[18].

Важливим елементом алгоритму є механізм Random Seed Persistence, який забезпечує контрольовану випадковість генерації. Навіть якщо текстовий промпт залишається незмінним, зміна seed призводить до створення нового варіанту зображення. Це дозволяє реалізувати функцію згенерувати ще раз, яка дає користувачу можливість отримувати різні візуальні варіації однієї і тієї ж ідеї без необхідності змінювати опис.

Після генерації зображення запускається автоматизований процес збереження. Отримані бінарні дані передаються на бекенд, де формується унікальне ім'я файлу за допомогою Str::random(). Далі файл зберігається у файловій системі через Storage::disk('public'), а його шлях повертається на фронтенд. Після цього зображення автоматично підставляється у відповідне поле статті, що повністю усуває необхідність ручного завантаження або обробки.

Додатково реалізовано механізм Garbage Collection, який відповідає за очищення непотрібних файлів. Якщо користувач змінює згенеровану ілюстрацію або видаляє чернетку статті, система автоматично видаляє відповідний файл із

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

сервера. Це дозволяє уникнути накопичення зайвих даних і підтримує оптимальне використання дискового простору при активній роботі з генерацією зображень.

Висновки до другого розділу

У цьому розділі було виконано комплексне проєктування та реалізацію програмного забезпечення для вебплатформи робочого місця фотографа з інтеграцією інструментів штучного інтелекту. На основі попереднього аналізу предметної області сформовано загальну архітектуру системи, визначено основні модулі та їхню взаємодію, а також спроектовано структуру бази даних, яка забезпечує коректне зберігання та обробку всіх типів даних, що використовуються в межах платформи. Окрему увагу приділено реалізації ключових алгоритмів, які автоматизують типові бізнес-процеси та зменшують ручне навантаження на користувача.

Розроблена система побудована на модульній архітектурі з чітким розмежуванням ролей та прав доступу, що дозволяє розділити функціональність клієнтської та адміністративної частин. Використання сучасних технологій для фронтенду забезпечило створення зручного та адаптивного інтерфейсу, який підтримує повний цикл роботи фотографа — від взаємодії з портфоліо та замовленнями до керування контентом і постпродакшном. При цьому логіка взаємодії користувача з системою була спроектована таким чином, щоб максимально спростити основні операції та зробити їх інтуїтивними.

База даних, побудована на основі PostgreSQL, має нормалізовану структуру та охоплює всі ключові сутності системи, включаючи користувачів, замовлення, послуги, портфоліо, блогові матеріали та системні налаштування. Такий підхід забезпечує цілісність даних, зменшує надлишковість інформації та створює умови для подальшого масштабування системи без необхідності суттєвих змін у її структурі.

Окремим важливим результатом стало впровадження алгоритмів автоматизації, зокрема фонові синхронізації з Google Drive, що дозволяє ефективно працювати з великими обсягами медіафайлів без впливу на швидкодію

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу. Також реалізовано мультимодельний AI-пайплайн, який виступає єдиним інтелектуальним шлюзом для взаємодії з різними нейромережами, забезпечуючи нормалізацію даних і підтримку гнучких сценаріїв обробки зображень у режимі non-destructive editing.

У підсумку можна зазначити, що реалізовані архітектурні та алгоритмічні рішення повністю відповідають поставленим вимогам до системи. Розроблена платформа є цілісним, продуктивним і масштабованим рішенням, яке поєднує класичні підходи до побудови вебсистем із сучасними можливостями штучного інтелекту, тим самим суттєво підвищуючи ефективність роботи фотографа та якість взаємодії з клієнтами.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1. Порядок встановлення та налаштування параметрів системи

Розроблена вебплатформа для автоматизації робочого місця фотографа реалізована як сучасний вебзастосунок на базі фреймворку Laravel версії 12 для серверної частини (back-end) та інструментарію Inertia.js версії 2 у зв'язці з бібліотекою React версії 19 для клієнтської частини (front-end). Система використовує архітектуру моноліту з чітким розділенням відповідальності між шарами, де back-end повністю керує складною бізнес-логікою, безпекою, взаємодією з реляційними даними та інтеграціями зі штучним інтелектом, а front-end забезпечує швидкий інтерактивний інтерфейс.

Перед початком розгортання та ініціалізації інфраструктури проєкту необхідно забезпечити наявність та коректне функціонування базового системного програмного забезпечення на локальній машині розробника або цільовому сервері. До переліку обов'язкових компонентів середовища виконання входить інтерпретатор мови PHP версії 8.4 або новішої, укомплектований специфічними розширеннями для роботи з рядками, шифруванням та базами даних, а також інструмент керування пакетами Composer. Клієнтська інфраструктура потребує розгортання серверної JavaScript-платформи Node.js версії 22 або новішої разом із пакетним менеджером NPM, тоді як ізоляція, оркестрація та керування базою даних реалізується через середовище Docker та інструмент Docker Compose [7]. Крім того, для завантаження, автентифікації та подальшого супроводу вихідного коду платформи в обов'язковому порядку використовується розподілена система контролю версій Git.

Після повної підготовки та перевірки системного середовища виконується безпосереднє клонування сховища проєкту з віддаленого репозиторію на GitHub у локальну робочу директорію. Отримана структура каталогу повністю відповідає загальноприйнятим стандартам фреймворку Laravel, де основна бізнес-логіка, контролери, сервіси взаємодії з API та моделі даних розміщені в директорії застосунку, а реактивні компоненти front-end частини на базі React локалізовані

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

всередині каталогу ресурсів. Для надійного збереження та обробки інформації вебплатформа використовує об'єктно-реляційну систему керування базами даних PostgreSQL, розгортання та первинна конфігурація якої здійснюється у повністю ізольованому контейнері. Запуск відповідного інфраструктурного файлу docker-compose.yml у фоновому режимі дозволяє автоматично завантажити офіційний образ СУБД, підняти мережевий контейнер та налаштувати його на прослуховування стандартного системного порту.

Наступним важливим кроком після успішної ініціалізації бази даних є створення та налаштування файлу локального оточення системи, що реалізується шляхом копіювання базового шаблону конфігурації у робочий файл .env. У цьому файлі розробник детально вказує параметри мережевого підключення до створеного контейнера PostgreSQL, включаючи тип драйвера, адресу хоста, порт, унікальну назву бази даних, а також ідентифікаційні дані адміністратора у вигляді імені користувача та пароля. Процес первинного налаштування середовища завершується генерацією унікального шифрувального ключа застосунку, який записується в конфігурацію і надалі використовується для захисту призначених для користувача сесій та безпечного збереження токенів.

Наступним технологічним етапом розгортання вебплатформи є встановлення та збірка всіх необхідних зовнішніх залежностей як для серверної, так і для клієнтської архітурних частин проєкту. Шляхом виконання консольної команди composer install здійснюється автоматичне сканування конфігураційного маніфесту та завантаження ізольованих PHP-пакетів, серед яких присутні спеціалізовані модулі SDK для автентифікації та взаємодії з хмарним сховищем Google Drive через API, а також клієнтські бібліотеки для надсилання запитів до інтегрованих платформ штучного інтелекту. Одночасно з цим на стороні клієнтського інтерфейсу викликається пакетний менеджер за допомогою команди npm install, яка розгортає в середовищі проєкту базові компоненти React, утилітарний CSS-фреймворк Tailwind CSS та архітектурний міст Inertia.js. Оскільки ключовий функціонал розробленого робочого місця фотографа передбачає глибоку

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

інтеграцію зі сторонніми інтелектуальними сервісами, до робочого конфігураційного файлу оточення обов'язково вносяться приватні сервісні ключі доступу до Google Drive API, а також персональні токени та API-ключі для платформ Groq API на базі моделі Llama 3 та Pollinations AI. Завершальним кроком підготовки даних є запуск процедури міграції бази даних із застосуванням механізму початкового наповнення за допомогою команди `php artisan migrate --seed`, що дозволяє автоматично згенерувати структуру реляційних таблиць на стороні PostgreSQL та наповнити їх базовими системними даними, включаючи доступні жанри фотозйомок, початкові статуси замовлень та глобальні налаштування системи.

Для повноцінного запуску, моніторингу та коректного функціонування всіх модулів вебплатформи в режимі розробки необхідно паралельно ініціювати три незалежні системні процеси. Першим здійснюється старт вбудованого вебсервера додатку за допомогою команди `php artisan serve`, який відкриває мережевий порт і забезпечує прямий доступ до інтерфейсу платформи за локальною адресою в браузері.

Другим процесом виступає інструмент збірки та компіляції фронтенду, який активується командою `npm run dev` і працює в інтерактивному режимі гарячої заміни модулів, забезпечуючи миттєву перезбірку та відображення будь-яких змін у дизайні чи логіці React-компонентів без повного перезавантаження сторінок.

Третім, і найбільш критичним з точки зору архітектури процесом, є запуск фонового обробника черг за допомогою консольної команди `php artisan queue:listen`. Цей фоновий воркер бере на себе виконання асинхронних завдань, ізольовано взаємодіє із зовнішнім хмарним сховищем для передачі великих масивів фотографій, а також очікує відповіді від віддалених нейромереж при генерації текстів та обкладинок. Винесення цих операцій у черги дозволяє back-end серверу миттєво відповідати користувачу, повністю усуваючи проблему блокування або тривалого зависання клієнтського інтерфейсу. Після успішного та безпомилкового

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

старту всіх описаних процесів платформа вважається повністю готовою до експлуатації.

3.2. Структура інтерфейсу

В цьому розділі розглянемо інтерфейс створеної програми з точки зору кінцевих користувачів, щоб оцінити його зручність, логіку побудови та відповідність сучасним вимогам до вебзастосунків. Основна увага приділяється тому, наскільки інтуїтивно користувач може взаємодіяти з системою без додаткового навчання, як швидко він знаходить потрібні функції та наскільки ефективно реалізовано навігацію між основними модулями платформи. Окремо аналізується адаптивність інтерфейсу для різних типів пристроїв, стабільність роботи елементів взаємодії, а також загальна візуальна узгодженість компонентів, що формує цілісне сприйняття системи як єдиного робочого середовища.

Огляд почнемо з головної сторінки сайту (рис. 3.1). В верхній панелі ми бачимо доступні сторінки для користувачів. Відразу можна перейти на сторінки фотосесії, портфоліо, сторінки блогу та сторінку контактів.



Рис. 3.1. Головна сторінка

На сторінці фотосесій (рис. 3.2) представлено розділ з доступними видами фотосесій, організований у форматі візуальної сітки. Основний простір займають

великі блоки з тематичними зображеннями та відповідними заголовками, наприклад, сімейна фотосесія та весільна фотосесія. Під кожною назвою розміщено лаконічний опис формату зйомки, який розкриває її основні особливості, можливі локації проведення та складові послуги.

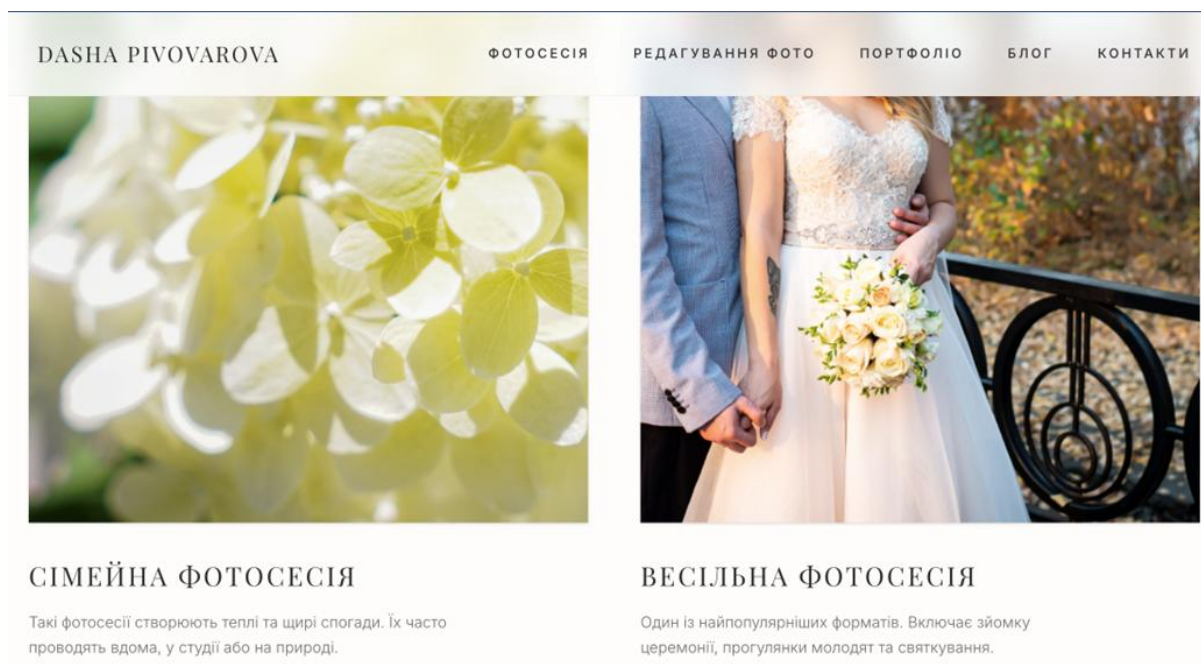


Рис. 3.2. Сторінка фотосесій

Визначившись із необхідним видом послуги, користувач може ініціювати процес замовлення. Передбачено, що після натискання на кнопку бронювання обраної фотосесії відбувається автоматичний перехід на окрему сторінку з детальним брифом. Там клієнт матиме змогу заповнити форму, описати свої побажання, вказати очікування від зйомки та надати всі необхідні організаційні деталі.

Сторінка замовлення фотосесії (рис. 3.3) реалізована у вигляді контактної форми де клієнту необхідно пройти процес оформлення замовлення. Вона починається з поля для введення імені та прізвища, після чого користувач за допомогою кнопок перемикавання обирає зручний спосіб комунікації (написати або зателефонувати) та відповідну платформу (Telegram, Viber, WhatsApp чи Email), а потім вводить самі контактні дані. Нижче розташований блок вибору виду зйомки, де клієнт вказує потрібну категорію із запропонованого переліку.

ДАША ПИВОВАРОВА

КОНТАКТИ & БРИФ

Давайте
створювати
разом

EMAIL

anthony.jonhsoncm@gmail.com

SOCIAL

INSTAGRAM

TELEGRAM

PINTEREST

LOCATION

Житомир, Україна

ФОТОСЕСІЯ

РЕДАГУВАННЯ ФОТО

ПОРТФОЛІО

БЛОГ

КОНТАКТИ

ІМ'Я ТА ПРИЗВИЩЕ *

Як вас звати?

ЗРУЧНИЙ ЗВ'ЯЗОК

НАПИСАТИ

ЗАТЕЛЕФОНУВАТИ

ПЛАТФОРМА

TELEGRAM

VIBER

WHATSAPP

EMAIL

КОНТАКТНІ ДАНІ *

@username або +380...

ВИБЕРІТЬ ВИД ЗЙОМКИ *

Сімейна фотосесія

Весільна фотосесія

Вулична фотосесія

Фотосесія для instagram

Інше

В кінці форми є інтерактивний календар, у якому користувач виділяє бажану дату проведення фотосесії. Під календарем реалізовано блок додаткових послуг, де за допомогою стандартних прапорців можна додати до замовлення роботу візажиста або оренду студії. Завершується бриф полями для введення тексту, однорядковим полем для додавання посилань на візуальні референси та текстовим блоком Деталі, де замовник має можливість вільно описати ідею та концепцію майбутньої зйомки.

Додатково на сайті реалізована функція замовлення редагування власних фотографій, яка дозволяє клієнту швидко сформулювати технічне завдання для постпродакшну (рис. 3.4). Для цього на сторінці редагування фото передбачено форму, що за структурою подібна до попередніх форм взаємодії в системі. У верхній частині розміщені поля для введення контактної інформації користувача: обов'язкові ім'я та email, а також додаткове поле для номера телефону, яке використовується для оперативного зв'язку у разі уточнення деталей замовлення. Такий підхід забезпечує базову ідентифікацію клієнта та створює канал комунікації між виконавцем і замовником.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

DASHA PIVOVAROVA
ФОТОСЕСІЯ РЕДАГУВАННЯ ФОТО ПОРТФОЛІО БЛОГ КОНТАКТИ

ЗАМОВИТИ РЕТУШ

Залиште запит на обробку

Опишіть ваші побажання, завантажте фото або надайте посилання на диск. Я підготую для вас найкращий результат.

ЯК ЦЕ ПРАЦЮЄ?

1. Ви надсилаєте фото та опис завдання
2. Я створюю робочу папку на Google Drive
3. Після виконання ви отримаєте посилання з результатом

ВАШЕ ІМ'Я *
Олександр

EMAIL *
example@mail.com

ТЕЛЕФОН (ОПЦІОНАЛЬНО)
+380...

ЩО САМЕ ПОТРІБНО ЗРОБИТИ? *
Ретуш шкіри, кольорокорекція, видалення зайвих об'єктів...

ЯК ПЕРЕДАТИ ФОТО?

ЗАВАНТАЖИТИ ФАЙЛ
ПОСИЛАННЯ НА GOOGLE DRIVE

Перетягніть фото сюди або натисніть для вибору
JPEG, PNG, WEBP, HEIC

НАДІСЛАТИ ЗАМОВЛЕННЯ

Рис. 3.4. Форма для редагування фото

Нижче розташоване велике текстове поле, призначене для детального опису технічного завдання, де користувач може вказати бажані зміни, зокрема кольорокорекцію, ретуш, видалення об'єктів або інші види обробки зображень. Під ним реалізовано блок завантаження матеріалів, у якому користувач обирає зручний спосіб передачі файлів за допомогою перемикачів. Доступні два варіанти: пряме завантаження через drag-and-drop зону, яка підтримує формати JPEG, PNG, WEBP та HEIC, або ж надання посилання на Google Drive для роботи з великими обсягами даних. Завершується форма великою кнопкою підтвердження Надіслати замовлення, яка ініціює відправку даних на сервер для подальшої обробки.

Також клієнтам доступна сторінка зі статтями блогу, де зібрано корисні матеріали на тематичні теми, пов'язані з фотографією, обробкою зображень та особливостями роботи сервісу (рис. 3.5). Тут користувач може ознайомитися з порадами щодо підготовки фотографій до редагування, рекомендаціями по зйомці, а також прикладами використання різних інструментів платформи. І водночас така сторінка дуже корисна для просування сайту в пошукових видачах.



Рис. 3.5. Сторінка Блогу

Тепер перейдемо безпосередньо до робочої області фотографа. На головній сторінці робочий простір поділено на ліву бічну панель навігації та основну центральну область. У бічному меню згруповано всі необхідні інструменти для адміністрування платформи: звідси користувач має доступ до дашборду, переліку заповнених брифів, замовлень на ретуш, налаштувань видів фотосесій, управління портфоліо та блогом. У нижній частині панелі навігації відображається профіль поточного авторизованого користувача (рис. 3.6).

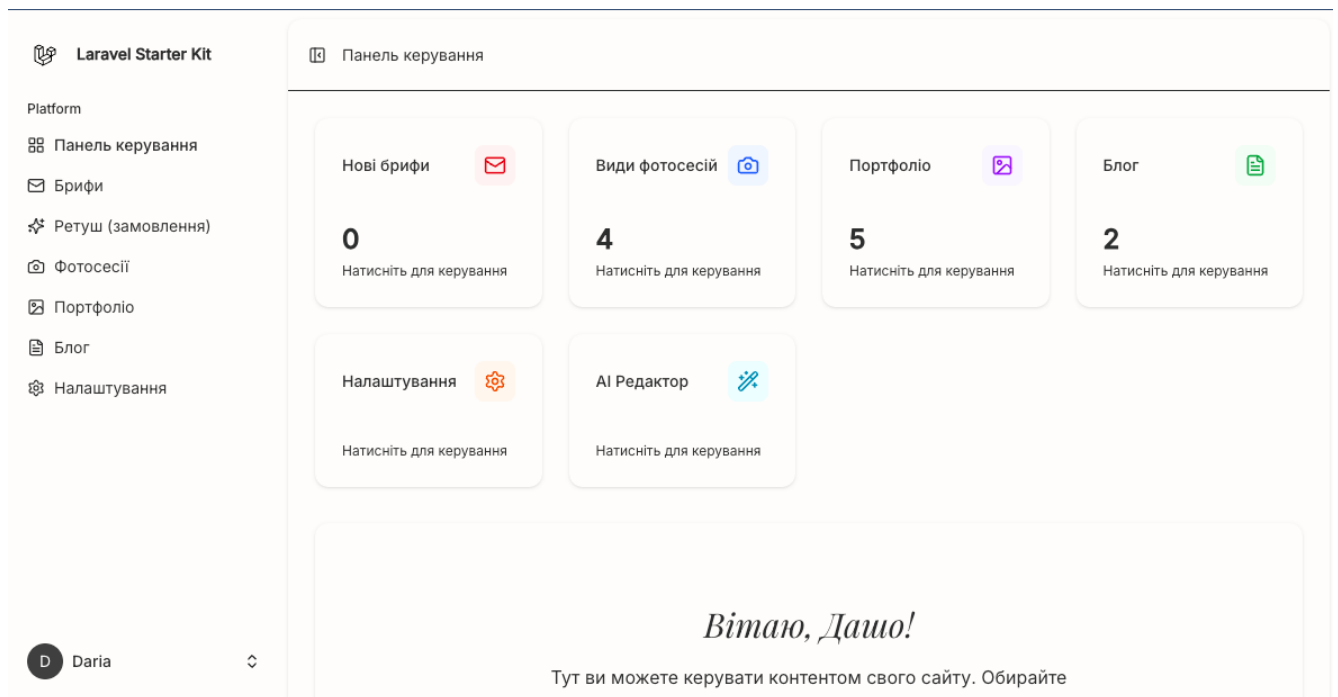


Рис. 3.6. Головна сторінка фотографа

Всі форми які заповнює клієнт для запису на фотосесію потрапляють на сторінку Брифи (рис. 3.7) Основна частина екрана відведена під таблицю, яка призначена для управління всіма заявками від клієнтів. Таблиця структурована за колонками і відображає ключову інформацію про кожне замовлення: ім'я замовника разом із його контактними даними для зв'язку, обраний вид зйомки, заплановану дату фотосесії та точний час створення самого запиту. У крайній правій колонці для кожного рядка передбачені інтерактивні іконки, які дозволяють адміністратору відкрити конкретний бриф для детального перегляду або повністю видалити його із системи.

Laravel Starter Kit Platform Панель керування Брифи Ретуш (замовлення) Фотосесії Портфоліо Блог Налаштування D Daria	Брифи				
	Вхідні брифи				
	Користувач	Вид зйомки	Дата зйомки	Дата запиту	Дії
	Дар'я Telegram: @Testname	Весільна Фотосесія	26 травня 2026	22 трав., 13:19	👁️ 🗑️
	Дар'я Telegram: @Testname	Сімейна Фотосесія	22 травня 2026	22 трав., 13:15	👁️ 🗑️
	Ігор Валерійович +380 (53) 534-44-32	Сімейна Фотосесія	16 червня 2026	12 трав., 15:18	👁️ 🗑️
	Олексій Telegram: +380 (22) 222-22-22	В Басейні	25 травня 2026	12 трав., 15:08	👁️ 🗑️
	Дар'я Telegram: +380 (98) 711-24-41	Вулична Фотосесія	15 травня 2026	12 трав., 15:06	👁️ 🗑️

Рис. 3.7. Брифи

Сторінку AI Редактор (рис. 3.8) в адміністративній панелі платформи поділено на дві основні частини: панель інструментів зліва та простору зону попереднього перегляду зображення справа. У верхній частині екрана розташований заголовок розділу та кнопка очищення редактора, яка дозволяє завантажити інше зображення для редагування. Ліва бічна панель, як і на інших сторінках, містить стандартне навігаційне меню для переходу між адміністративними розділами та профіль користувача.

Блок інструменти надає доступ до функцій автоматизованої обробки фотографії. Серед доступних опцій у вигляді випадаючих вкладок з видаленням та

розмиттям фону, покращення якості зображення, зміна освітлення, а також видалення небажаних об'єктів. На рис 3.8 в інтерфейсі відкрито інструмент для видалення фону, де користувач може обрати цільовий режим обробки через випадаючий список (в даному випадку обрано режим Людина) та застосувати дію за допомогою кнопки Виконати. У правій частині екрана на великому полотні відображається завантажена фотографія, яка слугує для візуального контролю процесу та відображення результатів застосованих змін..

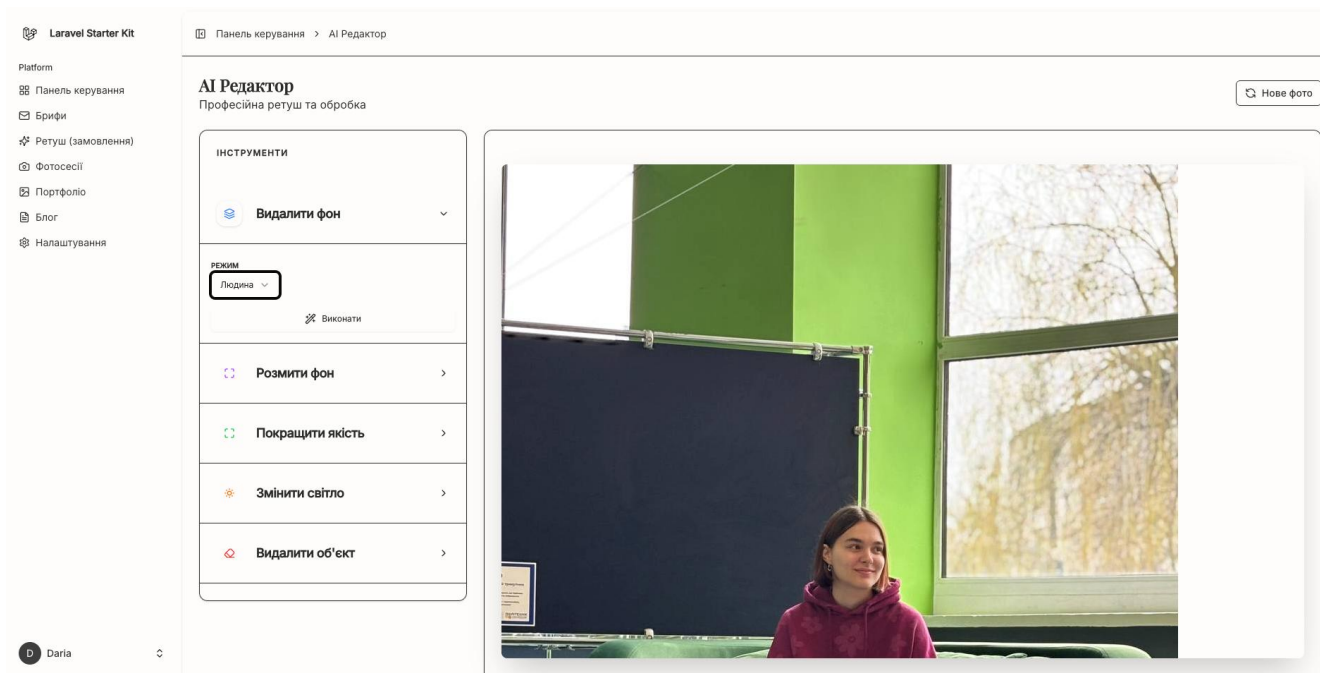


Рис. 3.8. AI редактор

Перед тим як перейти до розгляду функціоналу сторінки створення статей, варто зазначити, що цей модуль є однією з ключових складових контентної частини системи. Він дозволяє не лише вручну формувати публікації, але й суттєво спрощує процес їх підготовки завдяки інтеграції інструментів штучного інтелекту. У результаті користувач отримує можливість швидко створювати структурований та якісний текстовий контент, не витрачаючи зайвого часу на рутинне редагування та форматування. Саме тому дана сторінка відіграє важливу роль у загальній екосистемі платформи, поєднуючи класичний редакторський підхід із сучасними AI-можливостями.

Для генерування тексту використовується Groq API, у якому користувач має можливість обрати відповідну модель та задати промт для формування потрібного

результату (рис. 3.9). Такий підхід дозволяє гнучко керувати процесом генерації контенту залежно від складності задачі та бажаного стилю тексту. Після вибору всі параметри, включаючи обрану модель і введений промт, зберігаються в базі даних, що дає змогу повторно використовувати налаштування або відтворювати попередні результати генерації. Це забезпечує зручність роботи з AI-інструментом та підвищує ефективність створення контенту в межах системи..

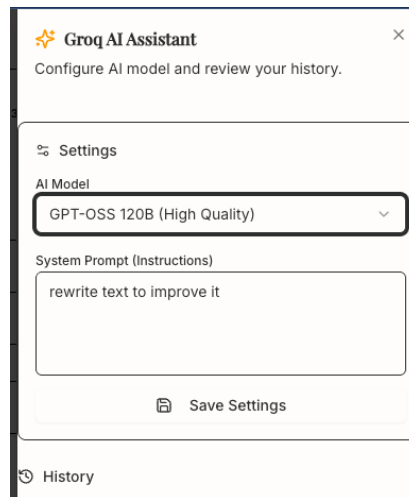


Рис. 3.9. Налаштування Groq

Для редагування тексту користувач може виділити потрібний фрагмент у редакторі, після чого натиснути праву кнопку миші. У результаті відкривається контекстне меню з доступними AI-інструментами, яке дозволяє одразу обрати спосіб обробки виділеного тексту без додаткових переходів між сторінками або сервісами (рис. 3.10). Це значно спрощує процес роботи з контентом і робить редагування більш інтуїтивним.

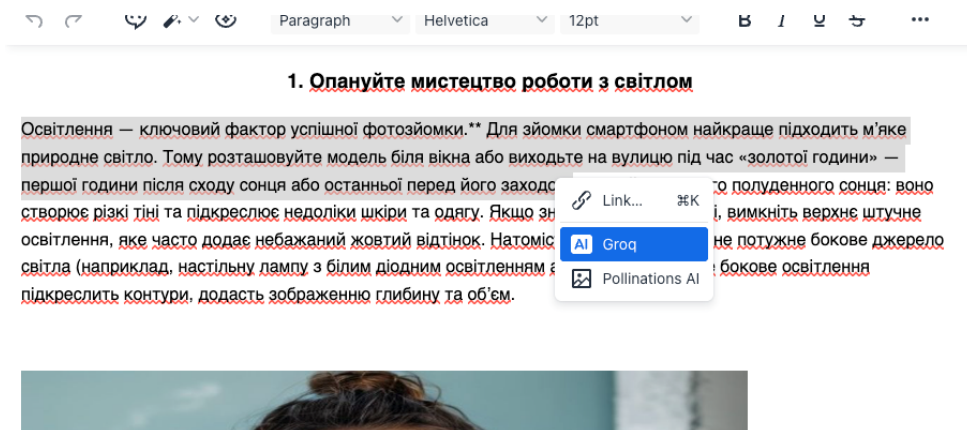


Рис. 3.10. Використання інструментів генерування

У цьому меню доступні два основні варіанти взаємодії зі штучним інтелектом. Перший — Grog, який використовується для генерації або покращення тексту: користувач обирає модель та задає промт, після чого система миттєво формує оновлений варіант тексту прямо в редакторі. Другий варіант — Pollinations AI, який відповідає за генерацію зображень; у цьому випадку необхідно задати промт та параметри розміру після чого сервіс створює відповідну ілюстрацію, що може бути вставлена у статтю. Таким чином, система об'єднує текстову та візуальну AI-генерацію в одному контекстному інтерфейсі.

На сторінці керування портфоліо, центральній частині сторінки відображено заголовок із коротким описом функціоналу, а у верхньому правому куті знаходиться кнопка швидкої дії «Додати фото» (рис. 3.10).

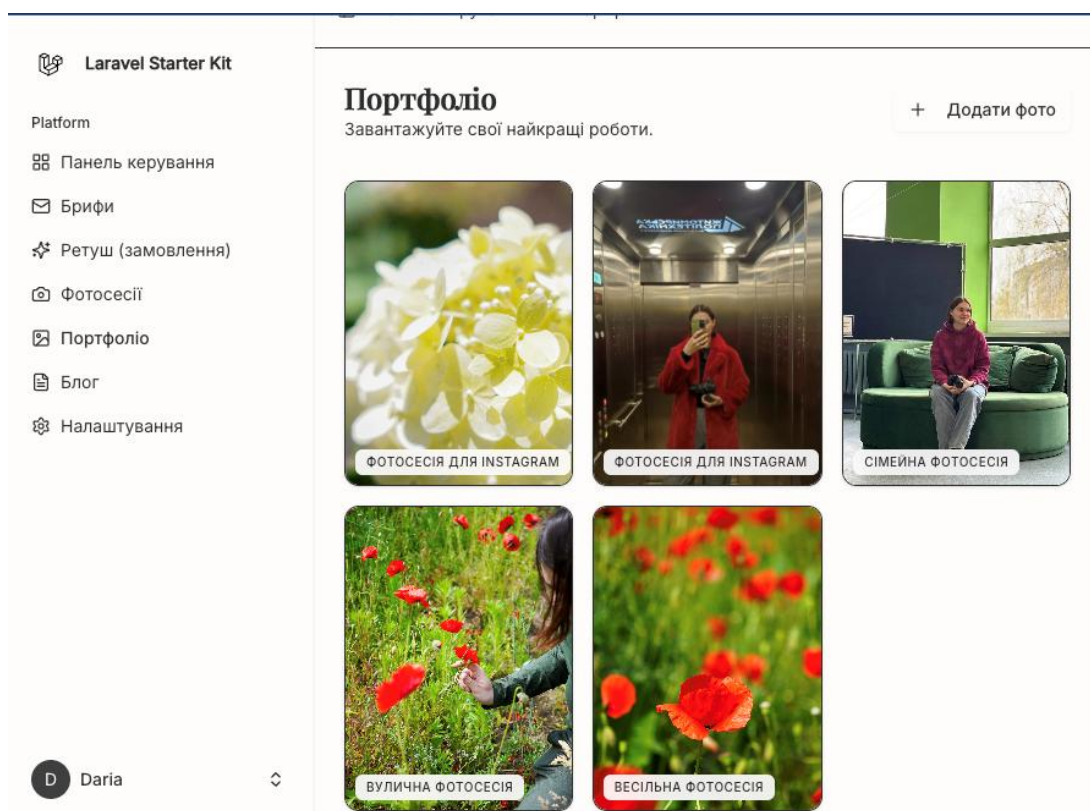


Рис. 3.11. Сторінка керування портфоліо

Основний контент представлений у вигляді візуальної сітки завантажених робіт. Кожна картка містить зображення та підпис із назвою відповідної категорії. Для додавання нового елемента до портфоліо користувач натискає кнопку створення, після чого обирає категорію зйомки та завантажує відповідний графічний файл у систему.

3.3. Тестування роботи програмного комплексу

Тестування роботи програмного комплексу є важливим етапом розробки, оскільки дозволяє перевірити коректність функціонування всіх його компонентів у реальних умовах використання та оцінити відповідність функціоналу поставленим вимогам. Саме на цьому етапі виявляються можливі помилки, недоліки інтерфейсу та логіки роботи системи, що безпосередньо впливають на якість і стабільність програмного продукту. У межах даного розділу тестування буде здійснюватися вручну, що дає змогу детально проаналізувати поведінку системи в різних сценаріях використання та з точки зору кінцевого користувача[1].

Модуль отримання замовлення на ретуш від клієнта є одним із найважливіших у системі, оскільки саме він відповідає за первинне отримання та обробку інформації від клієнта, що фактично є відправною точкою всього бізнес-процесу. Будь-яка помилка на цьому етапі може призвести до втрати даних, некоректного формування замовлення або порушення подальшої логіки обробки в чергах і сховищі, тому його стабільна робота є критично важливою для всієї системи.

Метою даного тестового сценарію є перевірка коректної взаємодії між клієнтською формою оформлення замовлення, механізмом черг Laravel та інтеграцією зі стороннім API Google Drive. Особливу увагу приділено перевірці асинхронної обробки даних, стабільності фонових процесів, а також правильності створення та структурування хмарних папок відповідно до отриманого замовлення. Даний сценарій дозволяє оцінити повний життєвий цикл обробки замовлення, починаючи від дії користувача і завершуючи результатом у хмарному сховищі.

У межах тестування було сформовано набір тестових кейсів, які охоплюють основні варіанти використання системи. Перший кейс перевіряє успішне створення замовлення через заповнення клієнтської форми з подальшою передачею даних у систему та постановкою задачі в чергу обробки. У результаті виконання даного сценарію система повинна згенерувати відповідне повідомлення про успіх, створити запис у базі даних та ініціювати Job ProcessEditingOrder у черзі.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Другий тестовий кейс спрямований на перевірку роботи фонові обробки замовлення та взаємодії з Google Drive API. Після запуску worker-процесу Laravel система повинна обробити задачу з черги та автоматично створити структуру папок у хмарному сховищі, яка включає основну директорію замовлення з унікальним ідентифікатором, а також підпапки для вхідних і результативних матеріалів. Додатково перевіряється коректність завантаження вихідних зображень у відповідний каталог Input (рис. 3.12).

Замовлення на ретуш					
ID	Клієнт	Статус	Дата	Google Drive	Дії
#15	Дар'я ✉ anthony.jonhsoncm@gmail.com ☎ +380987112441	Нове	2 черв. 2026, 13:16	Папка	👁️ 🗑️

Рис. 3.12. Успішне створення замовлення

Третій тестовий кейс стосується логіки роботи з Google Drive у випадку використання посилань замість фізичного завантаження файлів. У такому сценарії система повинна створювати ярлик (shortcut), який посилається на існуючу папку клієнта, забезпечуючи таким чином коректну навігацію між ресурсами без дублювання даних (рис. 3.13).

My Drive > siteorders > [Замовлення #15] - Дар'я ▾ 👤

Type ▾

People ▾

Modified ▾

Source ▾

Name 

Owner

 Input

 me

 Result

 me

Рис. 3.13. Успішне створення директорій в гугл диску

Четвертий тестовий кейс перевіряє систему валідації введених даних. Під час тестування аналізується поведінка системи при відправці порожньої форми, введенні некоректного формату електронної пошти, а також спробі завантаження заборонених типів файлів. Результатом має бути блокування відправки форми та відображення відповідних повідомлень про помилки валідації (рис. 3.14).

ФОТОСЕСІЯ РЕДАГУВАННЯ ФОТО ПОРТФОЛІО БЛОГ КОНТАКТИ

ВАШЕ ІМ'Я *
Олександр

EMAIL *
example@mail.com

ТЕЛЕФОН
+380...
! Заповніть це поле.

ЩО САМЕ ПОТРІБНО ЗРОБИТИ? *
Ретуш шкіри, кольорокорекція, видалення зайвих об'єктів...

ЯК ПЕРЕДАТИ ФОТО?
ЗАВАНТАЖИТИ ФАЙЛИ ПОСИЛАННЯ НА GOOGLE DRIVE

Перетягніть фото сюди або натисніть для вибору
JPEG, PNG, WEBP, HEIC

НАДІСЛАТИ ЗАМОВЛЕННЯ

3.14. Валідація форми

Опис процесу тестування базується на використанні методів чорної скриньки для перевірки клієнтської частини системи, а також інтеграційного тестування для аналізу взаємодії з Google Drive API. Під час перевірки фронтенду було підтверджено коректну передачу стану форми через Inertia.js та блокування повторної відправки запиту після ініціації процесу, що запобігає дублюванню замовлень. Тестування черг показало, що система забезпечує асинхронну обробку даних, завдяки чому клієнт отримує відповідь майже миттєво, тоді як робота з хмарним сховищем виконується у фоновому режимі.

Висновки до третього розділу

У даному розділі було розглянуто інтерфейс розробленого програмного комплексу та порядок його встановлення і налаштування, а також описано загальну

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

структуру взаємодії користувача із системою. Детально проаналізовано основні етапи розгортання платформи, включаючи підготовку програмного середовища, налаштування бази даних, встановлення залежностей та запуск ключових компонентів системи, що забезпечують її повноцінне функціонування.

Окрему увагу приділено опису клієнтської та адміністративної частин інтерфейсу, що дозволило продемонструвати логіку побудови навігації, взаємозв'язок між модулями та зручність використання системи для різних категорій користувачів. Було показано, як реалізовані основні функціональні сторінки, зокрема оформлення замовлень, керування контентом, робота з портфоліо, блогом та інструментами штучного інтелекту, що формує цілісну екосистему платформи.

Також у розділі було наведено підхід до тестування програмного комплексу, зокрема перевірку критично важливого модуля обробки замовлень, який забезпечує взаємодію між клієнтською формою, серверною логікою та зовнішніми сервісами. Результати тестування підтвердили коректність роботи основних функціональних компонентів, стабільність асинхронної обробки даних та правильність інтеграції з хмарним сховищем Google Drive.

Таким чином, можна зробити висновок, що розроблена система є функціонально завершеною, логічно структурованою та готовою до використання в справжніх умовах, забезпечуючи ефективну взаємодію між користувачем і всіма компонентами програмного комплексу.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено комплексний вебзастосунок для професійного фотографа, який об'єднує функціональність сайту-візитки, інтерактивного портфоліо та системи управління замовленнями. Система побудована на основі фреймворку Laravel, що забезпечує стабільну архітектуру, високу продуктивність та можливість подальшого масштабування.

Для адміністрування системи створено функціональну панель керування, яка забезпечує зручну роботу з вхідними заявками, контентом портфоліо та блогом. Важливим розширенням системи став модуль «AI Редактор», що дозволяє автоматизувати обробку зображень безпосередньо в межах платформи, зокрема виконувати видалення або розмиття фону та покращення якості фотографій.

Особливу увагу під час реалізації клієнтської частини було приділено створенню сучасного, мінімалістичного та зручного інтерфейсу. Розроблено структурований каталог послуг, багатокрокову форму оформлення брифів із можливістю вибору дати через календар та додавання додаткових параметрів замовлення. Окремо реалізовано модуль замовлення ретуші із підтримкою завантаження файлів, що значно спрощує процес збору вимог від клієнтів та переводить його в повністю цифровий формат.

Проведене тестування та огляд інтерфейсу підтвердило стабільну роботу всіх ключових компонентів системи, коректну обробку та збереження даних, безпечну роботу з файлами, а також ефективність інтегрованих AI-інструментів. Окремо відзначено адаптивність інтерфейсу та відповідність сучасним вимогам до зручності користувацької взаємодії.

В результаті, розроблений вебсайт є завершеним програмним продуктом, що повністю відповідає поставленим вимогам. Він ефективно виконує функцію представлення послуг фотографа, забезпечує залучення клієнтів та водночас оптимізує внутрішні бізнес-процеси завдяки впровадженню сучасних засобів для автоматизації.

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Види тестування та відмінності між ними. QAGroup. URL: <https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymu/> (дата звернення: 02.06.2026).
2. Нейромережі для написання текстів, крім ChatGPT. Gen.Tech. URL: <https://journal.gen.tech/post/5-neiromerezh-dlya-napysannya-tekstiv-krim-chatgpt> (дата звернення: 02.06.2026).
3. Топ-10 нейромереж для обробки фото. GoIT. URL: <https://goit.global/ua/articles/top-10-neyromerezh-dlia-obrobky-foto/> (дата звернення: 02.06.2026).
4. Штучний інтелект у фотографії: як технології змінюють процес зйомки та редагування. AI360. URL: <https://ai360.com.ua/shtuchnyy-intelekt-u-fotohrafii-yak-tekhnohii-zminiuiut-protsezyomky-ta-redahuvannia/> (дата звернення: 02.06.2026).
5. Codewithsjm. Laravel Routing: A Deep Dive into Laravel 12 Routing. Medium. URL: <https://medium.com/@codewithsjm/laravel-routing-a-deep-dive-into-laravel-12-routing-b955ae29d7ad> (дата звернення: 02.06.2026).
6. Design Patterns in PHP. Refactoring.Guru. URL: <https://refactoring.guru/design-patterns/php> (дата звернення: 02.06.2026).
7. Docker Documentation. URL: <https://docs.docker.com/> (дата звернення: 02.06.2026).
8. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler ; with contributions by D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford. Boston : Addison-Wesley Longman, 2003. 544 p.
9. Fowler, M.. Refactoring: Improving the Design of Existing Code. Boston: Addison-Wesley, 2018, 448 p.
10. Ganglani K. PostgreSQL vs MySQL 2026. Kunal Ganglani's Blog. 2026. URL: <https://www.kunalganglani.com/blog/postgresql-vs-mysql-2026> (дата звернення: 02.06.2026).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

11. Google Cloud Documentation. Google Cloud. URL: <https://docs.cloud.google.com/docs> (дата звернення: 02.06.2026).
12. Google Drive API. Google for Developers. URL: <https://developers.google.com/workspace/drive/api/guides/about-sdk> (дата звернення: 02.06.2026).
13. Groq API Documentation URL: <https://trpc.io/docs> (дата звернення: 02.06.2026)
14. Inertia.js Documentation. URL: <https://inertiajs.com/docs/v3/getting-started> (дата звернення: 02.06.2026).
15. Laravel Documentation URL: <https://laravel.com/docs/12.x/installation> (дата звернення: 02.06.2026).
16. Martin, R. C.. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Upper Saddle River: Prentice Hall, 2017, 432 p.
17. Myers G. J. The Art of Software Testing / G. J. Myers, T. Badgett, C. Sandler. 3rd ed. Hoboken : John Wiley & Sons, 2012. 256 p.
18. Pollinations API Documentation. Pollinations.ai. URL: <https://gen.pollinations.ai/docs> (дата звернення: 02.06.2026).
19. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 02.06.2026).
20. React.js Documentation URL: <https://react.dev/> (дата звернення: 02.06.2026)
21. Remove.bg API documentation. URL: <https://www.remove.bg/uk/api> (дата звернення: 02.06.2026).
22. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O'Reilly Media, 2020. 432 p.
23. Segmind Docs. URL: <https://docs.segmind.com/> (дата звернення: 02.06.2026).
24. Shadcn/ui Documentation URL: <https://ui.shadcn.com/> (дата звернення: 02.06.2026).
25. Tailwind CSS Documentation URL: <https://tailwindcss.com/docs> (дата звернення: 02.06.2026).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

26. TinyMCE Documentation. URL: <https://www.tiny.cloud/docs/tinymce/latest/> (дата звернення: 02.06.2026).
27. TypeScript Documentation. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 02.06.2026).
28. Understanding the MVC Pattern in Laravel: A Deep Dive. Medium. URL: <https://medium.com/@imtaslim/understanding-the-mvc-pattern-in-laravel-a-deep-dive-be80d8338fef> (дата звернення: 02.06.2026).
29. Web Performance Optimization URL: <https://web.dev/performance/> (дата звернення: 02.06.2026).
30. What is Use Case Diagram? Visual Paradigm. URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> (дата звернення: 02.06.2026).

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

ЛІСТИНГ РОУТИНГ

<?php

```

use App\Http\Controllers\Admin\AdminController;
use App\Http\Controllers\Admin\BlogController;
use App\Http\Controllers\Admin\BriefController as AdminBriefController;
use App\Http\Controllers\Admin\GroqController;
use App\Http\Controllers\Admin\PollinationsController;
use App\Http\Controllers\Admin\PhotoshootController;
use App\Http\Controllers\Admin\PortfolioController;
use App\Http\Controllers\Admin\SettingsController;
use App\Http\Controllers\BriefController;
use App\Http\Controllers\PublicController;
use Illuminate\Support\Facades\Route;

Route::get('/', [PublicController::class, 'home'])->name('home');
Route::get('/photoshoots', [PublicController::class, 'photoshoots'])->name('photoshoots');
Route::get('/editing', [PublicController::class, 'editing'])->name('editing');
Route::get('/portfolio', [PublicController::class, 'portfolio'])->name('portfolio');
Route::get('/blog', [PublicController::class, 'blog'])->name('blog');
Route::get('/blog/{slug}', [PublicController::class, 'showPost'])->name('blog.show');
Route::get('/contacts', [PublicController::class, 'contacts'])->name('contacts');
Route::post('/briefs', [BriefController::class, 'store'])->name('briefs.store');
Route::post('/editing', [\App\Http\Controllers\EditingOrderController::class, 'store'])->name('editing.store');

Route::middleware(['auth', 'verified', 'admin'])->prefix('admin')->name('admin.')->group(function () {
    Route::get('/', [AdminController::class, 'index'])->name('dashboard');

    Route::resource('photoshoots', PhotoshootController::class);
    Route::resource('portfolio', PortfolioController::class);
    Route::resource('blog', BlogController::class);
    Route::resource('briefs', AdminBriefController::class)->only(['index', 'show', 'destroy']);
    Route::post('briefs/{brief}/read', [AdminBriefController::class, 'markAsRead'])->name('briefs.read');
    Route::resource('editing-orders', \App\Http\Controllers\Admin\EditingOrderController::class)->only(['index', 'show', 'update', 'destroy']);

    Route::get('groq/settings', [GroqController::class, 'getSettings'])->name('groq.settings');
    Route::post('groq/settings', [GroqController::class, 'saveSettings'])->name('groq.settings.save');
    Route::post('groq', GroqController::class)->name('groq');
    Route::post('pollinations', PollinationsController::class)->name('pollinations');
    Route::delete('pollinations', [PollinationsController::class, 'destroy'])->name('pollinations.destroy');

    Route::get('settings', [SettingsController::class, 'index'])->name('settings.index');
    Route::post('settings', [SettingsController::class, 'update'])->name('settings.update');

    Route::get('google/redirect', [\App\Http\Controllers\Admin\GoogleAuthController::class, 'redirect'])->
>name('google.redirect');
    Route::get('google/callback', [\App\Http\Controllers\Admin\GoogleAuthController::class, 'callback'])->
>name('google.callback');

    Route::get('ai-editor', [\App\Http\Controllers\Admin\AiEditorController::class, 'index'])->name('ai-
editor.index');

```

					ІПЗ.КР.Б – 121 – 26 - ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Route::post('ai-editor/remove-bg', [\App\Http\Controllers\Admin\AiEditorController::class, 'removeBg'])->
>name('ai-editor.remove-bg');
Route::post('ai-editor/segmind', [\App\Http\Controllers\Admin\SegmindController::class, 'process'])->
>name('ai-editor.segmind');
});

Route::middleware(['auth', 'verified'])->group(function () {
    Route::get('dashboard', function () {
        return redirect()->route('admin.dashboard');
    }->name('dashboard'));
});

require __DIR__.'./settings.php';

```

<?php

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\Controller;
use Inertia\Inertia;
use Inertia\Response;
```

```
class AdminController extends Controller
{
    public function index(): Response
    {
        return Inertia::render('admin/dashboard', [
            'stats' => [
                'briefs' => \App\Models\Brief::whereNull('read_at')->count(),
                'photoshoots' => \App\Models\PhotoshootType::count(),
                'portfolio' => \App\Models\PortfolioItem::count(),
                'posts' => \App\Models\Post::count(),
            ],
        ]);
    }
}
```

<?php

```
namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Inertia\Inertia;
use Inertia\Response;
```

```
class AiEditorController extends Controller
{
    public function index(): Response
    {
        return Inertia::render('admin/ai-editor/index');
    }

    public function removeBg(Request $request)
    {

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 78 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

\Illuminate\Support\Facades\Log::info('AI Editor: Request received');

$request->validate([
    'image' => 'required|image|max:10240', // 10MB max
    'size' => 'nullable|string|in:auto,preview',
    'type' => 'nullable|string|in:auto,person,product,car',
    'bg_color' => 'nullable|string',
]);

$apiKey = config('services.remove_bg.key') ?? env('REMOVE_BG_API_KEY');

if (!$apiKey) {
    \Illuminate\Support\Facades\Log::error('AI Editor: API key missing');
    return response()->json(['error' => 'API key not configured'], 500);
}

$imageFile = $request->file('image');

$params = [
    'size' => $request->input('size', 'auto'),
    'type' => $request->input('type', 'auto'),
];

if ($request->filled('bg_color')) {
    $params['bg_color'] = str_replace('#', '', $request->input('bg_color'));
}

\Illuminate\Support\Facades\Log::info('AI Editor: Sending request to remove.bg', [
    'filename' => $imageFile->getClientOriginalName(),
    'params' => $params,
]);

try {
    $response = Http::withHeaders([
        'X-API-Key' => $apiKey,
    ])->attach(
        'image_file',
        file_get_contents($imageFile->getPathname()),
        $imageFile->getClientOriginalName()
    )->post('https://api.remove.bg/v1.0/removebg', $params);

    if ($response->successful()) {
        \Illuminate\Support\Facades\Log::info('AI Editor: Success');
        $base64Image = base64_encode($response->body());
        return response()->json([
            'image' => 'data:image/png;base64,' . $base64Image,
        ]);
    }

    $error = $response->json();
    \Illuminate\Support\Facades\Log::error('AI Editor: remove.bg error', ['error' => $error]);
    return response()->json([
        'error' => $error['errors'][0]['title'] ?? 'Помилка при видаленні фону',
    ], 400);
} catch (\Exception $e) {
    \Illuminate\Support\Facades\Log::error('AI Editor: Exception', ['message' => $e->getMessage()]);
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 79 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        return response()->json(['error' => 'Сталася помилка на сервері: ' . $e->getMessage()], 500);
    }
}
}
<?php

```

```

namespace App\Http\Controllers\Admin;

```

```

use App\Http\Controllers\Controller;
use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
use Inertia\Inertia;
use Inertia\Response;

```

```

class BlogController extends Controller

```

```

{
    public function index(): Response
    {
        return Inertia::render('admin/blog/index', [
            'posts' => Post::orderBy('created_at', 'desc')->get(),
        ]);
    }

```

```

    public function create(): Response
    {
        return Inertia::render('admin/blog/form', [
            'isEditing' => false,
            'post' => new Post,
        ]);
    }

```

```

    protected function savePostToFile(Post $post)
    {
        $directory = resource_path('views/public/posts');
        if (! File::exists($directory)) {
            File::makeDirectory($directory, 0755, true);
        }

```

```

        $content = "@extends('layouts.public')\n\n";
        $content .= "@section('title', '".addslashes($post->title). "')\n\n";
        $content .= "@section('content')\n";
        $content .= "<section class='py-24 px-6'\>\n";
        $content .= "    <div class='mx-auto max-w-3xl'\>\n";
        $content .= "        <div class='mb-12'\>\n";
        $content .= "            <a href='\"{{ route('blog') }}\"' class='text-[10px] uppercase tracking-[0.3em] text-
[#6B6B6B] hover:text-[#2D2D2D] transition\">← Назад до блогу</a>\n";
        $content .= "        </div>\n";
        $content .= "        <h1 class='text-4xl md:text-5xl font-serif mb-8 uppercase tracking-tight\">{$post-
>title}</h1>\n";
        $content .= "        <div class='flex items-center gap-6 mb-12 text-[10px] uppercase tracking-[0.3em] text-
[#6B6B6B]'\>\n";
        $content .= "            <span>{$post->created_at->translatedFormat('d F, Y')}</span>\n";

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата | | 80 |

```

        $content .= "        @if(\$post->category) <div class=\"h-[1px] w-8 bg-[#E8E2D9]\"></div> <span>{\$post-
>category}</span> @endif\n";
        $content .= "    </div>\n";
        $content .= "    <div class=\"blog-content text-[#6B6B6B] font-light leading-relaxed\">\n";
        $content .= "        {!! \$post->content !!}\n";
        $content .= "    </div>\n";
        $content .= " </div>\n";
        $content .= "</section>\n";
        $content .= "<style>\n";
        $content .= "    .blog-content p { margin-bottom: 1.5rem; }\n";
        $content .= "    .blog-content ul { list-style-type: disc; padding-left: 1.5rem; margin-bottom: 1.5rem; }\n";
        $content .= "    .blog-content ol { list-style-type: decimal; padding-left: 1.5rem; margin-bottom: 1.5rem; }\n";
        $content .= "    .blog-content strong { font-weight: 600; color: #2D2D2D; }\n";
        $content .= "    .blog-content blockquote { border-left: 2px solid #E8E2D9; padding-left: 1.5rem; font-style:
italic; margin-bottom: 1.5rem; }\n";
        $content .= "    .blog-content h1, .blog-content h2, .blog-content h3 { font-family: serif; color: #2D2D2D;
margin-top: 2rem; margin-bottom: 1rem; text-transform: uppercase; letter-spacing: -0.02em; }\n";
        $content .= "</style>\n";
        $content .= "@endsection\n";

```

```

        File::put($directory.'/'.$post->slug.'.blade.php', $content);
    }

```

```

public function store(Request $request)
{

```

```

    $validated = $request->validate([
        'title' => 'required|string|max:255',
        'content' => 'required|string',
        'category' => 'nullable|string|max:255',
        'image' => 'nullable|image|max:5120',
    ]);

```

```

    $validated['slug'] = Str::slug($validated['title']);
    $validated['published_at'] = now();

```

```

    if ($request->hasFile('image')) {
        $validated['image_path'] = $request->file('image')->store('blog', 'public');
    }

```

```

    $post = Post::create($validated);
    $this->savePostToFile($post);

```

```

    return redirect()->route('admin.blog.index')->with('success', 'Статтю успішно опубліковано.');
```

```

public function edit(Post $blog): Response
{

```

```

    return Inertia::render('admin/blog/form', [
        'isEditing' => true,
        'post' => $blog,
    ]);
}

```

```

public function update(Request $request, Post $blog)
{

```

```

    $validated = $request->validate([

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 81 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        'title' => 'required|string|max:255',
        'content' => 'required|string',
        'category' => 'nullable|string|max:255',
        'image' => 'nullable|image|max:5120',
    ]);

    $oldSlug = $blog->slug;
    $validated['slug'] = Str::slug($validated['title']);

    if ($request->hasFile('image')) {
        if ($blog->image_path) {
            Storage::disk('public')->delete($blog->image_path);
        }
        $validated['image_path'] = $request->file('image')->store('blog', 'public');
    }

    $blog->update($validated);

    // Видаляємо старий файл, якщо slug змінився
    if ($oldSlug !== $blog->slug) {
        File::delete(resource_path("views/public/posts/{$oldSlug}.blade.php"));
    }

    $this->savePostToFile($blog);

    return redirect()->route('admin.blog.index')->with('success', 'Статтю оновлено.');
```

```

    public function destroy(Post $blog)
    {
        if ($blog->image_path) {
            Storage::disk('public')->delete($blog->image_path);
        }

        File::delete(resource_path("views/public/posts/{$blog->slug}.blade.php"));
        $blog->delete();

        return redirect()->route('admin.blog.index')->with('success', 'Статтю видалено.');
```

```

    }
}
<?php
```

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use App\Models\Brief;
use Illuminate\Http\RedirectResponse;
use Inertia\Inertia;
use Inertia\Response;
```

```

class BriefController extends Controller
{
    public function index(): Response
    {
        $briefs = Brief::with('photoshootType')
            ->orderBy('created_at', 'desc')
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 82 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        ->get();

        return Inertia::render('admin/briefs/index', [
            'briefs' => $briefs,
        ]);
    }

    public function show(Brief $brief): Response
    {
        if (!$brief->read_at) {
            $brief->update(['read_at' => now()]);
        }

        $brief->load('photoshootType');

        return Inertia::render('admin/briefs/show', [
            'brief' => $brief,
        ]);
    }

    public function destroy(Brief $brief): RedirectResponse
    {
        $brief->delete();

        return redirect()->route('admin.briefs.index')->with('success', 'Бриф видалено.');
```

```

    }

    public function markAsRead(Brief $brief): RedirectResponse
    {
        $brief->update(['read_at' => now()]);

        return back()->with('success', 'Бриф позначено як прочитаний.');
```

```

    }
}
<?php

```

```

namespace App\Http\Controllers\Admin;

```

```

use App\Http\Controllers\Controller;
use App\Models\EditingOrder;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Inertia\Response;

```

```

class EditingOrderController extends Controller
{

```

```

    public function index(): Response
    {
        $orders = EditingOrder::orderBy('created_at', 'desc')->get();

        return Inertia::render('admin/editing-orders/index', [
            'orders' => $orders,
            'statuses' => EditingOrder::getStatuses(),

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 83 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    });
}

public function show(EditingOrder $editingOrder): Response
{
    return Inertia::render('admin/editing-orders/show', [
        'order' => $editingOrder,
        'statuses' => EditingOrder::getStatuses(),
    ]);
}

public function update(Request $request, EditingOrder $editingOrder): RedirectResponse
{
    $validated = $request->validate([
        'status' => 'required|string|in:' . implode(',', array_keys(EditingOrder::getStatuses())),
    ]);

    $editingOrder->update($validated);

    return back()->with('success', 'Статус замовлення оновлено.');
```

```

}

public function destroy(EditingOrder $editingOrder): RedirectResponse
{
    $editingOrder->delete();

    return redirect()->route('admin.editing-orders.index')->with('success', 'Замовлення видалено.');
```

```

}
}
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Setting;
use Google\Client;
use Google\Service\Drive;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class GoogleAuthController extends Controller
{
    public function redirect(): RedirectResponse
    {
        $client = new Client();
        $client->setClientId(config('services.google.client_id'));
        $client->setClientSecret(config('services.google.client_secret'));
        $client->setRedirectUri(config('services.google.redirect_uri'));
        $client->addScope(Drive::DRIVE);
        $client->setAccessType('offline');
        $client->setPrompt('consent');

        return redirect()->away($client->createAuthUrl());
    }
}

```

```

public function callback(Request $request): RedirectResponse
{
    $client = new Client();
    $client->setClientId(config('services.google.client_id'));
    $client->setClientSecret(config('services.google.client_secret'));
    $client->setRedirectUri(config('services.google.redirect_uri'));

    $token = $client->fetchAccessTokenWithAuthCode($request->get('code'));

    if (isset($token['error'])) {
        return redirect()->route('admin.settings.index')->with('error', 'Помилка авторизації Google: ' .
$token['error_description']);
    }

    Setting::updateOrCreate(
        ['key' => 'google_drive_oauth_token'],
        ['value' => json_encode($token), 'type' => 'textarea', 'group' => 'system']
    );

    return redirect()->route('admin.settings.index')->with('success', 'Google Drive успішно підключено!');
}
}
<?php

```

```

namespace App\Http\Controllers\Admin;

```

```

use App\Http\Controllers\Controller;
use App\Models\Setting;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;

```

```

class GroqController extends Controller
{

```

```

    public function __invoke(Request $request): \Illuminate\Http\JsonResponse
    {
        $request->validate([
            'input' => 'required|string',
        ]);

```

```

        $apiKey = config('api.groq');

```

```

        if (!$apiKey) {
            return response()->json(['error' => 'Groq API key is not configured.'], 500);
        }

```

```

        $model = Setting::where('key', 'groq_model')->value('value') ?: 'llama-3.3-70b-versatile';

```

```

        $instructions = Setting::where('key', 'groq_instructions')->value('value') ?: 'You are a helpful assistant.
Provide the response as text that will replace the selected text in a blog post editor. Only return the
improved/transformed text, no explanations.';

```

```

        $response = Http::withToken($apiKey)
            ->post('https://api.groq.com/openai/v1/responses', [
                'model' => $model,
                'input' => $request->input('input'),
                'instructions' => $instructions,
            ]);

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 85 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

if ($response->failed()) {
    return response()->json([
        'error' => 'Failed to communicate with Groq API.',
        'details' => $response->json(),
    ], $response->status());
}

$data = $response->json();

\Illuminate\Support\Facades\Log::info('Groq API Response:', $data);

// The 'output' array can contain 'reasoning' followed by 'message'
$message = collect($data['output'] ?? [])
    ->firstWhere('type', 'message');

$text = $message['content'][0]['text'] ?? null;

if (!$text) {
    return response()->json([
        'error' => 'Could not extract text from Groq response.',
        'raw_response' => $data,
    ], 500);
}

return response()->json(['text' => $text]);
}

public function getSettings(): \Illuminate\Http\JsonResponse
{
    return response()->json([
        'model' => Setting::where('key', 'groq_model')->value('value') ?: 'llama-3.3-70b-versatile',
        'instructions' => Setting::where('key', 'groq_instructions')->value('value') ?: 'You are a helpful assistant.
Provide the response as text that will replace the selected text in a blog post editor. Only return the
improved/transformed text, no explanations.',
    ]);
}

public function saveSettings(Request $request): \Illuminate\Http\JsonResponse
{
    $request->validate([
        'model' => 'required|string',
        'instructions' => 'required|string',
    ]);

    Setting::updateOrCreate(
        ['key' => 'groq_model'],
        ['value' => $request->model, 'type' => 'string', 'group' => 'groq']
    );

    Setting::updateOrCreate(
        ['key' => 'groq_instructions'],
        ['value' => $request->instructions, 'type' => 'text', 'group' => 'groq']
    );

    return response()->json(['message' => 'Settings saved successfully']);
}

```

```

    }
}
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\PhotoshootType;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
use Inertia\Inertia;
use Inertia\Response;

class PhotoshootController extends Controller
{
    public function index(): Response
    {
        return Inertia::render('admin/photoshoots/index', [
            'photoshoots' => PhotoshootType::orderBy('order')->get(),
        ]);
    }

    public function create(): Response
    {
        return Inertia::render('admin/photoshoots/form', [
            'isEditing' => false,
            'photoshoot' => new PhotoshootType,
        ]);
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'nullable|string',
            'order' => 'nullable|integer',
            'image' => 'nullable|image|max:5120',
        ]);

        $validated['slug'] = Str::slug($validated['title']);

        if ($request->hasFile('image')) {
            $validated['image_path'] = $request->file('image')->store('photoshoots', 'public');
        }

        PhotoshootType::create($validated);

        return redirect()->route('admin.photoshoots.index')->with('success', 'Вид фотосесії успішно додано.');
```

```

    }

    public function edit(PhotoshootType $photoshoot): Response
    {
        return Inertia::render('admin/photoshoots/form', [
            'isEditing' => true,
            'photoshoot' => $photoshoot,

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 87 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    });
}

public function update(Request $request, PhotoshootType $photoshoot)
{
    $validated = $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'nullable|string',
        'order' => 'nullable|integer',
        'image' => 'nullable|image|max:5120',
    ]);

    $validated['slug'] = Str::slug($validated['title']);

    if ($request->hasFile('image')) {
        if ($photoshoot->image_path) {
            Storage::disk('public')->delete($photoshoot->image_path);
        }
        $validated['image_path'] = $request->file('image')->store('photoshoots', 'public');
    }

    $photoshoot->update($validated);

    return redirect()->route('admin.photoshoots.index')->with('success', 'Оновлено успішно.');
```

```

}

public function destroy(PhotoshootType $photoshoot)
{
    if ($photoshoot->image_path) {
        Storage::disk('public')->delete($photoshoot->image_path);
    }
    $photoshoot->delete();

    return redirect()->route('admin.photoshoots.index')->with('success', 'Вид фотосесії видалено.');
```

```

}
}
<?php

```

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;

```

```

class PollinationsController extends Controller
{
    public function __invoke(Request $request): \Illuminate\Http\JsonResponse
    {
        $request->validate([
            'prompt' => 'required|string',
            'width' => 'nullable|numeric|min:64|max:2048',
            'height' => 'nullable|numeric|min:64|max:2048',
        ]);
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 88 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    $prompt = $request->input('prompt');
    $width = $request->input('width', 1024);
    $height = $request->input('height', 1024);
    $seed = rand(0, 1000000);

    $encodedPrompt = urlencode($prompt);
    $imageUrl =
    "https://image.pollinations.ai/prompt/{ $encodedPrompt }?width={ $width }&height={ $height }&nologo=true&seed={ $seed }";

    try {
        $response = Http::get($imageUrl);

        if ($response->successful()) {
            $filename = 'pollinations_' . Str::random(10) . '_' . time() . '.jpg';
            $path = 'blog/' . $filename;

            Storage::disk('public')->put($path, $response->body());

            $localUrl = Storage::url($path);

            return response()->json([
                'url' => $localUrl,
                'path' => $path,
                'prompt' => $prompt
            ]);
        }
    } catch (\Exception $e) {
        return response()->json([
            'error' => 'Failed to download image from Pollinations.',
            'message' => $e->getMessage()
        ], 500);
    }

    return response()->json([
        'error' => 'Failed to generate image.'
    ], 500);
}

public function destroy(Request $request): \Illuminate\Http\JsonResponse
{
    $request->validate([
        'path' => 'required|string',
    ]);

    $path = $request->input('path');

    if (Storage::disk('public')->exists($path)) {
        Storage::disk('public')->delete($path);
        return response()->json(['message' => 'Image deleted successfully']);
    }

    return response()->json(['error' => 'Image not found'], 404);
}
}
<?php

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 89 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\PhotoshootType;
use App\Models\PortfolioItem;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;
use Inertia\Inertia;
use Inertia\Response;

class PortfolioController extends Controller
{
    public function index(): Response
    {
        return Inertia::render('admin/portfolio/index', [
            'portfolio' => PortfolioItem::with('photoshootType')->orderBy('order', 'desc')->get(),
        ]);
    }

    public function create(): Response
    {
        return Inertia::render('admin/portfolio/form', [
            'photoshootTypes' => PhotoshootType::all(),
        ]);
    }

    public function store(Request $request)
    {
        $request->validate([
            'images.*' => 'required|image|max:10240', // 10MB per image
            'photoshoot_type_id' => 'nullable|exists:photoshoot_types,id',
        ]);

        if ($request->hasFile('images')) {
            $maxOrder = PortfolioItem::max('order') ?? 0;
            foreach ($request->file('images') as $index => $image) {
                $path = $image->store('portfolio', 'public');
                PortfolioItem::create([
                    'image_path' => $path,
                    'photoshoot_type_id' => $request->photoshoot_type_id,
                    'order' => $maxOrder + $index + 1,
                ]);
            }
        }

        return redirect()->route('admin.portfolio.index')->with('success', 'Зображення успішно додано до портфоліо.');
```

```

    }

    public function destroy(PortfolioItem $portfolio)
    {
        Storage::disk('public')->delete($portfolio->image_path);
        $portfolio->delete();

        return redirect()->route('admin.portfolio.index')->with('success', 'Зображення видалено.');
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 90 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    }
}
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;

class SegmindController extends Controller
{
    public function process(Request $request)
    {
        $request->validate([
            'model' => 'required|string',
            'image' => 'required|string',
            'mask' => 'nullable|string',
            'params' => 'nullable|array',
        ]);

        $apiKey = config('services.segmind.key') ?? env('SEGMIND_API_KEY');

        if (!$apiKey) {
            return response()->json(['error' => 'Segmind API key not configured'], 500);
        }

        $model = $request->input('model');
        $data = $request->input('params', []);

        $cleanImage = $this->cleanBase64($request->input('image'));
        $data['base64_image'] = $cleanImage; // Most models use this key

        // Model specific adjustments
        switch ($model) {
            case 'sdxl1.0-inpainting':
            case 'lama-inpainting':
                if (!$request->filled('mask')) {
                    return response()->json(['error' => 'Mask is required for this model'], 400);
                }
                $data['base64_mask'] = $this->cleanBase64($request->input('mask'));
                // Segmind inpainting usually needs 'base64_image' and 'base64_mask'
                break;

            case 'esrgan-4x':
                $data['image'] = $cleanImage;
                unset($data['base64_image']);
                break;

            case 'bg-blur':
                $data['image'] = $cleanImage;
                unset($data['base64_image']);
                break;

            case 'ic-light':

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата | | 91 |

```

        $data['image'] = $cleanImage;
        unset($data['base64_image']);
        break;
    }

    Log::info("Segmind: Calling model $model", ['params' => array_keys($data)]);

    try {
        $response = Http::withHeaders([
            'x-api-key' => $apiKey,
            'Content-Type' => 'application/json',
        ])->timeout(120)->post("https://api.segmind.com/v1/$model", $data);

        if ($response->successful()) {
            // Segmind returns the image binary
            $base64Image = base64_encode($response->body());
            return response()->json([
                'image' => 'data:image/png;base64,' . $base64Image,
            ]);
        }

        $error = $response->json();
        Log::error("Segmind Error: $model", ['response' => $error]);

        return response()->json([
            'error' => $error['message'] ?? $error['error'] ?? 'Помилка AI сервісу Segmind',
        ], $response->status());

    } catch (\Exception $e) {
        Log::error("Segmind Exception: " . $e->getMessage());
        return response()->json(['error' => 'Помилка сервера: ' . $e->getMessage()], 500);
    }
}

private function cleanBase64(string $base64): string
{
    if (str_contains($base64, ',')) {
        return explode(',', $base64)[1];
    }
    return $base64;
}
}
<?php

```

```
namespace App\Http\Controllers\Admin;
```

```

use App\Http\Controllers\Controller;
use App\Models\Setting;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Inertia\Response;

```

```

class SettingsController extends Controller
{
    public function index(): Response
    {

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 92 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

return Inertia::render('admin/settings/index', [
    'settings' => Setting::where('group', '!=', 'system')
        ->get()
        ->groupBy('group'),
]);
}

public function update(Request $request)
{
    $settings = $request->input('settings');

    foreach ($settings as $key => $value) {
        Setting::updateOrCreate(
            ['key' => $key],
            ['value' => $value]
        );
    }

    return redirect()->back()->with('success', 'Налаштування успішно збережено.');
```

Фронтенд сторінки:

```

import AppLayout from '@layouts/app-layout';
import { Head, useForm, Link } from '@inertiajs/react';
import { BreadcrumbItem } from '@types';
import { Save, ArrowLeft } from 'lucide-react';
import { Button } from '@components/ui/button';
import { Input } from '@components/ui/input';
import { Label } from '@components/ui/label';
import { Card, CardContent } from '@components/ui/card';
import { Textarea } from '@components/ui/textarea';
import {
    Sheet,
    SheetContent,
    SheetDescription,
    SheetHeader,
    SheetTitle,
    SheetTrigger,
} from '@components/ui/sheet';
import {
    Select,
    SelectContent,
    SelectItem,
    SelectTrigger,
    SelectValue,
} from '@components/ui/select';
import { Editor } from '@tinymce/tinymce-react';
import { useState, useEffect } from 'react';
import {
    Sparkles,
    History,
    Trash2,
    Settings2,
    Save as SaveIcon,
} from 'lucide-react';
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| Змн. | Арк. | № докум. | Підпис | Дата | | 93 |

```

import { toast } from 'sonner';

// Wayfinder
import {
  index,
  store,
  update,
} from '@actions/App/Http/Controllers/Admin/BlogController';

interface GroqResponse {
  id: string;
  text: string;
  timestamp: string;
}

const MODELS = [
  { id: 'llama-3.3-70b-versatile', name: 'Llama 3.3 70B (Best Balance)' },
  { id: 'openai/gpt-oss-120b', name: 'GPT-OSS 120B (High Quality)' },
  { id: 'llama-3.1-8b-instant', name: 'Llama 3.1 8B (Fastest)' },
  { id: 'qwen/qwen3-32b', name: 'Qwen 3 32B (Advanced)' },
];

export default function BlogForm({
  isEditing,
  post: blogPost,
}): {
  isEditing: boolean;
  post: any;
}) {
  const { data, setData, post, put, processing, errors } = useForm({
    title: blogPost?.title || "",
    content: blogPost?.content || "",
    category: blogPost?.category || "",
    image: null as File | null,
  });

  const [groqHistory, setGroqHistory] = useState<GroqResponse[]>([]);
  const [groqSettings, setGroqSettings] = useState({
    model: 'llama-3.3-70b-versatile',
    instructions:
      'You are a helpful assistant. Provide the response as text that will replace the selected text in a blog post editor. Only return the improved/transformed text, no explanations.',
  });
  const [isSavingSettings, setIsSavingSettings] = useState(false);

  // Update form data when blogPost changes (important for editing)
  useEffect(() => {
    if (isEditing && blogPost) {
      setData({
        title: blogPost.title || "",
        content: blogPost.content || "",
        category: blogPost.category || "",
        image: null,
      });
    }
  }, [blogPost]);

```

```

useEffect(() => {
  fetch('/admin/groq/settings')
    .then((res) => res.json())
    .then((data) => setGroqSettings(data))
    .catch((err) => console.error('Failed to load Groq settings', err));
}, []);

const saveSettings = () => {
  setIsSavingSettings(true);
  fetch('/admin/groq/settings', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'X-CSRF-TOKEN':
        (
          document.querySelector(
            'meta[name="csrf-token"]',
          ) as HTMLMetaElement
        )?.content || '',
    },
    body: JSON.stringify(groqSettings),
  })
    .then((res) => res.json())
    .then(() => toast.success('Settings saved'))
    .catch((err) => toast.error('Failed to save settings'))
    .finally(() => setIsSavingSettings(false));
};

const breadcrumbs: BreadcrumbItem[] = [
  { title: 'Панель керування', href: '/admin' },
  { title: 'Блог', href: index.url() },
  { title: isEditing ? 'Редагувати' : 'Створити', href: '#' },
];

const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();

  if (isEditing) {
    put(update.url(blogPost.id));
  } else {
    post(store.url());
  }
};

return (
  <AppLayout breadcrumbs={breadcrumbs}>
    <Head title={isEditing ? 'Редагувати статтю' : 'Нова стаття'} />

    <div className="mx-auto w-full max-w-5xl p-6">
      <div className="mb-8 flex items-center justify-between">
        <h1 className="text-2xl font-semibold">
          {isEditing ? 'Редагувати' : 'Написати'} статтю
        </h1>
        <div className="flex gap-2">
          <Sheet>

```

```

<SheetTrigger asChild>
  <Button
    variant="outline"
    size="sm"
    className="flex items-center gap-2"
  >
    <Sparkles
      size={16}
      className="text-amber-500"
    />
    Groq AI
  </Button>
</SheetTrigger>
<SheetContent className="flex w-[400px] flex-col gap-6 sm:w-[540px]">
  <SheetHeader>
    <SheetTitle className="flex items-center gap-2">
      <Sparkles
        size={20}
        className="text-amber-500"
      />
      Groq AI Assistant
    </SheetTitle>
    <SheetDescription>
      Configure AI model and review your
      history.
    </SheetDescription>
  </SheetHeader>

  <div className="bg-muted/20 space-y-4 rounded-lg border p-4">
    <div className="mb-2 flex items-center gap-2 text-sm font-medium">
      <Settings2 size={14} />
      Settings
    </div>
    <div className="space-y-3">
      <div className="space-y-1.5">
        <Label className="text-xs">
          AI Model
        </Label>
        <Select
          value={groqSettings.model}
          onValueChange={(v) =>
            setGroqSettings((s) => ({
              ...s,
              model: v,
            }))
        >
          <SelectTrigger className="border-muted-foreground/20 w-full bg-background">
            <SelectValue placeholder="Select model" />
          </SelectTrigger>
          <SelectContent
            className="z-[100] min-w-[var(--radix-select-trigger-width)] border shadow-
lg"
            style={{
              backgroundColor:
                'white',

```

```

        color: 'black',
      }}
    >
    {MODELS.map((m) => (
      <SelectItem
        key={m.id}
        value={m.id}
        className="cursor-pointer text-black focus:bg-slate-100 focus:text-black"
        style={{
          color: 'black',
        }}
      >
        {m.name}
      </SelectItem>
    ))}
  </SelectContent>
</Select>
</div>
<div className="space-y-1.5">
  <Label className="text-xs">
    System Prompt (Instructions)
  </Label>
  <Textarea
    value={
      groqSettings.instructions
    }
    onChange={(e) =>
      setGroqSettings((s) => ({
        ...s,
        instructions:
          e.target.value,
      }))
    }
    rows={4}
    className="resize-none text-xs"
    placeholder="Instructions for the AI..."
  />
</div>
<Button
  size="sm"
  className="w-full"
  onClick={saveSettings}
  disabled={isSavingSettings}
>
  <SaveIcon
    size={14}
    className="mr-2"
  />
  Save Settings
</Button>
</div>
</div>

<div className="flex flex-1 flex-col overflow-hidden">
  <div className="mb-4 flex items-center justify-between">
    <div className="flex items-center gap-2 text-sm font-medium">

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 97 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        <History size={16} />
        History
      </div>
      {groqHistory.length > 0 && (
        <Button
          variant="ghost"
          size="sm"
          className="text-muted-foreground h-8 text-xs"
          onClick={() =>
            setGroqHistory([])
          }
        >
        <Trash2
          size={14}
          className="mr-1"
        />
        Clear
      </Button>
    )}
  </div>

  <div className="flex-1 space-y-4 overflow-y-auto pr-2">
    {groqHistory.length === 0 ? (
      <div className="text-muted-foreground flex h-40 flex-col items-center justify-center
text-center text-sm">
        <History
          size={32}
          className="mb-2 opacity-10"
        />
        <p>
          No responses yet.
        <br />
          Select text and use "Groq"
          from context menu.
        </p>
      </div>
    ) : (
      groqHistory.map((item) => (
        <div
          key={item.id}
          className="bg-muted/30 group relative rounded-lg border p-3 text-xs"
        >
          <div className="text-muted-foreground mb-1 flex justify-between text-
[10px]">
            <span>
              {item.timestamp}
            </span>
            <button
              onClick={() => {
                navigator.clipboard.writeText(
                  item.text,
                );
                toast.success(
                  'Copied to clipboard',
                );
              }}
            >

```

```

group-hover:opacity-100"
                                className="hover:text-foreground underline opacity-0 transition-opacity
                                >
                                Copy
                                </button>
                                </div>
                                <div className="whitespace-pre-wrap">
                                {item.text}
                                </div>
                                </div>
                                </div>
                                ))
                                }}
                                </div>
                                </div>
                                </SheetContent>
                                </Sheet>
                                <Button variant="ghost" size="sm" asChild>
                                <Link
                                href={index.url()}
                                className="flex items-center gap-2"
                                >
                                <ArrowLeft size={16} />
                                Назад
                                </Link>
                                </Button>
                                </div>
                                </div>

                                <Card className="border-none shadow-sm">
                                <CardContent className="pt-6">
                                <form onSubmit={handleSubmit} className="space-y-6">
                                <div className="space-y-2">
                                <Label htmlFor="title">Заголовок</Label>
                                <Input
                                id="title"
                                type="text"
                                value={data.title}
                                onChange={(e) =>
                                setData('title', e.target.value)
                                }
                                className="py-6 text-lg font-medium"
                                />
                                {errors.title && (
                                <p className="text-destructive text-sm">
                                {errors.title}
                                </p>
                                )}
                                </div>

                                <div className="grid grid-cols-2 gap-6">
                                <div className="space-y-2">
                                <Label htmlFor="category">Категорія</Label>
                                <Input
                                id="category"
                                type="text"
                                value={data.category}

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 99 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        onChange={e =>
          setData('category', e.target.value)
        }
        placeholder="Наприклад: Поради"
      />
    </div>

    <div className="space-y-2">
      <Label htmlFor="image">Обкладинка</Label>
      <Input
        id="image"
        type="file"
        onChange={e =>
          setData(
            'image',
            e.target.files
              ? e.target.files[0]
              : null,
          )
        }
        className="cursor-pointer"
      />
    </div>
  </div>

  <div className="space-y-2">
    <Label htmlFor="content">Зміст статті</Label>
    <div className="min-h-[650px] overflow-hidden rounded-md border">
      <Editor
        apiKey={
          import.meta.env.VITE_TINYMCE_API_KEY
        }
        value={data.content}
        onEditorChange={(content) =>
          setData('content', content)
        }
        init={{
          height: 650,
          menubar: true,
          plugins: [
            'anchor',
            'autolink',
            'charmap',
            'codesample',
            'emoticons',
            'link',
            'lists',
            'media',
            'searchreplace',
            'table',
            'visualblocks',
            'wordcount',
            'checklist',
            'mediaembed',
            'casechange',
            'formatpainter',

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 100 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

'pageembed',
'al lychecker',
'tinymcespellchecker',
'permanentpen',
'powerpaste',
'advtable',
'advcode',
'advtemplate',
'tinymceai',
'uploadcare',
'mentions',
'tinycomments',
'tableofcontents',
'footnotes',
'mergetags',
'autocorrect',
'typography',
'inlinecss',
'importword',
'exportword',
'exportpdf',

```

```
],
```

```
toolbar:
```

```

'undo redo | tinymceai-chat tinymceai-quickactions tinymceai-review | blocks
fontfamily fontsize | bold italic underline strikethrough | link media table mergetags | addcomment showcomments
| spellcheckdialog al lycheck typography uploadcare | align lineheight | checklist numlist bullist indent outdent |
emoticons charmap | removeformat',

```

```
contextmenu:
```

```

'link table groq pollinations',

```

```
relative_urls: false,
```

```
remove_script_host: false,
```

```
convert_urls: true,
```

```
images_upload_handler: (
```

```
blobInfo,
```

```
) => {
```

```
return new Promise(
```

```
(resolve) => {
```

```
resolve(
```

```
blobInfo.base64(),
```

```
);
```

```
},
```

```
);
```

```
},
```

```
notification_filter: (
```

```
notification,
```

```
) => {
```

```
// Suppress "Image operation failed" notifications which are noisy in local dev
```

```
if (
```

```
notification.text &&
```

```
notification.text.indexOf(
```

```
'Image operation failed',
```

```
) !== -1
```

```
) {
```

```
return null;
```

```
}
```

```
return notification;
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 101 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

    },
    setup: (editor) => {
        editor.ui.registry.addMenuItem(
            'pollinations_delete',
            {
                text: 'Delete from server',
                icon: 'remove',
                onAction: () => {
                    const node =
                        editor.selection.getNode();
                    if (
                        node &&
                        node.tagName ===
                            'IMG'
                    ) {
                        const path =
                            node.getAttribute(
                                'data-path',
                            );
                        if (path) {
                            fetch(
                                '/admin/pollinations',
                                {
                                    method: 'DELETE',
                                    headers:
                                        {
                                            'Content-Type':
                                                'application/json',
                                            'X-CSRF-TOKEN':
                                                (
                                                    document.querySelector(
                                                        'meta[name="csrf-token"]',
                                                    ) as HTMLMetaElement
                                                ).content ||
                                                "",
                                        },
                                    body: JSON.stringify(
                                        {
                                            path: path,
                                        },
                                    ),
                                },
                            )
                                .then(
                                    (
                                        response,
                                    ) => {
                                        if (
                                            response.ok
                                        ) {
                                            editor.dom.remove(
                                                node,
                                            );
                                            editor.undoManager.add();
                                        } else {

```

```

        console.error(
            'Server failed to delete image',
        );
    }
},
)
.catch(
    (
        err,
    ) =>
        console.error(
            'Error deleting image:',
            err,
        ),
);
}
}
},
);

```

```

editor.ui.registry.addContextMenu(
    'pollinations_image',
    {
        update: (element) => {
            return element &&
                element.tagName ===
                    'IMG' &&
                    element.getAttribute(
                        'data-path',
                    )
                ? 'pollinations_delete'
                : "";
        },
    },
);

```

```

editor.ui.registry.addMenuItem(
    'pollinations',
    {
        text: 'Pollinations AI',
        icon: 'image',
        onAction: () => {
            const selectedText =
                editor.selection.getContent(
                    {
                        format: 'text',
                    },
                );
        },
    );

    editor.windowManager.open(
        {
            title: 'Generate Image (Pollinations)',
            body: {
                type: 'panel',
                items: [

```

```

{
  type: 'textarea',
  name: 'prompt',
  label: 'Prompt',
  placeholder:
    'Describe the image you want to generate...!',
  initialValue:
    selectedText,
},
{
  type: 'grid',
  columns: 2,
  items: [
    {
      type: 'input',
      name: 'width',
      label: 'Width (px)',
      inputMode:
        'numeric',
      initialValue:
        '1024',
    },
    {
      type: 'input',
      name: 'height',
      label: 'Height (px)',
      inputMode:
        'numeric',
      initialValue:
        '1024',
    },
  ],
},
{
  type: 'htmlpanel',
  name: 'previewPanel',
  html:
    '<div id="pollinations-preview-container"
style="margin-top: 15px; border: 1px dashed #ccc; border-radius: 4px; min-height: 200px; display: flex; align-
items: center; justify-content: center; background: #f9f9f9; overflow: hidden;"><p style="color: #666;">Image
preview will appear here</p></div>',
},
],
},
buttons: [
  {
    type: 'cancel',
    name: 'closeBtn',
    text: 'Close',
  },
  {
    type: 'submit',
    name: 'generateBtn',
    text: 'Generate',
    primary: true,
    disabled:
      !selectedText,
  },
]

```

```

    },
    {
      type: 'custom',
      name: 'discardBtn',
      text: 'Discard & Delete',
      enabled: false,
    },
    {
      type: 'custom',
      name: 'acceptBtn',
      text: 'Accept & Insert',
      primary: true,
      enabled: false,
    },
  ],
  onChange: (
    api,
  ) => {
    const data =
      api.getData();

    // Regex to check if strings are purely numeric
    const isNumeric =
      (
        val: string,
      ) =>
        /\d+$/ .test(
          val,
        );

    const width =
      isNumeric(
        data.width,
      )
        ? parseInt(
            data.width,
          )
        : 0;
    const height =
      isNumeric(
        data.height,
      )
        ? parseInt(
            data.height,
          )
        : 0;

    const isValid =
      data.prompt.trim()
        .length >
        0 &&
      isNumeric(
        data.width,
      ) &&
      width >=
        64 &&

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 105 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

width <=
    2048 &&
isNumeric(
    data.height,
) &&
height >=
    64 &&
height <=
    2048;

const hasResult =
    api.isEnabled(
        'acceptBtn',
    );
api.setEnabled(
    'generateBtn',
    isValid &&
        !hasResult,
);
},
onAction: (
    api,
    details,
) => {
    const state =
        api.getData();
    // We'll use a hidden property or global-ish state to track the last
generated image
    // TinyMCE dialogs don't have a great way to store arbitrary state,
so we'll use the DOM or a local variable scope
    const container =
        document.getElementById(
            'pollinations-preview-container',
        );
    const img =
        container?.querySelector(
            'img',
        );
    const path =
        img?.getAttribute(
            'data-temp-path',
        );
    const url =
        img?.src;

    if (
        details.name ===
            'discardBtn' &&
            path
    ) {
        api.block(
            'Deleting from server...',
        );
        fetch(
            '/admin/pollinations',
            {

```

generate again.</p>;

```
method: 'DELETE',
headers:
{
  'Content-Type':
    'application/json',
  'X-CSRF-TOKEN':
    (
      document.querySelector(
        'meta[name="csrf-token"]',
      ) as HTMLMetaElement
    )
    ?.content ||
    "",
},
body: JSON.stringify(
{
  path: path,
},
),
).finally(
() => {
  if (
    container
  ) {
    container.innerHTML =
      '<p style="color: #666;">Image discarded. You can
generate again.</p>;
  }
  api.setEnabled(
    'discardBtn',
    false,
  );
  api.setEnabled(
    'acceptBtn',
    false,
  );
  api.setEnabled(
    'generateBtn',
    true,
  );
  api.unblock();
},
);
} else if (
  details.name ===
    'acceptBtn' &&
    url &&
    path
) {
  const prompt =
    state.prompt;
  editor.insertContent(
    ``,
  );
}
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 107 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        editor.undoManager.add();
        api.close();
    }
},
onSubmit: (
    api,
) => {
    const data =
        api.getData();
    api.block(
        'Generating image...',
    );

    fetch(
        '/admin/pollinations',
        {
            method: 'POST',
            headers:
                {
                    'Content-Type':
                        'application/json',
                    'X-CSRF-TOKEN':
                        (
                            document.querySelector(
                                'meta[name="csrf-token"]',
                            ) as HTMLMetaElement
                        )
                            ?.content ||
                        "",
                },
            body: JSON.stringify(
                {
                    prompt: data.prompt,
                    width: data.width,
                    height: data.height,
                },
            ),
        },
    ).then(
        (
            response,
        ) => {
            if (
                !response.ok
            )
                throw new Error(
                    'Server error',
                );
            return response.json();
        },
    ).then(
        (
            resData,
        ) => {

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 108 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

const container =
  document.getElementById(
    'pollinations-preview-container',
  );
if (
  container &&
  resData.url
) {
  container.innerHTML = ``;
  api.setEnabled(
    'discardBtn',
    true,
  );
  api.setEnabled(
    'acceptBtn',
    true,
  );
  api.setEnabled(
    'generateBtn',
    false,
  );
}
api.unblock();
},
)
.catch(
  (
    error,
  ) => {
    console.error(
      'Error:',
      error,
    );
    const container =
      document.getElementById(
        'pollinations-preview-container',
      );
    if (
      container
    ) {
      container.innerHTML = `<p style="color:
#dc2626;">Error: ${error.message}</p>`;
    }
    api.unblock();
  },
);
},
},
);
},
);
editor.ui.registry.addMenuItem(
  'groq',
  {

```

```

text: 'Groq',
icon: 'ai',
onAction: () => {
  const selectedText =
    editor.selection.getContent(
      {
        format: 'text',
      },
    );
  if (!selectedText)
    return;

  editor.setProgressState(
    true,
  );

  fetch(
    '/admin/groq',
    {
      method: 'POST',
      headers: {
        'Content-Type':
          'application/json',
        'X-CSRF-TOKEN':
          (
            document.querySelector(
              'meta[name="csrf-token"]',
            ) as HTMLMetaElement
          ).content ||
          "",
      },
      body: JSON.stringify(
        {
          input: selectedText,
        },
      ),
    },
  )
    .then(
      (
        response,
      ) =>
        response.json(),
    )
    .then(
      (data) => {
        editor.setProgressState(
          false,
        );
        if (
          data.text
        ) {
          editor.selection.setContent(
            data.text,
          );
        }
      }
    )

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 110 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        setGroqHistory(
            (
                prev,
            ) => [
                {
                    id: Math.random()
                        .toString(
                            36,
                        )
                        .substr(
                            2,
                            9,
                        ),
                    text: data.text,
                    timestamp:
                        new Date().toLocaleTimeString(),
                },
                ...prev,
            ],
        );
    } else if (
        data.error
    ) {
        console.error(
            'Groq Error:',
            data,
        );
        toast.error(
            data.error,
        );
    }
},
)
)
.catch(
    (error) => {
        editor.setProgressState(
            false,
        );
        console.error(
            'Groq Fetch Error:',
            error,
        );
        toast.error(
            'Network error occurred',
        );
    },
);
},
},
);
},
tinycomments_mode: 'embedded',
tinycomments_author: 'Author name',
mergetags_list: [
    {
        value: 'First.Name',
    },

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 111 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        title: 'First Name',
      },
      {
        value: 'Email',
        title: 'Email',
      },
    ],
    tinymceai_token_provider:
      async () => {
        const apiKey = import.meta
          .env
          .VITE_TINYMCE_API_KEY;
        await fetch(
          `https://demo.api.tiny.cloud/1/${apiKey}/auth/random`,
          {
            method: 'POST',
            credentials:
              'include',
          },
        );
        return {
          token: await fetch(
            `https://demo.api.tiny.cloud/1/${apiKey}/jwt/tinymceai`,
            {
              credentials:
                'include',
            },
          ).then((r) => r.text()),
        };
      },
    uploadcare_public_key:
      'b2bf888694ea7c74b2a5',
    content_style:
      'body { font-family:Helvetica,Arial,sans-serif; font-size:16px; line-height:1.6; }',
  }}
/>
</div>
{errors.content && (
  <p className="text-destructive pt-2 text-sm">
    {errors.content}
  </p>
)}
</div>

<div className="pt-6">
  <Button
    type="submit"
    className="w-full py-6 text-lg"
    disabled={processing}
  >
    <Save className="mr-2 h-5 w-5" />
    {processing
      ? 'Збереження...'
      : isEditing
      ? 'Оновити статтю'
      : 'Опублікувати статтю'}
  </Button>
</div>

```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 112 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```
        </Button>
    </div>
</form>
</CardContent>
</Card>
</div>
</AppLayout>
);
}
```

| | | | | | | |
|------|------|----------|--------|------|--------------------------|------|
| | | | | | ІПЗ.КР.Б – 121 – 26 - ПЗ | Арк. |
| | | | | | | 113 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |