МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Кафедра комп'ютерних наук та кібербезпеки

Бакалаврська робота

РОЗРОБКА ВЕБСАЙТУ ДЛЯ ПРОДАЖУ КАРТИН З ІНТЕГРАЦІЄЮ ПЛАТІЖНИХ СИСТЕМ ТА ВЕБФОРМАМИ

Виконав:
Русанович Андрій Андрійович,
студент групи КНІТ-44
факультету інформаційних
технологій і математики

Науковий керівник:
Крестьянполь Любов Юріївна,
кандидат тех. наук, доцент
кафедри прикладної логістики

Луцьк 2025

ЗМІСТ

ВСТУП **3**

РОЗДІЛ 1

РОЗРОБКА ПЕРСОНАЛЬНОГО САЙТУ **5**

1.1 Загальні підходи до розробки вебсайтів 5

1.2 Огляд та аналіз аналогічних програмних розробок вебсайтів 7

РОЗДІЛ 2

РОЗРОБКА ПЕРСОНАЛЬНОГО САЙТУ ХУДОЖНИКА ЗАСОБАМИ PHP **12**

2.1 Постановка задачі та призначення 12

2.2 Вибір моделі розробки програмного засобу 13

2.3 Загальний опис проекту 13

2.4 Обґрунтування вибору інструментів засобів розробки 14

2.5 Особливості програмної реалізації 16

2.6 Організація тестування та налагодження програмного засобу 17

2.7 Рекомендації по використанню та впровадженню програмного засобу .. 18

ВИСНОВКИ **20**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **22**

ДОДАТКИ **24**

ВСТУП

У сучасному цифровому просторі інтернет відіграє вирішальну роль у просуванні творчості та залученні аудиторії. Для художника власний вебсайт стає не лише онлайн-галереєю, а й ефективним засобом комунікації з поціновувачами мистецтва та потенційними клієнтами. Інтеграція сучасних технологій, зокрема хмарних сервісів, дозволяє зробити його більш функціональним і доступним.

Актуальність цієї теми обумовлена тим, що персональний сайт художника значно перевершує друковані каталоги та соцмережі. Він дає змогу зручно організовувати контент, розподіляючи роботи за жанрами, техніками чи напрямами, що покращує навігацію для відвідувачів. Завдяки розміщенню на хмарній платформі, такій як Amazon Web Services, сайт отримує високу швидкість завантаження, надійність та масштабованість. Крім того, можливість інтеграції платіжних систем, таких як PayPal, спрощує процес продажу картин та прийому індивідуальних замовлень.

Окрім цього, власний сайт забезпечує швидке оновлення інформації, охоплення широкої аудиторії без додаткових витрат на друк, ефективне просування через SEO та повний контроль над контентом без сторонньої реклами. Все це робить його незамінним інструментом для митців, які прагнуть розширити свою присутність в інтернеті та налагодити зручну взаємодію з покупцями.

Метою роботи є розробка персонального сайту художника, з інтеграцією платіжних систем та веб формами.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Розробити вебсайт із сучасним дизайном, реалізованим на базі фреймворку Laravel
2. Інтегрувати платіжну систему для оплати замовлення.
3. Розробити форму для замовлення картини згідно побажання замовника.

4. Розмістити веб сайт на хмарному сервісі Amazon Web Services.
5. Провести тестування розробленого вебсайту.
6. Забезпечити багатомовність ресурсу.

Забезпечити адаптацію сайту для різних пристроїв і SEO-оптимізацію

Об’єктом дослідження є хмарне середовище AWS, платіжна система Paypal та вебформи.

Предметом дослідження є технології та методи, які забезпечують роботу сайту в хмарному середовищі, інтеграцію з платіжною системою та побудову вебформ.

РОЗДІЛ 1
РОЗРОБКА ПЕРСОНАЛЬНОГО САЙТУ

1.1 Загальні підходи до розробки вебсайтів

Для розробки персонального сайту можна застосовувати різні архітектурні підходи, вибір яких залежить від цілей і масштабів проєкту. Важливим фактором є вибір між монолітною та мікросервісною архітектурою, а також рішення щодо розгортання – локально, у хмарних середовищах або в гібридних системах.

Монолітна архітектура передбачає єдину кодову базу, що включає в себе всі ключові компоненти сайту – інтерфейс користувача, бізнес-логіку та рівень доступу до даних. Такий підхід забезпечує простоту розробки та управління, оскільки всі елементи інтегровані в єдину систему. Перевагами моноліту є швидкий процес розгортання, легкість моніторингу та налагодження. Цей підхід особливо ефективний для невеликих і середніх проєктів, де важлива централізація управління. Проте, недоліки моноліту включають складність масштабування та оновлення, оскільки зміни в одній частині можуть впливати на всю систему.

На відміну від моноліту, мікросервісна архітектура ділить програмне забезпечення на незалежні сервіси, що взаємодіють через API. Це дозволяє масштабувати кожен сервіс окремо, що значно покращує продуктивність при високих навантаженнях. Мікросервіси також сприяють розподіленій розробці, адже різні команди можуть працювати над окремими частинами сайту паралельно. Головним викликом мікросервісного підходу є складність управління взаємодією між сервісами та підвищені вимоги до інфраструктури, що потребує ефективних інструментів моніторингу та оркестрації контейнерів, таких як Kubernetes чи Docker.

Щодо розгортання сайту, існують три основні варіанти: локальні сервери, хмарні рішення та гібридні інфраструктури. Локальні сервери забезпечують повний контроль над ресурсами, але вимагають значних витрат на обладнання, технічне обслуговування та безпеку. Хмарні рішення, зокрема AWS, Google

Cloud і Microsoft Azure, надають гнучкість, масштабованість і автоматизоване управління ресурсами, що особливо важливо для великих і динамічних проєктів. Гібридний підхід поєднує переваги обох варіантів, дозволяючи розміщувати критично важливі компоненти локально, а менш ресурсоємні завдання передавати у хмару.

Архітектурні рішення також впливають на роботу сайту з базами даних. SQL-бази, такі як MySQL або PostgreSQL, використовують структурований підхід і забезпечують високу узгодженість даних, що важливо для транзакційних систем. NoSQL-бази, такі як MongoDB чи Cassandra, пропонують гнучкість і горизонтальне масштабування, що робить їх ефективними для розподілених систем з великим обсягом інформації.

Ключовим аспектом веброзробки є взаємодія з платіжними системами. Сучасні сайти часто інтегрують онлайн-платежі через сервіси, такі як PayPal, Stripe або Square, які забезпечують швидке та безпечне проведення фінансових операцій. Для реалізації безпечних платежів використовуються протоколи шифрування, двофакторна автентифікація та токенізація, що мінімізує ризики шахрайства.

Серед популярних архітектурних патернів варто згадати MVC (Model-View-Controller), який поділяє програму на три основні компоненти: модель (обробка даних), представлення (користувацький інтерфейс) та контролер (зв’язок між ними). Така структура забезпечує розподіл функціональності та спрощує підтримку коду. Іншим поширеним рішенням є односторінкові додатки (SPA), які працюють у межах однієї сторінки та оновлюють вміст динамічно. Це підвищує швидкість завантаження і створює більш плавний користувацький досвід.

Server-Side Rendering (SSR) є ще одним підходом, який передбачає формування сторінки на сервері перед її передачею у браузер. Це покращує SEO та забезпечує швидке відображення контенту. SSR часто використовується у проєктах, де важливо забезпечити швидке завантаження та оптимізацію під пошукові системи.

Таким чином, вибір архітектури та технологій для розробки персонального сайту залежить від багатьох факторів: масштабованості, продуктивності, безпеки та інтеграції з іншими сервісами. Монолітна архітектура підходить для невеликих проєктів, мікросервісна – для масштабних систем. Локальні сервери надають контроль, а хмарні рішення – гнучкість та надійність. Використання SQL або NoSQL залежить від типу даних і вимог до продуктивності. Інтеграція платіжних систем робить сайт зручним для комерційного використання, а архітектурні патерни, такі як MVC чи SPA, впливають на користувацький досвід і швидкість роботи.

Ретельний аналіз цих факторів допоможе створити ефективний, швидкий та безпечний вебресурс, що відповідатиме сучасним вимогам ринку.

1.2 Огляд та аналіз аналогічних програмних розробок вебсайтів

Sotheby’s – одна з найвідоміших у світі аукціонних компаній, що спеціалізується на продажі предметів мистецтва, рідкісних антикварних виробів, ювелірних прикрас, ексклюзивних вин, колекційних автомобілів та інших предметів розкоші. Заснована в 1744 році в Лондоні, компанія за століття своєї діяльності здобула бездоганну репутацію та стала символом престижу у сфері мистецтва та аукціонної торгівлі.

Для ефективного управління своїми аукціонами Sotheby’s використовує комплексне програмне забезпечення, яке розробляється або внутрішнім IT-відділом компанії, або спеціалізованими технологічними партнерами. Деякі компоненти системи можуть створюватися сторонніми підрядниками, що дозволяє впроваджувати найсучасніші рішення у сфері цифрових технологій. Архітектура платформи є багаторівневою та включає веб-інтерфейс для онлайн-доступу, серверні рішення для обробки даних, мобільний додаток для користувачів, а також окреме програмне забезпечення для співробітників аукціонного дому.

Технологічна основа системи включає кілька мов програмування: JavaScript використовується для розробки веб-інтерфейсу та мобільних додатків,

Python – для бекенд-розробки та аналізу даних, C# – для створення десктопних і серверних рішень, а SQL застосовується для управління базами даних.

Функціональні можливості платформи включають створення детальних каталогів лотів, інструменти для організації та управління аукціонами, автоматизовану систему ставок у режимі реального часу, а також потужні аналітичні модулі для відстеження ринкових тенденцій і формування звітності. Крім того, інтегрована CRM-система дозволяє ефективно працювати з клієнтами, вести історію їхніх покупок і створювати персоналізовані пропозиції.

Головні переваги програмного забезпечення Sotheby’s полягають у високому рівні автоматизації, зручності користування, можливості масштабування системи відповідно до зростання обсягу лотів та учасників, а також у надійній системі безпеки, що гарантує захист даних. Водночас деякі аспекти роботи системи можуть становити виклики – висока вартість підтримки та розробки, складність налаштування й інтеграції з іншими платформами, необхідність постійного оновлення та забезпечення безперебійної роботи під час пікових навантажень.

Для клієнтів Sotheby’s передбачена можливість зручної онлайн-оплати через офіційний сайт, що дозволяє швидко та безпечно здійснювати фінансові операції. Крім того, компанія пропонує послугу створення картин за індивідуальним замовленням. Клієнт може подати опис бажаного твору мистецтва, а фахівці аукціонного дому підберуть художника або допоможуть знайти відповідний лот серед наявних пропозицій. Це дає змогу замовникам отримати унікальні витвори мистецтва, створені з урахуванням їхніх побажань і стилістичних уподобань.

Таким чином, Sotheby’s поєднує традиції аукціонної справи з інноваційними технологічними рішеннями, забезпечуючи ефективність, безпеку та зручність як для продавців, так і для покупців ексклюзивних предметів мистецтва та розкоші.

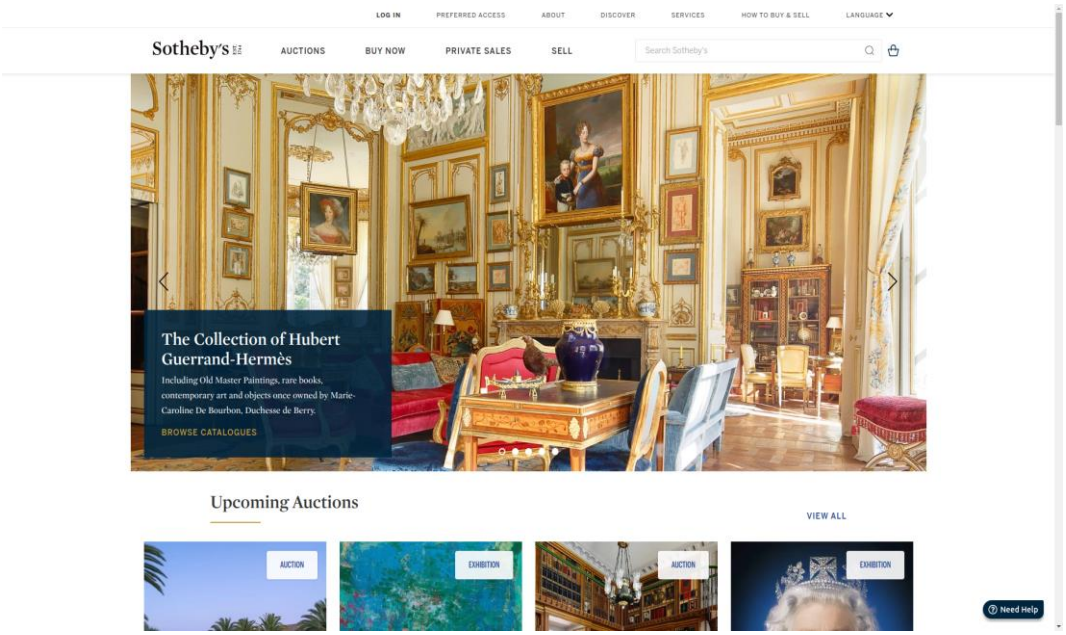

Рисунок 1.1 - Головна сторінка Sotheby's

Персональний сайт anninaroescheisen.com належить художниці Анніні Рошайзен, яка народилася у 1982 році в Розенгаймі, Німеччина, і проживає в Мюнхені та Нью-Йорку. Вона зосереджується на живописі та графіці, проте також експериментує зі скульптурою, інсталяціями, відео, фотографією та перформансами.

Розробка платформи Annina Roescheisen Art здійснювалася спеціалізованою студією, яка створює технологічні рішення для митців та творчих проєктів. Студія "Digital Canvas Solutions" має досвід у створенні інтерактивних платформ для представлення мистецтва. Сайт побудований на основі веб-додатка, що підтримує доступ через браузер і використовує клієнт-серверну архітектуру для обробки запитів користувачів і взаємодії з базою даних. Хоча дизайн оптимізовано для мобільних пристроїв, окремого мобільного додатка наразі не існує.

Для створення сайту використано кілька мов програмування: JavaScript забезпечує динамічні елементи інтерфейсу, Python застосовується для бекенд-розробки та обробки даних, PHP відповідає за генерацію контенту, а HTML і CSS створюють візуальний стиль та адаптивний дизайн.

Основний функціонал сайту включає інтерактивну галерею, що дозволяє переглядати роботи Анніни Рошайзен у високій якості з детальним описом. Крім того, сайт пропонує віртуальні виставки, що дають змогу відвідувачам зануритися у мистецький простір через 3D-експозиції. Інтеграція мультимедійного контенту, включаючи відео, аудіо та анімацію, допомагає розкрити концепції авторки. Також є можливість зв’язатися з художницею через контактну форму для замовлення робіт або обговорення співпраці. Блог містить статті про творчість, майбутні події та новини зі світу сучасного мистецтва.

Додатково сайт підтримує можливість оплати за замовлення творів мистецтва відповідно до побажань користувача. Оплата здійснюється через інтегровані платіжні системи, зокрема PayPal, Stripe та банківські перекази. Користувачі можуть вибирати зручний спосіб розрахунку, а також отримувати рахунки для оплати через електронну пошту. Для безпеки фінансових операцій застосовуються сучасні методи шифрування, двофакторна автентифікація та система токенізації платежів, що гарантує захист персональних даних клієнтів.

Перевагами платформи є простий та елегантний інтерфейс, що робить її доступною для широкої аудиторії, висока якість зображень і відеоматеріалів, які точно передають естетику робіт художниці, можливість організації віртуальних виставок для розширення глядацької аудиторії, зручна система управління контентом для оновлення інформації та адаптивний дизайн, що забезпечує коректну роботу на різних пристроях.

Недоліки включають відсутність мобільного додатка, що обмежує зручність перегляду для користувачів смартфонів, можливі технічні збої під час високого навантаження, несумісність із застарілими браузерами та певні обмеження щодо міжнародних платіжних операцій, залежно від країни користувача.

Ця платформа є важливим інструментом для популяризації мистецтва Анніни Рошайзен, поєднуючи сучасні цифрові технології з естетичним підходом до презентації творчості та зручною системою замовлення та оплати робіт.

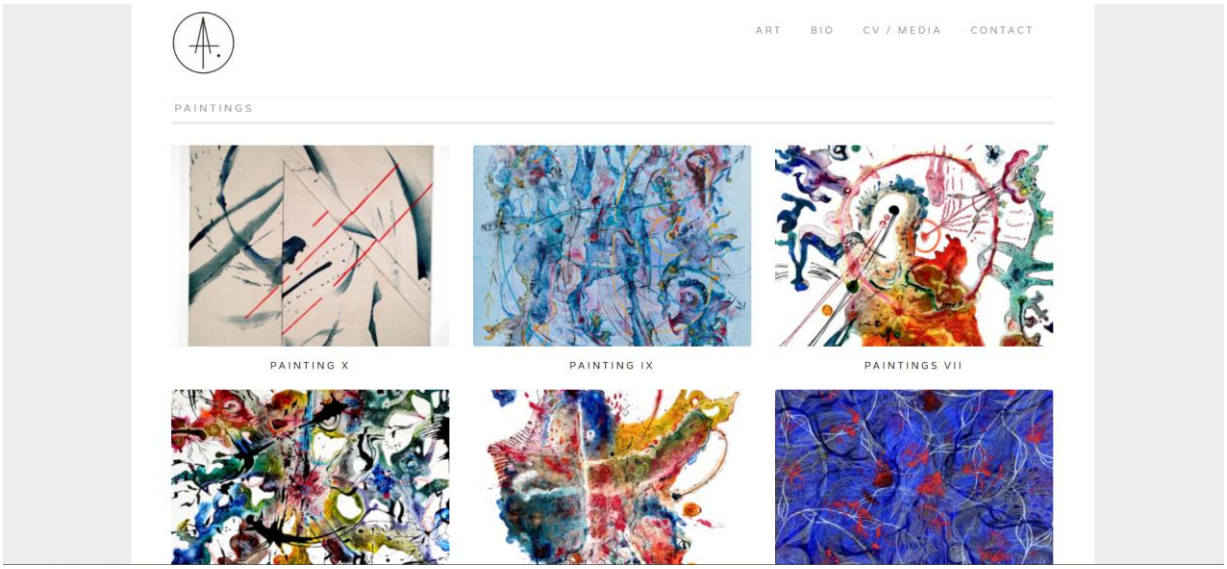

Рис. 1.2 - Головна сторінка anninaroescheisen.com

РОЗДІЛ 2
РОЗРОБКА ПЕРСОНАЛЬНОГО САЙТУ ХУДОЖНИКА ЗАСОБАМИ PHP

2.1 Постановка задачі та призначення

Метою створення персонального сайту художника є інтеграція платіжних систем та веб-форм для взаємодії з користувачами. Система забезпечить не лише презентацію творчості, а й можливість замовлення унікальних робіт за індивідуальними запитами. Відвідувачі зможуть не тільки переглядати портфоліо, а й заповнювати спеціальну форму, де вкажуть свої побажання щодо картини, виберуть розмір, стиль та кольорову гаму. Це дозволить художнику отримувати замовлення безпосередньо через сайт та ефективно комунікувати з клієнтами.

Для зручності користувачів інтегрована платіжна система PayPal, що дозволяє швидко та безпечно здійснювати оплату за картини та індивідуальні замовлення. Впровадження онлайн-платежів гарантує прозорість фінансових операцій і спрощує процес оформлення замовлень.

Хостинг сайту реалізовано через AWS, що забезпечує високу продуктивність, масштабованість та надійність зберігання даних. Інфраструктура включає автоматичне резервне копіювання, захист від перевантажень та швидке завантаження контенту незалежно від географічного розташування користувачів. Це дозволяє підтримувати стабільну роботу сайту навіть під час високого навантаження.

Система спрямована на спрощення процесу взаємодії між художником та аудиторією, пропонуючи інтуїтивно зрозумілий інтерфейс, швидку обробку запитів та ефективну організацію даних. Використання сучасних технологій дозволяє створити не просто веб-ресурс, а повноцінний інструмент для ведення мистецького бізнесу в цифровому середовищі.

2.2 Вибір моделі розробки програмного засобу

Для розробки веб-системи, яка демонструє творчі роботи художника, було обрано ітеративний підхід із застосуванням Kanban. Такий метод дозволяє поступово впроваджувати нові функції та вдосконалювати систему, адаптуючись до змін у вимогах. Гнучкість ітеративного процесу забезпечує стабільність і продуктивність на кожному етапі розробки.

Розробка проходить у циклах, кожен із яких включає аналіз вимог, проєктування, створення та тестування нових можливостей. Початковий етап передбачає реалізацію базової версії веб-системи, що поступово розширюється та оптимізується відповідно до зворотного зв’язку користувачів. Це дає змогу оперативно вносити коригування та покращувати функціонал без необхідності кардинальних змін в архітектурі продукту.

Методологія Kanban забезпечує ефективне управління завданнями та прозорість робочого процесу. Основним інструментом є візуалізація завдань за допомогою Kanban-дошки, де всі задачі розподілені за статусами: заплановані, у процесі виконання, на тестуванні та завершені. Контроль за кількістю одночасно виконуваних завдань сприяє оптимальному використанню ресурсів і дозволяє швидко адаптувати систему до нових потреб.

Застосування такого підходу дозволяє не лише зменшити затримки у розробці, а й покращити координацію між розробниками, тестувальниками та замовником. Поєднання ітеративної моделі та Kanban дає змогу ефективно впроваджувати оновлення, підтримувати стабільну роботу веб-системи та забезпечувати високу якість кінцевого продукту.

2.3 Загальний опис проекту

Розроблюваний веб-сайт є персоналізованою платформою, створеною для демонстрації творчих робіт художника та взаємодії з потенційними замовниками. Він надає можливість редагування контенту через адміністративну панель, що дозволяє митцю самостійно оновлювати інформацію про свої роботи, виставки та проєкти.

Окрім функції портфоліо, сайт містить інтегровану форму замовлення картин, де користувачі можуть детально описати бажане зображення – стиль, кольорову гаму, розмір та інші побажання. Це дає змогу художнику отримувати персоналізовані запити та реалізовувати унікальні творчі проєкти відповідно до побажань клієнтів.

Важливим аспектом є інтеграція платіжної системи PayPal, що дозволяє зручно та безпечно здійснювати оплату за замовлення. Це значно спрощує процес взаємодії між художником і покупцями, надаючи можливість швидких і захищених фінансових транзакцій.

Сайт розгорнутий на Amazon Web Services (AWS), що забезпечує високу продуктивність, масштабованість та надійність. Завдяки використанню хмарної інфраструктури AWS, ресурс отримує гнучкі можливості управління трафіком, ефективне кешування контенту та захист від можливих технічних збоїв.

Окрему увагу приділено SEO-оптимізації, що сприяє підвищенню видимості сайту у пошукових системах. Використання мета-тегів, оптимізованих URL-адрес та структурованих даних допомагає залучати нову аудиторію та покращує індексацію сторінок.

Для забезпечення швидкого завантаження контенту реалізовано автоматичну обробку зображень, адаптацію їх під різні пристрої та застосовано методи кешування. Це гарантує комфортний користувацький досвід та стабільну роботу сайту навіть при великій кількості відвідувачів.

Поєднання персоналізованого функціоналу, можливості замовлення картин за описом, інтегрованих платіжних систем і хмарного розгортання робить цей веб-сайт ефективним інструментом для розвитку творчості художника та залучення нових клієнтів.

2.4 Обґрунтування вибору інструментів засобів розробки

Для створення веб-сайту художника обрано сучасний технологічний стек, який поєднує високу продуктивність, зручність розробки та ефективне управління ресурсами. Основним серверним фреймворком став Laravel, що

забезпечує гнучку архітектуру, безпеку та зручний механізм роботи з базою даних через ORM Eloquent. Це дозволяє швидко реалізовувати необхідний функціонал, зокрема адміністративну панель для керування контентом.

Збереження та обробка даних здійснюється за допомогою MySQL, що забезпечує стабільність навіть при великих навантаженнях. Інтеграція Laravel з MySQL дозволяє створювати оптимізовані запити, пришвидшуючи доступ до інформації.

Для роботи з користувацьким інтерфейсом обрано Tailwind CSS, який завдяки утилітарному підходу дозволяє швидко розробляти адаптивний дизайн без необхідності написання кастомних стилів. Це спрощує підтримку сайту та подальше внесення змін у його зовнішній вигляд.

Оптимізація зображень виконується за допомогою утиліти cwebp, що дозволяє конвертувати файли у формат WebP. Завдяки цьому зменшується розмір графічного контенту без втрати якості, що позитивно впливає на швидкість завантаження сторінок.

Серверною платформою для розгортання обрано Amazon Web Services (AWS), що забезпечує масштабованість, надійність та високу доступність сайту. AWS дозволяє ефективно розподіляти навантаження, використовувати автоматизовані механізми резервного копіювання та забезпечує швидке реагування на зміни у трафіку.

Важливим функціональним елементом є інтеграція з платіжною системою PayPal, що надає можливість безпечної оплати безпосередньо через сайт. Це дозволяє користувачам не лише переглядати роботи художника, а й оформлювати замовлення на придбання картин або замовляти персоналізовані твори.

Для взаємодії з аудиторією реалізовано спеціальну форму, де відвідувачі можуть описати свої побажання щодо картини, вказавши стиль, кольорову гаму, розміри та інші деталі. Це сприяє створенню унікальних художніх робіт відповідно до індивідуальних запитів клієнтів.

Поєднання Laravel, MySQL, Tailwind CSS, cwebp, AWS та PayPal створює ефективну та гнучку систему, що забезпечує швидкодію, зручність адміністрування, безпечні фінансові операції та комфортну взаємодію з користувачами.

2.5 Особливості програмної реалізації

Розроблений веб-сайт художника є масштабованим та продуктивним рішенням, побудованим на Laravel, що забезпечує зручність керування контентом та взаємодії з користувачами. Архітектура передбачає хостинг на AWS EC2, що гарантує високу продуктивність і надійність, а для зберігання зображень використовується AWS S3, що дозволяє швидко завантажувати медіаконтент без перевантаження основного сервера. База даних розміщена на AWS Aurora, що забезпечує ефективне керування даними з мінімальними затримками та високою відмовостійкістю.

Для швидкої та якісної обробки зображень застосовується бібліотека Spatie Image, яка дозволяє змінювати розміри, оптимізувати якість та автоматично створювати варіації файлів у різних роздільних здатностях. Використання утиліти cwebp дає змогу конвертувати зображення у формат WebP, що значно зменшує їхній розмір без втрати якості, покращуючи швидкість завантаження сторінок та загальну продуктивність сайту.

Взаємодія користувачів із сайтом максимально спрощена завдяки інтеграції PayPal, що дозволяє здійснювати оплату через спеціально налаштовані посилання без необхідності реєстрації додаткових акаунтів. Це забезпечує зручність та безпеку фінансових транзакцій, що особливо важливо для продажу картин і приймання замовлень.

Веб-сайт підтримує багатомовність через вбудовані механізми Laravel, що дозволяє зберігати та відображати контент кількома мовами, адаптуючи його до потреб міжнародної аудиторії. Усі текстові ресурси зберігаються у мовних файлах, що полегшує додавання нових мов без значних змін у коді. База даних

також має структуру з підтримкою локалізованих даних, що дає змогу зручно керувати багатомовними описами картин та іншою текстовою інформацією.

Завдяки сучасним технологіям у фронтенді, таким як Tailwind CSS, CSS Grid та Vite, сайт залишається легким, швидким і адаптивним. Tailwind CSS дозволяє створювати гнучкі стилі без необхідності написання додаткового CSS-коду, а Vite оптимізує продуктивність, забезпечуючи мінімальний час завантаження сторінок.

Система безпеки побудована на базі вбудованих механізмів Laravel, що включають захист від SQL-ін’єкцій, XSS-атак та CSRF-загроз. Усі паролі користувачів шифруються за допомогою bcrypt або Argon2, що гарантує безпеку збереження даних. Додатковий рівень захисту забезпечує правильна конфігурація серверного середовища AWS, що передбачає налаштування прав доступу до ресурсів та використання шифрування даних.

Поєднання Laravel, AWS EC2, S3, Aurora, Tailwind CSS, Vite та сучасних рішень для обробки зображень дозволяє створити високопродуктивний, безпечний та зручний у використанні веб-сайт. Інтеграція PayPal забезпечує простоту та безпеку оплати, а підтримка багатомовності та ефективне керування контентом роблять цей ресурс ідеальним інструментом для художника, який прагне презентувати свої роботи світові.

2.6 Організація тестування та налагодження програмного засобу

Тестування веб-додатка охоплює як ручні, так і автоматизовані методи, що забезпечують комплексний контроль якості. Ручне тестування проводиться на стадії розробки, коли перевіряється відповідність функціональних можливостей вимогам, оцінюється зручність користування та відповідність дизайну макетам. Автоматизоване тестування реалізоване через вбудовані у Laravel інструменти, які дозволяють проводити юніт-тести, функціональні та інтеграційні перевірки.

Юніт-тестування виконується за допомогою PHPUnit і спрямоване на перевірку роботи окремих компонентів, таких як сервіси для обробки медіафайлів, механізми автентифікації та авторизації. Це допомагає виявити

логічні помилки в коді та забезпечити стабільну роботу кожного модуля. Функціональне тестування перевіряє взаємодію між модулями, зокрема користувацькі сценарії та відгук системи на дії користувачів. Використання Laravel Dusk дозволяє тестувати UI, а HTTP-тести забезпечують коректність роботи API.

Інтеграційне тестування спрямоване на перевірку взаємодії між базою даних, серверною частиною та клієнтським інтерфейсом. Для цього використовується SQLite у тестовому середовищі, що дозволяє швидко перевіряти роботу додатка без впливу на основну базу даних. API тестується за допомогою Postman та PHPUnit, що дозволяє виявити можливі проблеми з обробкою запитів і відповідей сервера.

Моніторинг та налагодження здійснюються за допомогою Laravel Telescope і Laravel Debugbar, які дозволяють відстежувати виконання запитів, продуктивність системи та помилки. Додатково інтегрований Sentry для збору інформації про збої в реальному часі. Логування реалізоване через стандартний механізм Laravel Logging із можливістю збереження критичних подій у файлах або зовнішніх сервісах, таких як Loggly та Papertrail.

Завдяки такому підходу забезпечується висока якість розробки, стабільність веб-додатка та зручність користування. Автоматизоване тестування та ефективний моніторинг дозволяють вчасно виявляти та виправляти помилки, що гарантує безперебійну роботу системи та комфорт для користувачів.

2.7 Рекомендації по використанню та впровадженню програмного засобу

Персональний сайт художника вже розміщений на AWS за адресою rusanovych.com.ua, що забезпечує його стабільну та безперебійну роботу. Оскільки сайт побудований на Laravel, його серверне середовище підтримує PHP 8.x, MySQL та всі необхідні інструменти для коректної роботи. Всі залежності налаштовані через Composer, а структура бази даних ініціалізована за допомогою міграцій.

Фронтенд використовує Vite у поєднанні з TailwindCSS, що забезпечує швидке завантаження та продуктивну роботу інтерфейсу. Для правильної оптимізації та збірки статичних ресурсів застосовується Node.js, а залежності встановлені через npm. Робота з медіаконтентом автоматизована завдяки утиліті cwebp, яка конвертує зображення у формат WebP, значно зменшуючи їхній розмір без втрати якості. Це дозволяє оптимізувати завантаження сторінок і покращити загальну продуктивність сайту.

Файли зберігаються та обробляються через Amazon S3, що дає змогу ефективно управляти ресурсами навіть при високих навантаженнях. База даних розгорнута на AWS Aurora, що забезпечує масштабованість і високу швидкість обробки запитів. Для підвищення продуктивності використовується кешування через Redis та Laravel Cache, що зменшує навантаження на базу даних і пришвидшує роботу сайту.

Щоб забезпечити безпечну передачу даних, встановлений SSL-сертифікат для шифрування трафіку. Автентифікація користувачів реалізована через Laravel Sanctum, що гарантує безпечний доступ до ресурсів. Для моніторингу роботи сервера та автоматичних резервних копій бази даних і медіафайлів налаштовані відповідні сервіси, що дозволяє підтримувати стабільну роботу платформи.

Для коректної роботи оплати необхідно відкрити фізичну особу-підприємця (ФОП) та заповнити дані на сайті PayPal. Це дозволить інтегрувати онлайн-оплати через систему PayPal, використовуючи згенеровані посилання для транзакцій. Після налаштування користувачі зможуть легко здійснювати покупки безпосередньо через сайт, що підвищить зручність взаємодії з платформою.

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було досягнуто поставленої мети — створено персональний вебсайт художника, який не лише презентує творчість, але й забезпечує інтерактивну взаємодію з потенційними клієнтами та спрощує процес продажу картин за допомогою сучасних цифрових технологій.

У вступі роботи було обґрунтовано, що персональний сайт митця в умовах цифрового суспільства стає важливим інструментом самопрезентації, просування і продажу. Відмова від традиційних друкованих каталогів на користь інтерактивного вебресурсу дозволяє митцю мати повний контроль над контентом, забезпечити швидке оновлення та використовувати переваги SEO-просування.

У процесі виконання роботи було вирішено такі завдання:

1. Розроблено функціональний вебсайт із сучасним дизайном, реалізованим на базі фреймворку Laravel із застосуванням Tailwind CSS, Vite та MySQL.

2. Інтегровано платіжну систему PayPal, що дозволяє безпечно та зручно оформлювати замовлення й здійснювати оплату безпосередньо на сайті.

3. Реалізовано вебформи для персоналізованого замовлення картин — з можливістю вибору стилю, розміру, кольору, прикріплення референсів тощо.

4. Сайт розгорнуто на хмарній платформі AWS з використанням EC2, S3, Aurora та інших сервісів для забезпечення масштабованості, надійності й високої продуктивності.

5. Проведено комплексне тестування (ручне та автоматизоване) функціоналу за допомогою Laravel Dusk, PHPUnit, Postman, а також впроваджено інструменти моніторингу (Laravel Telescope, Sentry).

6. Забезпечено багатомовність ресурсу та реалізовано механізми безпеки, включно з шифруванням, автентифікацією, авторизацією, захистом від SQL-ін’єкцій та XSS-атак.

7. Впроваджено оптимізацію зображень у формат WebP, адаптацію сайту для різних пристроїв і SEO-оптимізацію структури контенту.

На основі порівняльного аналізу аналогічних вебплатформ (Sotheby’s, Annina Roescheisen) зроблено висновки щодо найкращих практик у галузі онлайн-презентації та комерціалізації мистецтва, які частково були враховані в реалізованому проєкті.

Результатом проєкту є повноцінний сучасний вебресурс із продуманою архітектурою, зручним інтерфейсом, підтримкою адміністративної панелі, що дозволяє художнику самостійно оновлювати контент. Такий сайт не лише задовольняє потреби у презентації творчості, а й стає дієвим інструментом для реалізації бізнес-моделі у сфері мистецтва.

Перспективи подальшого розвитку:

- створення мобільного застосунку або впровадження PWA-версії;
- автоматизація розрахунку вартості персональних замовлень;
- інтеграція альтернативних платіжних систем;

Бакалаврська робота повністю реалізовує представлення інформації, охоплення широкої онлайн-аудиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що краще моноліт чи мікросервіси? | IAMPM. IAMPM. URL: https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu/ (дата звернення: 01.03.2025).

2. Cwebp | webp | google for developers. Google for Developers. URL: https://developers.google.com/speed/webp/docs/cwebp (дата звернення: 01.03.2025).

3. Frontend SSR: understanding server-side rendering and its importance. Verpex. URL: https://verpex.com/blog/website-tips/frontend-ssr-understanding-server-side-rendering-and-its-importance (дата звернення: 01.03.2025).

4. Introduction | image. Websites & webapplications in Laravel | Spatie. URL: https://spatie.be/docs/image/v3/introduction (дата звернення: 01.03.2025).

5. Laravel - the PHP framework for web artisans. Laravel - The PHP Framework For Web Artisans. URL: https://laravel.com/ (дата звернення: 01.03.2025).

6. Single-page applications: understanding the basics. webcodegenie.com. URL: https://webcodegenie.com/blog/single-page-applications/ (дата звернення: 01.03.2025).

7. SQL проти NoSQL – різниця між ними. Guru99. URL: https://www.guru99.com/uk/sql-vs-nosql.html (дата звернення: 01.03.2025).

8. Tailwind CSS - rapidly build modern websites without ever leaving your HTML. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. URL: https://tailwindcss.com/ (дата звернення: 01.03.2025).

9. Vite. vitejs. URL: https://vite.dev/ (дата звернення: 01.03.2025).

10. What is MVC? Advantages and disadvantages of MVC. interserver.net. URL: https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc (дата звернення: 01.03.2025).

11. AWS Documentation. docs.aws.amazon.com. URL: https://docs.aws.amazon.com/ (дата звернення: 01.03.2025).

12. A Simple and Safer Way to Pay and Get Paid | PayPal UA | PayPal UA. Pay, Send and Save Money with PayPal | PayPal DE. URL: https://www.paypal.com/ua/home (дата звернення: 01.03.2025).

ДОДАТКИ

Додаток А
Технічне завдання

1. Підстави для розробки

Підставою для розробки сайту є виконання бакалаврської роботи за темою "Розробка вебсайту для продажу картин з інтеграцією платіжних систем та вебформами", затвердженої на кафедрі комп'ютерних наук та кібербезпеки.

2. Призначення розробки

Сайт призначений для створення онлайн-каталогу картин із можливістю їх перегляду, покупки та оформлення індивідуальних замовлень. Він надає користувачам інформацію про художника, його роботи та виставки, а також інструменти адміністрування контенту.

3. Функціональні вимоги

3.1 Режим користувача

- Перегляд інформації про картину
- Перегляд інформації про виставку
- Перегляд картин за роками
- Перегляд інформації про художника
- Замовлення картини через форму запиту
- Оформлення індивідуального замовлення картини за описом клієнта
- Оплата замовлень через інтеграцію з PayPal

3.2 Режим адміністратора

- Редагування інформації про картини

- Редагування інформації про виставки
- Редагування інформації про художника
- Управління замовленнями картин та індивідуальними запитами
- Управління платежами через PayPal

4. Структура сайту

4.1 Основні сторінки сайту

Головна сторінка

- Слайдер з обраними картинами
- Скорочена інформація про художника (фото, текстовий опис)
- Попередній перегляд виставок із пагінацією

Сторінка художника

- Фото художника
- Детальний текстовий опис
- Перелік виставок з посиланнями

Сторінка перегляду картин за роками

- Фільтр картин за роками
- Галерея зображень картин

Сторінка виставки

- Інформація про виставку
- Перелік картин, представлених на виставці

Сторінка окремої картини

- Зображення картини
- Інформація про назву, рік створення, техніку виконання
- Форма запиту на купівлю картини

Сторінка індивідуального замовлення

- Форма введення опису замовлення
- Можливість прикріплення файлів (референси, ескізи)
- Вибір техніки виконання
- Розрахунок вартості та оформлення передоплати через PayPal

Адміністративна панель

- Управління інформацією про картини (додавання, редагування, видалення)
- Управління інформацією про виставки
- Редагування інформації про художника
- Перегляд та обробка замовлень
- Контроль платежів

5. Вимоги до інформаційної та програмної сумісності

5.1 Технології

- PHP 8.x
- Laravel
- MySQL
- Tailwind CSS
- cwebp
- Nginx
- AWS (хостинг, S3 для зберігання медіа)
- PayPal API для обробки платежів

5.2 Підтримувані браузери

- Google Chrome 110+
- Mozilla Firefox 110+
- Safari 16+ (macOS)

- Microsoft Edge 110+

5.3 Мовні версії сайту

- Українська (за замовчуванням)
- Англійська

6. Вимоги до розгортання

- Сайт розміщується на AWS
- Використання S3 для зберігання зображень картин
- Використання EC2 для основного сервера
- Автоматичне масштабування через AWS Load Balancer
- Налаштування резервного копіювання бази даних
- Наявність SSL-сертифіката

7. Вимоги до безпеки

- Автентифікація та авторизація через Laravel Sanctum або Passport
- Захист від SQL-ін'єкцій та XSS-атак
- Використання HTTPS для шифрування трафіку

8. Вимоги до програмної документації

- Наявність керівництва користувача
- Документація API
- UI Kit для стилізації інтерфейсу

9. Стадії і етапи розробки

1. Розробка технічного завдання
2. Створення діаграми прецедентів
3. Створення діаграми виконання
4. Створення wireframe та mockup

5. Розробка адміністративної панелі
6. Розробка користувацької частини сайту
7. Інтеграція з AWS та PayPal
8. Тестування та розгортання

Виконавець: Андрій Андрійович Русанович

Терміни: До дати захисту бакалаврської роботи

Додаток Б

Інструкція користувачу

Використання сайту

Головна сторінка

Після завантаження сайту ви потрапляєте на головну сторінку, де відображається добірка картин у вигляді слайдера, інформація про художника з його фотографією та коротким описом, а також анонси виставок із можливістю переходу на сторінку з детальними відомостями.

Перегляд картин

У розділі "Картини по роках" доступна галерея робіт, де можна переглядати твори, фільтрувати їх за роком створення та відкривати окрему сторінку кожної картини для отримання розширеної інформації.

Інформація про художника

У розділі "Про художника" представлена біографія митця, фотоматеріали, а також список виставок, у яких він брав участь.

Перегляд виставок

У секції "Виставки" доступна інформація про минулі та поточні експозиції, місце й час їх проведення, а також список представлених на виставках картин.

Запит на купівлю картини

Кожна картина має окрему сторінку з можливістю оформлення запиту на купівлю. Для цього необхідно натиснути кнопку "Запит на купівлю", заповнити контактні дані (ім'я, email, телефон), за бажанням додати коментар і натиснути "Надіслати". Після відправки адміністрація сайту зв’яжеться з вами для уточнення деталей.

Використання адміністративної панелі

Вхід в адмін-панель

Для управління контентом передбачено окремий розділ сайту, доступний тільки адміністраторам. Вхід здійснюється через спеціальну сторінку, де необхідно ввести логін та пароль, а потім натиснути "Увійти".

Управління картинами

Адміністратор має змогу додавати нові роботи, завантажуючи зображення та заповнюючи необхідні дані, редагувати інформацію про вже додані картини, а також видаляти непотрібні записи.

Управління виставками

Для адміністрування розділу виставок передбачена можливість додавання нових подій із зазначенням опису та дат проведення, редагування існуючих записів і видалення застарілих даних.

Управління інформацією про художника

Адміністратор може оновлювати фотографії, змінювати біографічні дані та додавати інформацію про нові досягнення й нагороди.

Важлива інформація щодо оплати

Для налаштування прийому платежів необхідно зареєструвати ФОП та заповнити дані на сайті PayPal. Це дозволить активувати функцію прийому платежів, після чого користувачі зможуть оформлювати покупки картин безпосередньо через сайт.

Додаток В

Програмна реалізація сервісу ImageService

```
<?php

namespace App\Services\ImageService;

use App\Contracts\ImageService;
use App\Enums\ImageSizing;
use App\Exceptions\ImageServiceException;
use App\Facades\AppPublicDiskFacade as AppPublicDisk;
use App\Models\Image;
use App\Models\ImageVariation;
use App\Traits\UuidFileNameGenerator;
use Exception;
use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;
use InvalidArgumentException;
use Spatie\Image\Exceptions\InvalidManipulation;
use Spatie\Image\Image as SpatieImageUtils;

class ImageServiceImpl implements ImageService
{

  use UuidFileNameGenerator;

  public function removeImage(Image $image): void
  {
     foreach ($image->imageVariations as $imageVariation) {
```

```
                AppPublicDisk::getStorageDisk()->delete($imageVariation->file_path_key);
            }
            $image->imageVariations()->delete();
            $image->delete();
        }

        /**
         * @throws ImageServiceException
         */
        public function storeUploadedImage(UploadedFile $imageFile, array $imageSizingArr): Image
        {
            //validate 'imageSizingArr' input parameter
            foreach ($imageSizingArr as $sizing) {
                if (!($sizing instanceof ImageSizing)) {
                    throw new InvalidArgumentException('$imageSizingArr must be type of ImageSizing[]');
                }
            }

            $tempStoredOriginalImage = $this->storeImgInTempStorage($imageFile);

            $image = Image::create();

            foreach ($imageSizingArr as $imageSizing) {

                $tmpStoredProcessedImage = $this->resizeAndOptimizeImage($tempStoredOriginalImage, $imageSizing);
```

```
            $storedImageFilePathKey = $this->storeImgInPermanentStorage($tmpStoredProcessedImage, $imageSizing);

            $image->imageVariations()->save(new ImageVariation([
                'sizing_type' => $imageSizing->getSizingType(),
                'file_path_key' => $storedImageFilePathKey,
                'width' => $tmpStoredProcessedImage->widthPx,
                'height' => $tmpStoredProcessedImage->heightPx
            ]));
        }

        $storedImageFilePathKey = $this->storeImgInPermanentStorage($tempStoredOriginalImage, ImageSizing::Original);

        $image->imageVariations()->save(new ImageVariation([
            'sizing_type' => ImageSizing::Original->getSizingType(),
            'file_path_key' => $storedImageFilePathKey,
            'width' => $tempStoredOriginalImage->widthPx,
            'height' => $tempStoredOriginalImage->heightPx
        ]));

        return $image;
    }

    private function storeImgInPermanentStorage(TempStoredImage $tempStoredImage, ImageSizing $imageSizing): string
    {

        $publicPath = 'images/' . $imageSizing->getSizingType() . '/' . $tempStoredImage->getFullName();
```

```
            // Specify the 'public' to set the correct permissions
            AppPublicDisk::getStorageDisk()->put($publicPath,
file_get_contents($tempStoredImage->getAbsolutePath()));

            Storage::disk('local')->delete($tempStoredImage->relativePath);

            return $publicPath;
        }

        /**
         * @throws ImageServiceException
         */
        private function storeImgInTempStorage(UploadedFile $file):
TempStoredImage
        {
            $extension = $file->extension();

            $filename = $this->generateFileSystemSafeUUID();

            $tempStoredImageRelativePath = $this-
>getTempStoredImageRelativePathOnLocalDisk($filename, $extension,
ImageSizing::Original);

            //make absolute path for output image
            $tempStoredImageAbsolutePath =
Storage::path($tempStoredImageRelativePath);

            $stored = Storage::disk('local')->put($tempStoredImageRelativePath,
file_get_contents($file));
```

```
            if (!$stored) {
                throw new ImageServiceException("Failed attempt to store an image " .
$file->getFilename());
            }

            try {
                $spatieTemStoredImage =
SpatieImageUtils::load($tempStoredImageAbsolutePath);
                $temStoredImageWidth = $spatieTemStoredImage->getWidth(); // width
in px
                $temStoredImageHeight = $spatieTemStoredImage->getHeight(); //
height in px
            } catch (Exception $e) {
                throw new ImageServiceException($e);
            }


            return new TempStoredImage($tempStoredImageRelativePath, $filename,
$extension, $temStoredImageWidth, $temStoredImageHeight);
        }

        /**
         * Run resizing and optimization on original image
         * @throws ImageServiceException
         */
        private function resizeAndOptimizeImage(TempStoredImage
$tmpStoredImage, ImageSizing $imageSizing): TempStoredImage
        {
            //construct relative path for output image
```

```
            $tempStoredImageRelativePath = $this->getTempStoredImageRelativePathOnLocalDisk($tmpStoredImage->fileName, 'webp', $imageSizing);
            //make absolute path for output image
            $tempStoredImageAbsolutePath = Storage::path($tempStoredImageRelativePath);

            //create dir for output image if not exist
            $this->createDirForSizingIfNotExist($imageSizing);

            //perform transformations and save
            try {
                SpatieImageUtils::load($tmpStoredImage->getAbsolutePath())
                    ->width($imageSizing->getWidth())
                    ->height($imageSizing->getHeigth())
                    ->optimize()
                    ->save($tempStoredImageAbsolutePath);

                $spatieTemStoredImage = SpatieImageUtils::load($tempStoredImageAbsolutePath);
                $temStoredImageWidth = $spatieTemStoredImage->getWidth(); // width in px
                $temStoredImageHeight = $spatieTemStoredImage->getHeight(); // height in px
            } catch (InvalidManipulation|Exception $e) {
                throw new ImageServiceException($e);
            }
```

```
            return new TempStoredImage($tempStoredImageRelativePath, $tmpStoredImage->fileName, 'webp', $temStoredImageWidth, $temStoredImageHeight);
        }

        private function getTempStoredImageRelativePathOnLocalDisk(string $name, string $extension, ImageSizing $imageSizing): string
        {
            return TempStoredImage::TEMP_IMAGE_FOLDER_PATH_ON_LOCAL_DISK . $imageSizing->getSizingType() . '/' . $name . '.' . $extension;
        }

        private function createDirForSizingIfNotExist(ImageSizing $imageSizing): void
        {
            $dirForSizing = TempStoredImage::TEMP_IMAGE_FOLDER_PATH_ON_LOCAL_DISK . $imageSizing->getSizingType();
            if (!Storage::directoryExists($dirForSizing)) {
                Storage::makeDirectory($dirForSizing);
            }
        }
    }
```