

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 «Інженерія програмного забезпечення»)
на тему:

«Вебзастосунок для управління контентом візажистів»

Виконав студент групи ВТ-22-1
ТАРАСЮК Марія Олександрівна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2026

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

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ТАРАСІЮК Марія Олександрівна**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **« Вебзастосунок для управління контентом візажистів»**,

затверджена Наказом закладу вищої освіти від **«28» лютого 2026 р., №92/с**

Вихідні дані для роботи: **Об'єктом дослідження є управління контентом візажистів із використанням сучасних вебтехнологій та інтелектуальних систем підтримки користувачів. Дослідження охоплює особливості автоматизації взаємодії між клієнтами та студією, а також застосування AI-консультанта для надання рекомендацій**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.В.	19.02.2026р.	19.02.2026р.
2	Чижмотря О.В.	01.03.2026р.	01.03.2026р.
3	Чижмотря О.В.	12.04.2026р.	12.04.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки, що містить вступ, три розділи, висновки, список використаних джерел (32 найменування), один додаток, а також 13 таблиць і 30 ілюстрацій. Загальний обсяг роботи становить 125 сторінок, з яких 86 сторінок припадає на основний текст.

Метою роботи є проектування та розробка вебзастосунку студії візажистів з інтегрованим AI-консультантом, який забезпечує автоматизацію онлайн-запису, підтримку клієнтських сесій у реальному часі, публікацію та коментування експертних статей, а також підвищення якості обслуговування за рахунок технологій штучного інтелекту. Для досягнення мети виконано аналіз предметної області, визначено технічні вимоги, обґрунтовано стек технологій та спроектовано NoSQL-базу даних для збереження інформації про користувачів, бронювання та діалоги. Розроблено серверну частину з JWT-автентифікацією, клієнтський інтерфейс, інтегровано Cloudinary для роботи з медіаконтентом та Nodemailer для відправки сповіщень. Також впроваджено AI-консультанта на базі OpenAI API з підтримкою контексту діалогу. Проведено функціональне тестування основних модулів системи.

У роботі досліджено предметну область, проаналізовано аналогічні рішення та сформовано вимоги до системи. Реалізовано архітектуру вебзастосунку та AI-консультанта для персоналізованих рекомендацій. Здійснено тестування та оцінку функціональності програмного комплексу. Вебзастосунок має інтуїтивний інтерфейс, що дозволяє користувачу (Reader) знаходити та стежити за авторами (Writer), переглядати та коментувати статті, а також записуватися на консультації.

Ключові слова: ВЕБЗАСТОСУНОК, ВІЗАЖИСТИ, REST API, АВТЕНТИФІКАЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧ

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Тарасюк М.О.			Вебзастосунок для управління контентом візажистів	Літ.	Арк.	Аркушів
Керівник		Чижмотря О.В.					3	125
						Житомирська політехніка, група ВТ-22-1		
Рецензент		Морозов А.В.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of a software system and an explanatory note, which includes an introduction, three chapters, conclusions, a list of references (32 sources), one appendix, as well as 13 tables and 30 figures. The total volume of the work is 125 pages, of which 86 pages constitute the main text.

The aim of the thesis is to design and develop a web application for a makeup studio with an integrated AI consultant, which provides automation of online appointment booking, support for real-time client sessions, publication and commenting of expert articles, and improvement of service quality through artificial intelligence technologies. To achieve this goal, the subject area was analyzed, technical requirements were defined, the technology stack was justified, and a NoSQL database was designed to store information about users, bookings, and conversations. The server-side was developed with JWT authentication, along with a client interface. Cloudinary was integrated for media content management, and Nodemailer was used for sending notifications. An AI consultant based on the OpenAI API was also implemented with support for conversational context. Functional testing of the main system modules was carried out.

The study examined the subject area, analyzed similar existing solutions, and defined system requirements. The web application architecture and the AI consultant for personalized recommendations were implemented. System testing and evaluation of functionality were performed. The web application features an intuitive interface that allows the user (Reader) to find and follow authors (Writer), view and comment on articles, and book consultations.

Keywords: WEB APPLICATION, MAKEUP STUDIO, ONLINE BOOKING, AI CONSULTANT, REST API, AUTHENTICATION, USER INTERFACE.

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ, ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	11
1.1. Постановка задачі	11
1.2. Аналіз аналогів програмного продукту.....	13
1.3. Архітектурна система.....	29
1.4. Обґрунтування вибору засобів реалізації.....	32
Висновки до першого розділу	35
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	38
2.1. Вимоги до системи	38
2.2. Розробка бази даних системи	45
2.3. Проєктування та реалізація алгоритмів роботи системи.....	56
2.4. Реалізація функціоналу платформи	59
Висновки до другого розділу	63
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ.....	64
3.1. Порядок встановлення та налаштування параметрів системи	64
3.2. Структура інтерфейсу	67
3.3. Тестування роботи вебзастосунку	82
Висновки до третього розділу	84
ВИСНОВКИ	85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	87
ДОДАТКИ.....	90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface, програмний інтерфейс застосунку

AI - Artificial Intelligence, штучний інтелект

CRUD - Create, Read, Update, Delete, основні операції створення, читання, оновлення та видалення даних

JSON - JavaScript Object Notation, текстовий формат обміну даними

JWT - JSON Web Token, стандарт передачі даних для автентифікації та авторизації користувачів

REST - Representational State Transfer, архітектурний стиль побудови вебсервісів

REST API - програмний інтерфейс, побудований відповідно до принципів REST

SQL - Structured Query Language, мова структурованих запитів до баз даних

UI - User Interface, інтерфейс користувача

UX - User Experience, користувацький досвід

URL - Uniform Resource Locator, уніфікований локатор ресурсу

DB – Database, база даних.

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

ВСТУП

Актуальність теми дослідження. Стрімкий розвиток інформаційних технологій та цифровізація сфери послуг зумовлюють потребу у впровадженні сучасних вебрішень для автоматизації взаємодії з клієнтами. Для студій візажу наявність функціонального вебзастосунку забезпечує зручний доступ до інформації про послуги, можливість онлайн-запису та ефективну комунікацію з користувачами (Reader). Впровадження AI-консультанта додатково сприяє персоналізації обслуговування та підвищенню якості клієнтського сервісу.

Особливого значення в контексті еволюції програмних продуктів набуває інтеграція технологій штучного інтелекту, що дозволяють ефективно оптимізувати комунікаційні процеси. Застосування інтелектуального AI-консультанта забезпечує автоматизацію первинного консультування користувачів у режимі реального часу. Такий модуль здатний оперативно генерувати відповіді на поширені запитання, а також надавати обґрунтовані рекомендації щодо вибору б'юті-послуг, стилів макіяжу та доглядових засобів, що суттєво зменшує часові витрати на обробку запитів і підвищує лояльність клієнтів. Важливим вектором розвитку сучасного вебдизайну та розробки є орієнтація на глибоку персоналізацію онлайн-сервісів. Використання алгоритмів штучного інтелекту на базі розроблюваного вебзастосунку дозволяє успішно реалізувати індивідуальний підхід до кожного відвідувача, створюючи комфортне та адаптивне інформаційне середовище для взаємодії зі студією. Проектування та програмна реалізація багатфункціонального вебзастосунку, який об'єднує платформу пошуку візажистів, систему онлайн-запису до майстрів, інтерактивний вебзастосунок та інтелектуального AI-консультанта, є актуальним практичним завданням. Воно поєднує сучасні вебтехнології з інструментами штучного інтелекту, сприяючи автоматизації процесів взаємодії з клієнтами та оптимізації діяльності студії візажистів.

Метою кваліфікаційної роботи є проектування та розробка вебзастосунку студії візажистів, який інтегрує платформу пошуку майстрів,

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

систему онлайн-запису на консультації, інтерактивний вебзастосунок та вбудованого AI-консультанта для забезпечення комплексної взаємодії між клієнтами та студією, надання персоналізованих рекомендацій щодо макіяжу, а також підвищення якості обслуговування користувачів за рахунок використання технологій штучного інтелекту.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

- Проаналізувати предметну область, дослідити наявні програмні рішення-аналоги та вебплатформи у б'юті-індустрії, виявити їхні переваги й недоліки та обґрунтувати доцільність розробки власного вебзастосунку.
- Визначити функціональні та нефункціональні вимоги до вебзастосунку, сформулювати архітектуру клієнт-серверної взаємодії та обґрунтувати вибір сучасного технологічного стеку розробки (React, Node.js, TypeScript, MongoDB).
- Спроектувати та реалізувати інфологічну модель бази даних NoSQL для оптимального збереження й структурування інформації про користувачів, медіа-портфолію візажистів, статті та коментарі, сесії записів і діалоги.
- Розробити серверну частину (Backend) застосунку, реалізувавши модульну логіку обробки HTTP-запитів, бізнес-логіку онлайн-бронювання, алгоритми створення публікацій та менеджменту коментарів у блозі, а також механізми безпечної автентифікації користувачів на базі JWT та захищених файлів cookies.
- Розробити клієнтську частину (Frontend) із адаптивним інтерфейсом користувача (із підтримкою світлої та темної тем) на основі методології Mobile-FirstS для коректного відображення функціоналу пошуку майстрів, онлайн-запису, інтерактивного чату та вебзастосунків на різних типах пристроїв.
- Інтегрувати сторонні мікросервіси та API, зокрема хмарне сховище Cloudinary для оптимізації та доставки медіаконтенту портфолію та ілюстрацій до статей, а також бібліотеку Nodemailer для автоматичної SMTP-генерації тригерних сповіщень про підтвердження записів.

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

- Реалізувати інноваційний інтелектуальний модуль AI-консультанта на базі OpenAI API з розробкою алгоритму збереження контексту діалогу в реальному часі для надання персоналізованих б'юті-рекомендацій клієнтам.
- Провести комплексне функціональне тестування розробленого програмного забезпечення для верифікації працездатності, надійності та відповідності системи всім встановленим технічним вимогам.

Об'єктом дослідження є процес розробки інтерактивного вебзастосунку для студії візажу, що включає створення інтерфейсу користувача (Reader), серверної частини, модулів й онлайн-запису та організацію взаємодії між ними через API.

Предметом дослідження є процеси, методи та програмно-технологічні засоби проєктування й розробки вебплатформи студії візажу, зокрема архітектурні рішення клієнт-серверної взаємодії, логіка ведення та коментування статей, механізми автентифікації користувачів (Reader), а також інтеграція сторонніх API-сервісів та інструментів штучного інтелекту.

Методи дослідження. Для досягнення поставленої мети застосовано комплекс теоретичних методів, зокрема системний та порівняльний аналіз науково-технічних джерел для дослідження аналогів вебзастосунків у б'юті-індустрії та обґрунтування технологічного стеку розробки. Практична частина роботи базується на інженерних методах моделювання програмних систем, парадигмі об'єктно-орієнтованого та модульного програмування, методах штучного інтелекту для налаштування AI-консультанта, із подальшим застосуванням методів функціонального тестування для верифікації системи.

Практичне значення отриманих результатів полягає у розробці та впровадженні багатофункціонального вебзастосунку студії візажу з інтегрованою системою онлайн-бронювання, вебплатформи та AI-консультантом для автоматизації комунікації з клієнтами. Створений програмний комплекс дозволяє підвищити якість обслуговування користувачів (Reader) через персоналізацію б'юті-рекомендацій, спростити пошук майстрів та

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

оптимізувати процес запису. Результати роботи мають практичну цінність, оскільки розроблена модульна архітектура системи може бути легко адаптована для інших підприємств сфери послуг, що потребують автоматизації клієнтського запису та онлайн-взаємодії.

Апробація. За темою кваліфікаційної роботи було опубліковано 2 тези конференцій [1, 2].

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

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

1.1. Постановка задачі

У сучасних умовах розвитку цифрових технологій сфера краси активно використовує вебрішення для взаємодії з клієнтами. Наявність вебзастосунку дозволяє студіям візажистів ефективно презентувати свої послуги, розширювати клієнтську базу та забезпечувати зручний доступ до інформації про діяльність студії.

Незважаючи на велику кількість вебресурсів у сфері краси, багато з них мають обмежений функціонал та не забезпечують належного рівня автоматизації процесів обслуговування клієнтів. Зокрема, користувачі (Reader) часто стикаються зі складністю отримання оперативної консультації щодо послуг, вартості або особливостей роботи студії.

Одним із перспективних напрямів удосконалення вебзастосунків є використання технологій штучного інтелекту. AI-консультант дозволяє автоматизувати процес спілкування з клієнтами, надавати відповіді на поширені запитання та допомагати користувачам (Reader) швидко знаходити необхідну інформацію.

Використання AI-консультанта сприяє зменшенню навантаження на адміністраторів студії та підвищує якість обслуговування клієнтів. Завдяки автоматизованій підтримці користувачі (Reader) можуть отримувати консультації у будь-який час без необхідності очікування відповіді від персоналу.

Для забезпечення ефективної роботи студії необхідно реалізувати систему керування послугами, майстрами, записами клієнтів та інформаційним наповненням вебзастосунку. Важливим аспектом є створення зручного інтерфейсу як для відвідувачів сайту, так і для адміністратора системи (Admin).

Для ефективного зберігання та обробки даних необхідно розробити структуру бази даних, яка забезпечуватиме збереження інформації про користувачів, послуги, записи та результати взаємодії з AI-консультантом.

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

Особливу увагу слід приділити реалізації AI-консультанта, який повинен забезпечувати швидке отримання відповідей на запити користувачів (Reader) та надавати рекомендації щодо вибору послуг відповідно до потреб клієнта.

Отже, основною задачею кваліфікаційної роботи є проєктування та розробка вебзастосунку студії візажистів з інтегрованим AI-консультантом, яка забезпечить автоматизацію взаємодії з клієнтами, зручне керування записами на послуги та підвищення ефективності роботи студії. Для реалізації поставленої задачі необхідно виконати аналіз предметної області, спроектувати архітектуру системи та базу даних, розробити користувацький інтерфейс, реалізувати функціонал вебзастосунку й інтегрувати AI-консультанта для підтримки користувачів (Reader).

Значною перевагою розроблюваного вебзастосунку є можливість централізованого керування інформацією про візажистів (Writer), їхні послуги та портфоліо. Завдяки цьому відвідувачі платформи отримують актуальні дані про доступних спеціалістів, мають змогу детально вивчити приклади виконаних робіт, а також обрати майстра відповідно до власних потреб і естетичних вподобань. Наявність персональних цифрових профілів візажистів (Writer) не лише сприяє підвищенню довіри з боку клієнтів.

Використання передових інтернет-технологій гарантує високу адаптивність та інклюзивність програмного комплексу. Взаємодія з вебресурсом може здійснюватися як із персональних комп'ютерів, так і з мобільних пристроїв смартфонів чи планшетів без жодної втрати початкової функціональності або швидкодії інтерфейсу. Таке рішення суттєво підвищує загальний рівень зручності та забезпечує безперебійний доступ до всіх сервісів платформи незалежно від географічного розташування користувача (Reader).

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

1.2. Аналіз аналогів програмного продукту

Сучасний ринок програмного забезпечення у сфері beauty-індустрії переважно представлений спеціалізованими платформами, що орієнтовані на пошук майстрів, демонстрацію їхнього портфоліо та автоматизацію онлайн-запису на послуги. Подібні сервіси зазвичай інтегрують інструменти для тайм-менеджменту (керування розкладом), ведення клієнтських баз (CRM-модулі) та моніторингу бронювань, що суттєво спрощує інтерактивну взаємодію між візажистами та кінцевими споживачами.

Проте більшість існуючих систем є або занадто універсальними (орієнтованими на весь сектор послуг без врахування специфіки візажу), або не повною мірою задовольняють потреби користувачів (Reader) у гнучкому функціоналі та оперативній інтелектуальній комунікації. З метою виявлення переваг і недоліків наявного інструментарію та обґрунтування вимог до розроблюваного вебзастосунку, для подальшого порівняльного аналізу було обрано сервіси «Backstage MUA», «MakeUp Hub» та «GlossGenius» як найбільш релевантні програмні рішення у досліджуваній предметній області.

Першим аналогом для дослідження обрано міжнародний вебзастосунок «Backstage MUA» (рис.1.1), яка забезпечує взаємодію між візажистами та клієнтами. Головною перевагою сервісу є чітка структуризація послуг за категоріями, наявність цифрових портфоліо та інструментів для онлайн-бронювання.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

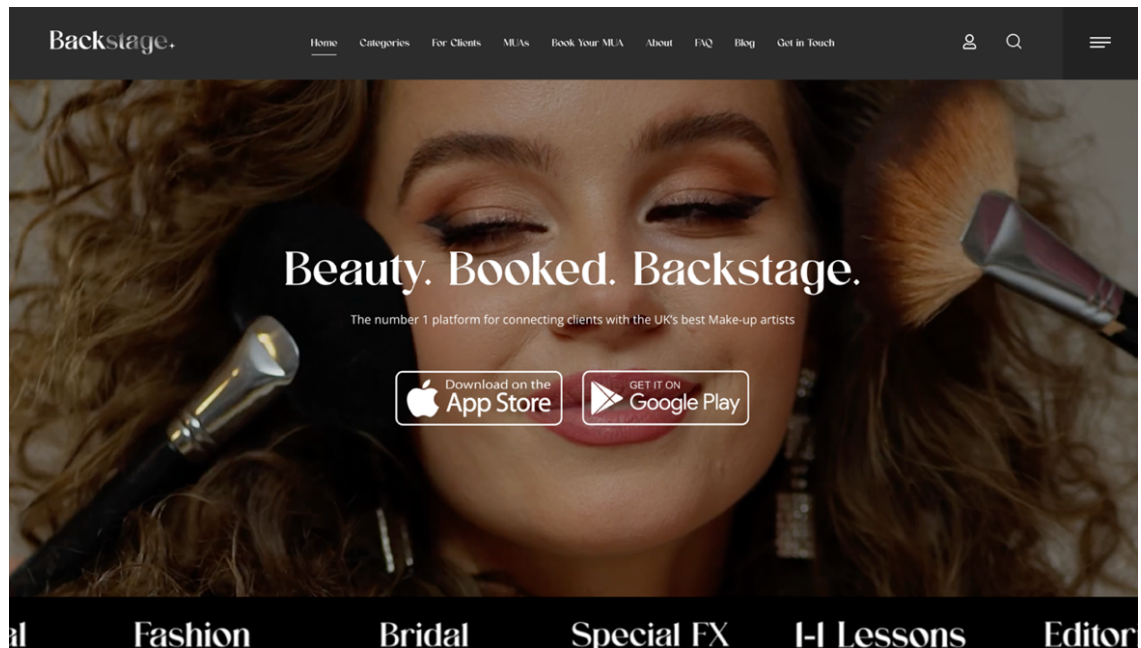


Рис. 1.1. Інтерфейс платформи Backstage MUA

Головна сторінка вебзастосунку розроблена на основі компонентного підходу для забезпечення високої швидкості завантаження та адаптивності інтерфейсу під мобільні пристрої. Панель навігації у верхній частині екрана містить логотип, елементи пошуку, автентифікації користувачів та інтерактивне меню для переходу між основними розділами системи.

Презентаційний блок відображає фонове тематичне зображення, головне маркетингове гасло та кнопки-посилання для завантаження мобільних застосунків. Модуль категорій послуг реалізований у вигляді клікабельного меню внизу екрана, яке дозволяє динамічно фільтрувати майстрів через REST API-запити без перезавантаження сторінки. Компонент AI-консультанта являє собою інтегрований віджет на основі технологій штучного інтелекту, який забезпечує асинхронний обмін даними із сервером та API OpenAI для генерації персоналізованих рекомендацій і відповідей на запити користувачів у режимі реального часу.

Важливим компонентом клієнтської частини системи є модуль каталогу спеціалістів, який відповідає за динамічну взаємодію з базою даних та фільтрацію контенту. На рис. 1.2 зображено інтерфейс сторінки з

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

представленими візажистами, яка реалізує функціонал пошуку та фільтрації майстрів за заданими критеріями.

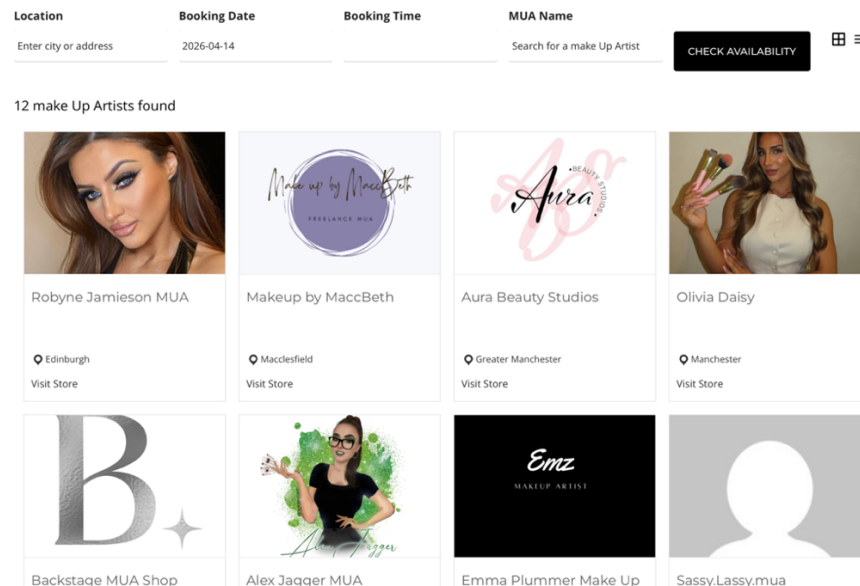


Рис. 1.2. Сторінка з представленими візажистами

Для забезпечення зручного вибору майстрів у системі реалізовано динамічний інтерфейс пошукової сторінки. На рис. 1.2 зображено головне екранне вікно каталогу, яке містить інтерактивну панель фільтрів у верхній частині для вибору локації, дати й часу запису, а також поле для введення імені конкретного спеціаліста. Натискання кнопки перевірки доступності активує асинхронний REST API-запит до серверної частини для миттєвого оновлення результатів без перезавантаження сторінки.

Основна область інтерфейсу відведена під сітку карток візажистів, де відображаються їхні логотипи, професійні імена, геолокація та посилання для переходу до детального профілю. Програмне рішення також підтримує автоматичний підрахунок кількості знайдених спеціалістів та можливість зміни режиму відображення списку.

Для забезпечення детального ознайомлення з послугами конкретного майстра та автоматизації процесу замовлення в системі реалізовано персональний профіль спеціаліста. На рис. 1.3 зображено інтерфейс персональної сторінки візажиста, який містить інформаційний банер з

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

фотографією, професійним описом досвіду, рейтингом та контактними даними майстра.

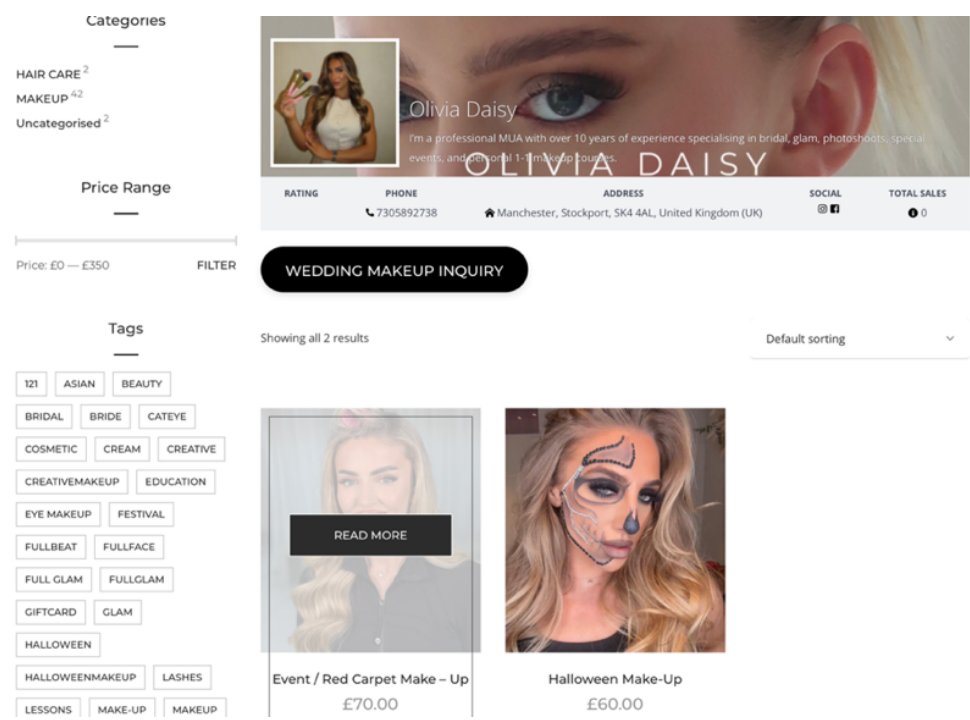


Рис. 1.3. Персональна сторінка візажиста

Простір персонального профілю організовано за принципом модульного розподілу інформаційних та керуючих зон. З лівого боку користувачеві доступна розвинена панель інструментів для кастомізації пошуку, де за допомогою вибору категорій, гнучкого налаштування цінового інтервалу через повзунок та активації специфічних тегів можна миттєво відфільтрувати наявні пропозиції. Поруч із блоком основних контактів інтегровано спеціалізовану кнопку «Wedding Makeup Inquiry», яка відкриває окреме вікно для надсилання таргетованих заявок на весільний макіяж.

Уся нижня частина інтерфейсу відведена під інтерактивну галерею послуг, яка функціонує як візуальне портфоліо спеціаліста. Кожен елемент цієї галереї є окремим блоком, що містить якісне зображення виконаної роботи, комерційну назву послуги, її точну ціну та опцію для перегляду детальних умов. Система автоматично синхронізує відображуваний контент із вибраними налаштуваннями сортування, оперативно виводячи на екран точну кількість актуальних карток, що задовольняють поточний запит користувача.

На рис. 1.4 зображено модальне вікно подачі заявки на весільний макіяж, яке відкривається поверх поточної сторінки та містить комплекс полів для деталізації замовлення.

Рис. 1.4. Сторінка бронювання візиту

Програмна форма включає календарний вибір дати урочистості, лічильник кількості подружок нареченої, а також прапорці для вибору інших членів весільної групи, які потребують послуг візажиста. Додатково в інтерфейсі передбачені текстові поля для введення локації та точного часу надання послуги, а також радіокнопки для зазначення потреби у проведенні репетиції макіяжу. Натискання акцентної кнопки ініціює валідацію введених користувачем даних та їх асинхронне відправлення на сервер через REST API для подальшої обробки адміністратором платформи.

Для забезпечення фінансової прозорості та завершального етапу взаємодії користувача з системою розроблено інтерфейс фіксації замовлень. Головну область екрана на рис. 1.5 відведено під сторінку перегляду бронювань послуг, яка реалізує функціонал користувацького кошика з детальною таблицею замовленого сервісу.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

SERVICE	PRICE	QUANTITY	SUBTOTAL
 EVENT / RED CARPET MAKE - UP BOOKING DATE: 1 MAY 2026 BOOKING TIME: 15:30 PERSONS: 1 SOLD BY: ALEX JAGGER MUA	£70.00		£70.00

Coupon code

BASKET TOTALS

SUBTOTAL: £70.00

TOTAL: £70.00

Рис. 1.5. Сторінка перегляду бронювань послуг.

У структурі цієї таблиці виведено інформативну картку послуги з фотографією-прикладом, комерційною назвою, узгодженою датою та часом візиту, кількістю осіб, а також іменем обраного візажиста. Система автоматично розраховує проміжну вартість, надає можливість скасування позиції за допомогою інтерактивного елемента видалення та містить поле для введення дисконтних промокодів із кнопкою активації купонів. У правій нижній частині інтерфейсу інтегровано підсумковий блок розрахунку, що відображає кінцеву суму до оплати та акцентну кнопку переходу до захищеного платіжного шлюзу для завершення транзакції. При аналізі даного сервісу було виявлено ряд недоліків, функціонал платформи є досить обмеженим і в основному зводиться до перегляду профілів та базового онлайн-запису. Також одним із недоліків є обмежена кількість представлених візажистів на платформі (рис. 1.2). При виборі спеціаліста користувач має досить вузький список варіантів, що зменшує можливість підібрати майстра відповідно до своїх потреб. Ще одним недоліком є відсутність функціоналу для прямої переписки між користувачем і візажистом, користувач не може уточнити деталі послуги, задати додаткові питання або повідомити про зміни. Вся взаємодія зводиться лише до форми запису, саме тому комунікація стає менш зручною і ускладнюється швидке узгодження деталей. Це може впливати на загальну зручність використання сервісу, оскільки вибір фактично є обмеженим і не завжди відповідає очікуванням користувача.

Другим обраним для системного аналізу аналогом є онлайн-платформа MakeUp Hub, яка функціонує за моделлю спеціалізованого маркетплейсу послуг краси та здоров'я. Дане програмне рішення дозволяє користувачам знаходити профільних візажистів і майстрів, детально переглядати їхні профілі та цифрові портфоліо, а також ознайомлюватися з повним переліком комерційних пропозицій. Платформа виступає інтерактивним майданчиком, де майстри мають змогу ефективно презентувати свої роботи, а кінцеві клієнти оперативно обирати спеціалістів та здійснювати запис на б'юті-послуги в онлайн-режимі.

На рис. 1.6 продемонстровано архітектуру головної сторінки MakeUp Hub, де візуальний акцент зміщено на інтерактивну пошукову систему в центрі екрана.

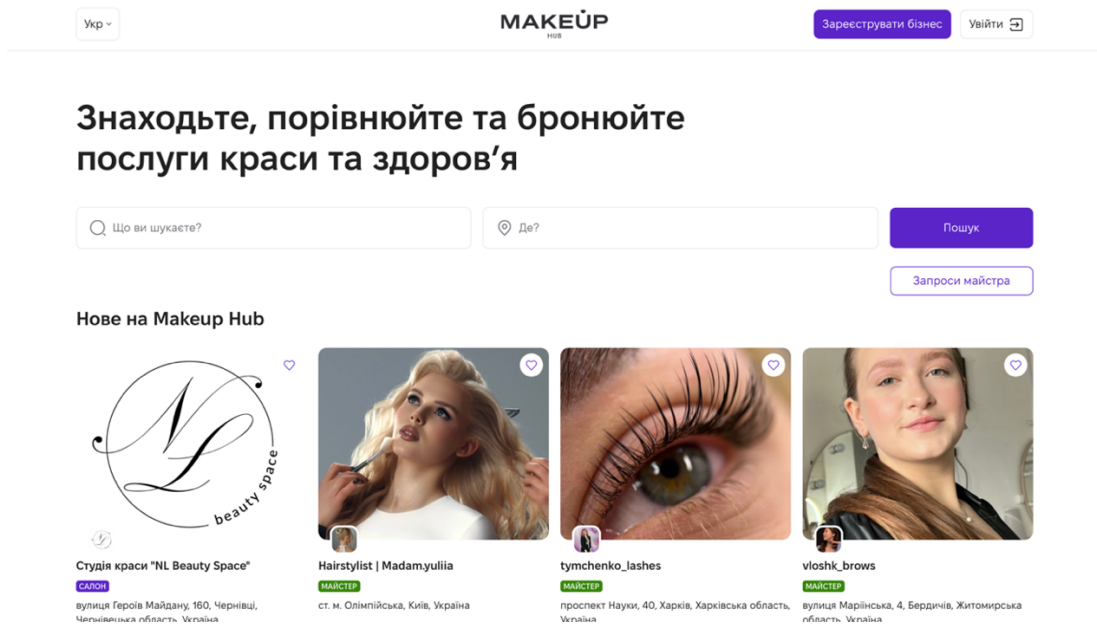


Рис. 1.6. Головна сторінка MakeUp Hub

Цей центральний модуль оснащений текстовими інпутами для введення цільового запиту і локації, а також функціональними кнопками, що ініціюють REST-запит до бази даних або дозволяють створити індивідуальну заявку на пошук спеціаліста. Над пошуковою зоною інтегровано лаконічний Header, який відповідає за вибір мовної версії сайту, авторизацію користувачів та реєстрацію нових бізнес-акаунтів. Нижню область інтерфейсу відведено під стрічку актуальних оновлень, реалізовану як динамічну сітку карток. Кожна така картка містить фотографію виконаної роботи, назву салону або ім'я майстра, їхній

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

поточний статус, точну адресу та кнопку-сердечко для збереження профілю в обране.

Для ознайомлення з детальними відомостями про обраного спеціаліста в системі реалізовано його індивідуальну вебсторінку. На рис. 1.7 продемонстровано інтерфейс персональної сторінки візажиста на платформі MakeUp Hub, де ключову інформаційну зону відведено під блок ідентифікації майстра.

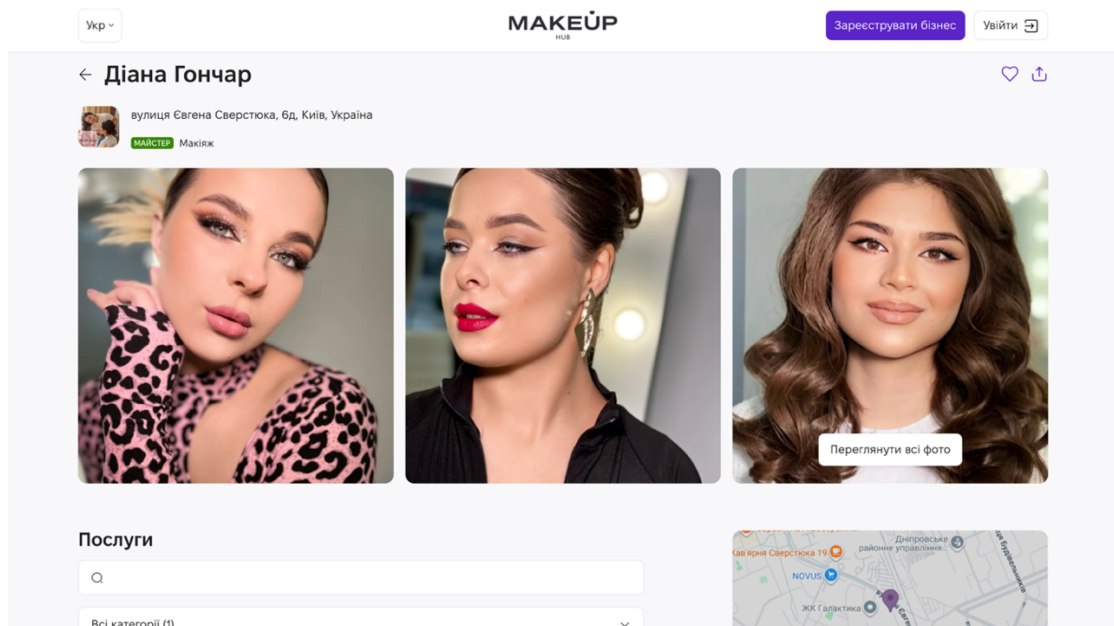


Рис. 1.7. Персональна сторінка візажиста

У верхній частині профілю відображаються ім'я спеціаліста, його професійний статус, аватар, адреса здійснення діяльності, а також елементи інтерактивної взаємодії у вигляді кнопок додавання до обраного та поширення сторінки. Основною складовою профілю є галерея портфоліо з картками виконаних робіт і можливістю переходу до повного переліку фотографій.

Нижня частина сторінки містить інструменти для взаємодії користувача з послугами, зокрема пошуковий рядок, меню вибору категорій та інтегровану карту для відображення місцезнаходження візажиста.

Дана платформа також надає користувачам доступ до головної сторінки, де представлено каталог візажистів із можливістю швидкого пошуку потрібного спеціаліста (рис. 1.6). Для кожного майстра передбачено окрему сторінку з

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

детальною інформацією про професійний досвід, перелік послуг, їх вартість, графік роботи та розташування салону на карті (рис. 1.7).

Важливою функціональною можливістю системи є онлайн-запис на послуги. За допомогою спеціальної форми користувач може обрати необхідну послугу, дату та зручний час візиту, після чого сформувавати заявку на запис (рис. 1.7).

Додатково реалізовано механізм збереження улюблених спеціалістів, що дозволяє користувачам швидко повертатися до раніше обраних майстрів. Також на платформі передбачено окремий розділ із найпопулярнішими візажистами, сформований на основі активності та вподобань користувачів.

На платформі MakeUp Hub наявна сторінка взаємодії з клієнтом фокусується на процесі безпосереднього запису до спеціаліста та ознайомлення з організаційними умовами його роботи. На рис. 1.8 продемонстровано інтерфейс форми для запису на послуги та графік роботи візажиста.

The screenshot displays the MakeUp Hub interface. At the top, there's a navigation bar with 'Укр' (Ukrainian), the 'MAKEUP hub' logo, and buttons for 'Зареєструвати бізнес' (Register business) and 'Профіль' (Profile). The main content is divided into two columns. The left column, titled 'Послуги' (Services), features a search bar, a category filter set to 'Всі категорії (1)', and a list of services: 'Вечірній макіяж' (Evening makeup) for 1000 ₴ (1h 00m), 'Денний макіяж' (Daytime makeup) for 900 ₴ (45m), and 'Весільний макіяж' (Wedding makeup) for 1500 ₴ (1h 00m). Each service has a 'Записатись' (Book) button. Below this is the 'Наша команда' (Our team) section, showing a profile for 'Діана Гончар' (Diana Gonchar), the owner. The right column shows a map of the location (Kyiv), the artist's name 'Діана Гончар', her address 'вулиця Свєгена Сверстюка, бд, Київ, Україна', a 'Як знайти нас' (How to find us) section with a phone icon and 'dianadi_makeup', and a 'Графік роботи' (Work schedule) table.

День	Час
понеділок	05:30 - 20:00
вівторок	05:30 - 20:00
середа	05:30 - 20:00
четвер	05:30 - 20:00
п'ятниця	05:30 - 20:00
субота	05:30 - 20:00
неділя	05:30 - 20:00

Рис. 1.8. Форма для запису на послуги та графік роботи

Основний простір відведено під структурований перелік доступних сервісів із зазначенням вартості, тривалості процедури та кнопками для ініціалізації резервування. Нижче розташовано блок з інформацією про

структуру та засновників студії. Права панель інтерфейсу містить інтегрований віджет карти для швидкої навігації, точну адресу, посилання на соціальні мережі, а також розгортається у деталізовану таблицю з розкладом робочих годин майстра по днях тижня.

При аналізі даного сервісу було виявлено ряд недоліків. Платформа частково обмежує користувача у виборі через не завжди достатню кількість активних візажистів у певних категоріях або регіонах. Це може ускладнювати пошук потрібного спеціаліста та звужувати варіанти для вибору. Також відсутня повноцінна система прямої комунікації між користувачем і візажистом, уточнення деталей послуги відбувається опосередковано та займає більше часу. Взаємодія зводиться до оформлення заявки на запис. Крім того, функціонал персоналізації користувацького досвіду є обмеженим, відсутні розширені рекомендації або автоматичний підбір майстрів на основі вподобань користувача. Це знижує зручність використання платформи при довготривалій роботі з сервісом.

Для дослідження світового досвіду автоматизації у б'юті-сфері було проаналізовано закордонний сервіс GlossGenius, який дозволяє ефективно керувати записами клієнтів, створювати персональні профілі та портфоліо, а також приймати онлайн-бронювання. Цей програмний продукт призначений для візажистів та інших спеціалістів у сфері краси, поєднуючи в собі потужні функції CRM-системи та інструменти для комплексної організації роботи майстра. Завдяки інтегрованій платформі клієнти мають можливість зручно переглядати профілі спеціалістів, детально ознайомлюватися з переліком послуг та самостійно записуватися на потрібний час через інтерактивний онлайн-календар.

Окремої уваги та аналізу заслуговує сторінка, представлена на рис. 1.9, концепція якої базується на поєднанні інформаційного та демонстраційного блоків для максимального залучення відвідувачів.

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

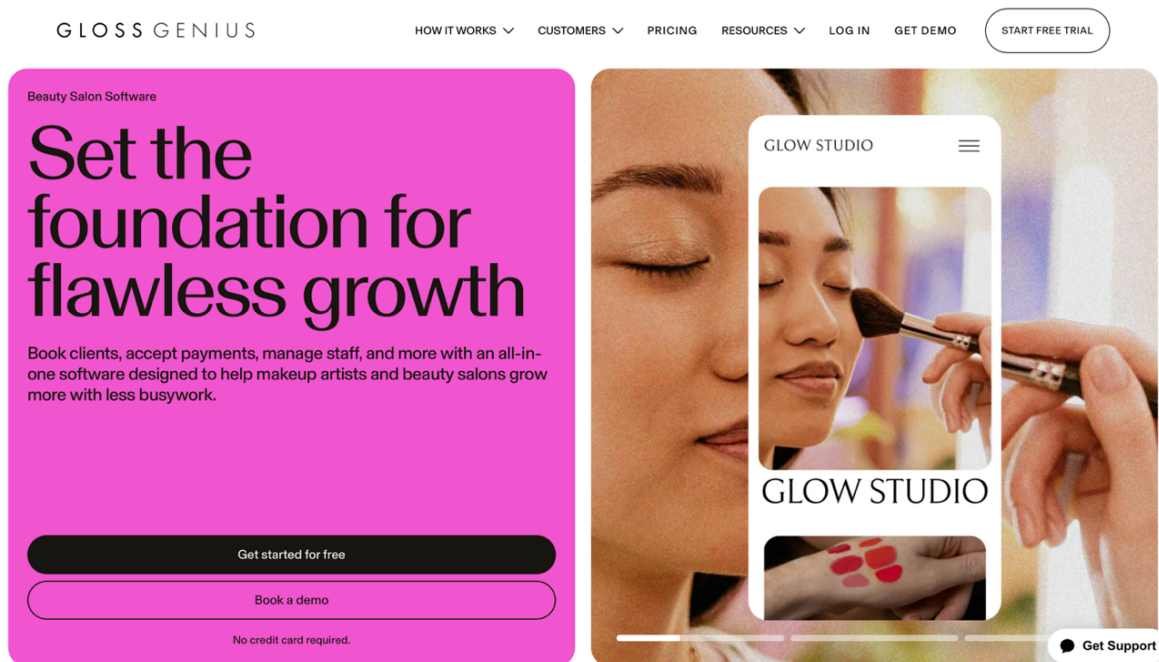


Рис. 1.9. Інтерфейс GlossGenius

Клієнтська область у верхній частині екрана виділена під панель швидких посилань, де згруповано відомості про цінову політику, можливості платформи та інструменти для авторизації користувачів. Розгорнуто акцентну зону з текстовим описом переваг продукту, що завершується функціональними кнопками для безкоштовного тестування системи або запису на демонстраційний тур. Праву сторону інтерфейсу відведено під графічний фрейм, який наочно презентує мобільну версію системи та демонструє практичне застосування додатка безпосередньо під час надання б'юті-послуг.

Окремої уваги та аналізу заслуговує сторінка, що представлена на рис. 1.10, де продемонстровано інтеграцію інтелектуальних сервісів підтримки користувачів у межах закордонного аналога GlossGenius.

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

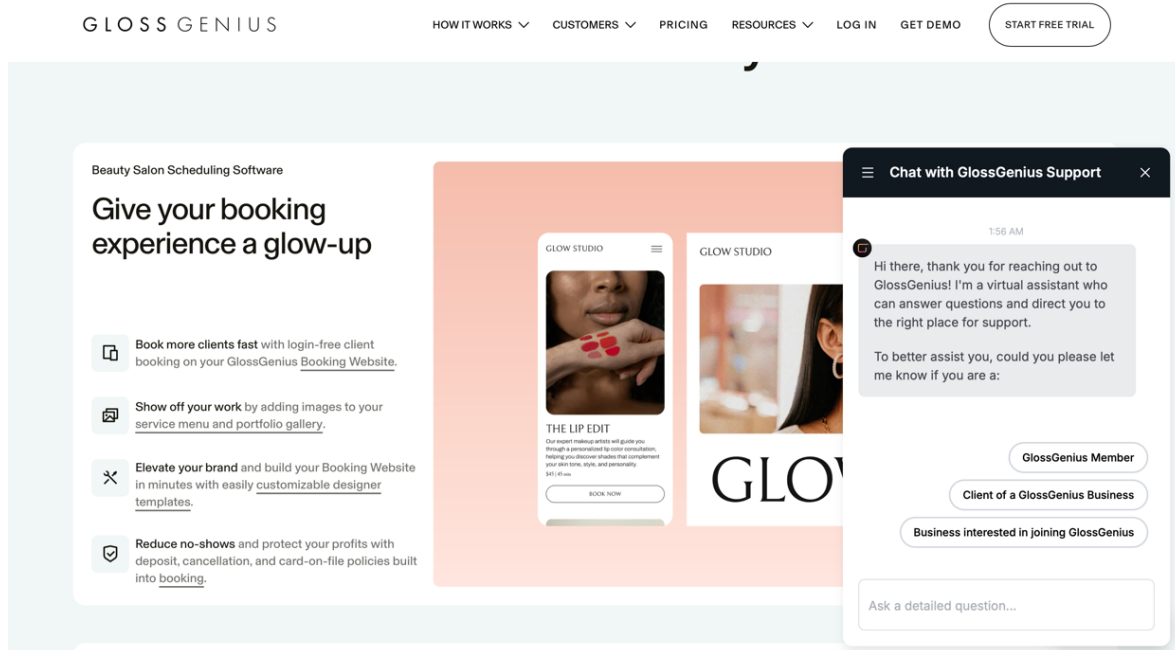


Рис. 1.10. Наявність AI-консультанта

Візуальний та функціональний акцент на даному екрані зміщено на роботу інтерактивного чат-бота, який реалізує концепцію штучного інтелекту для первинної комунікації та маршрутизації звернень. Це діалогове вікно яке містить автоматичне привітання від віртуального асистента, систему швидких інтерактивних кнопок для вибору статусу користувача, а також текстове поле для введення детальних індивідуальних запитань. Поза модальним вікном підтримки на сторінці розгорнуто інформаційний блок, який за допомогою структурованого списку з іконками описує ключові переваги платформи, включаючи швидке безпарольне бронювання, презентацію портфолію, кастомізацію шаблонів дизайну вебсайту та механізми захисту від скасування візитів.

Аналіз внутрішніх бізнес-процесів закордонного аналога дозволив виділити інтерфейсні рішення для фінансового менеджменту в межах б'юті-організацій та салонів краси. На рис. 1.11 продемонстровано архітектуру модуля сповіщень та автоматизації заробітної плати, який забезпечує високий рівень зручності для адміністрування командних виплат адміністрацією компаній лише за умови підключення преміум-підписки.

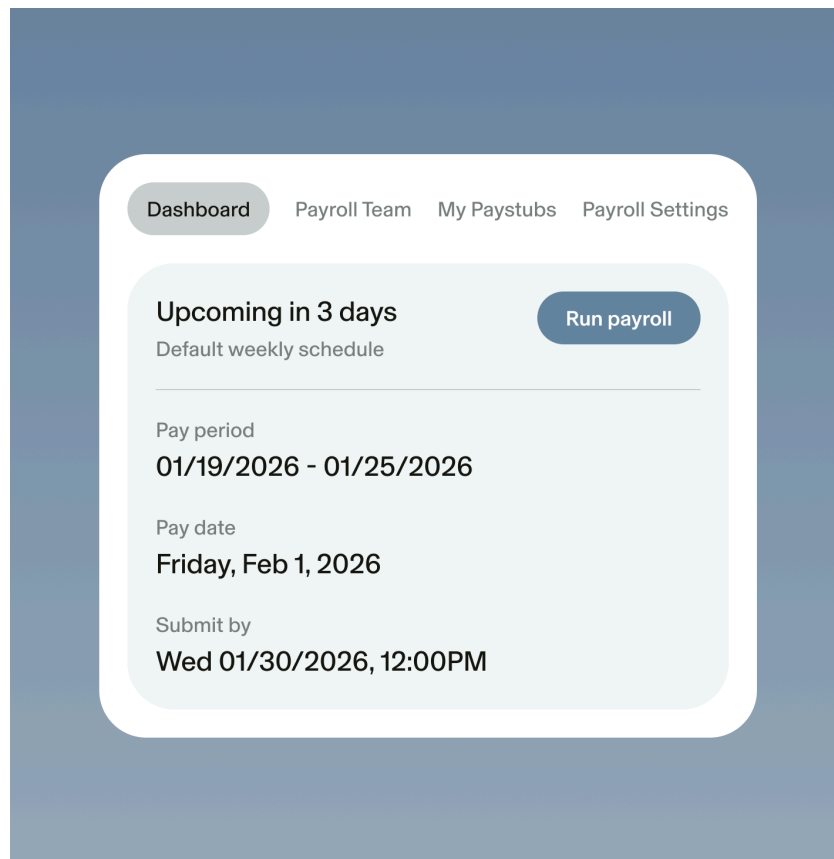


Рис. 1.11. Панель payroll із нагадуваннями

Даний функціонал реалізовано у вигляді лаконічного віджета з горизонтальною панеллю вкладок для швидкої навігації між загальною панеллю керування, налаштуваннями команди, платіжними квитанціями та конфігурацією системи. Центральний сегмент картки займає динамічне текстове нагадування про планову транзакцію та отримання заробітної плати ' із супутньою кнопкою для миттєвої активації виплат. Нижня частина віджета структурує часові обмеження операції, чітко відображаючи межі поточного платіжного період, фіксований день фактичного нарахування коштів та дедлайн для фінального подання фінансового звіту.

Логічним продовженням аналізу фінансового інструментарію системи GlossGenius є дослідження інтерфейсу безпосереднього виведення зароблених коштів. Окремої уваги та аналізу заслуговує сторінка, що представлена на рис. 1.12, де продемонстровано мобільний віджет керування поточними виплатами.

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

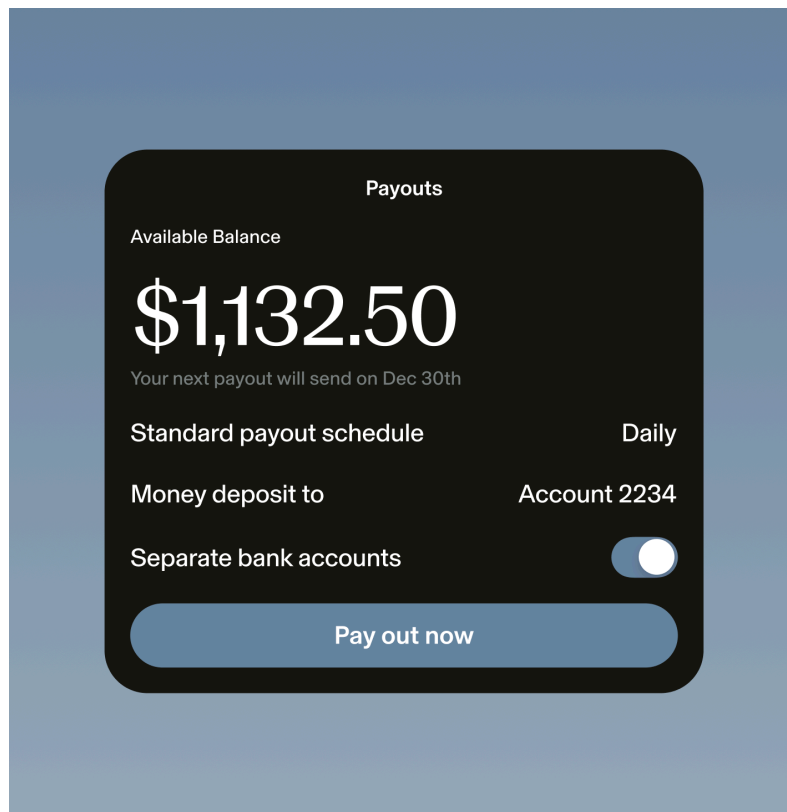


Рис. 1.12. Панель payroll із наявністю виплати

Візуальний центр даного інтерфейсу відведено під відображення доступного балансу користувача із великим інтерактивним шрифтовим виділенням суми та текстовим інформуванням про планову дату наступного автоматичного переказу. Нижче в структурі блока розташовані інформаційні поля, які фіксують стандартний графік виплат та верифікований номер розрахункового рахунку для депонування коштів. Особливістю архітектури цього модуля є наявність логічного перемикача для активації роздільних банківських акаунтів, а також інтеграція акцентної кнопки, яка дозволяє б'юті-спеціалісту здійснити миттєве позачергове виведення грошових коштів на свій рахунок у режимі реального часу.

При аналізі сервісу GlossGenius було виявлено ряд недоліків. Платформа орієнтована переважно на американський ринок, користувачі з інших країн стикаються з обмеженнями вже під час реєстрації. Для створення облікового запису необхідно мати номер телефону США, що фактично унеможливорює повноцінне використання сервісу для міжнародної аудиторії. Також суттєвим недоліком є те, що без авторизації функціонал платформи є дуже обмеженим.

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

Користувач без акаунта не має можливості повноцінно переглядати всі можливості сервісу, а також записуватися до майстрів. Майже всі основні функції, включаючи перегляд деталей послуг та бронювання, доступні лише після входу в систему.

Така модель доступу ускладнює первинне знайомство з платформою, оскільки користувач змушений спочатку проходити процес реєстрації, не маючи можливості детально оцінити зручність та функціональність сервісу. Це може негативно впливати на залучення нових користувачів, особливо з інших країн.

Для підсумкового порівняння вище наведених вебзастосунків було створено порівняльну таблицю (табл. 1.1)

Таблиця 1.1

Порівняння аналогів вебзастосунку

Критерій порівняння	Backstage MUA	MakeUp Hub	GlossGenius
Основна зручність використання	Зручний для швидкого пошуку візажистів і перегляду портфоліо, але функціонал обмежений	Зручний для перегляду каталогу майстрів і запису, є додаткові можливості (обрані, графік роботи)	Дуже зручний як CRM та система бронювання, але більше для майстрів ніж клієнтів
Простота реєстрації та старту	Проста реєстрація, але потрібен профіль	Дуже простий старт і швидкий доступ	Складніша реєстрація, є обмеження (регіон/номер)

Зручність запису/бронювання	Базова форма запису, простий функціонал	Зручна система запису з вибором часу та графіком	Дуже зручна онлайн-бронювання через календар але лише з підпискою
Загальний комфорт використання	Зручний, але функціонально обмежений	Достатньо комфортний для щоденного використання	Найбільш комфортний і сучасний, але з обмеженнями доступу
Недоліки	Обмежений функціонал, немає чату, мало майстрів	Обмежена кількість майстрів, відсутня пряма комунікація, слабка персоналізація	Обмежений доступ (регіон/реєстрація), частина функцій платна та недоступна без акаунта

Таким чином, проведений аналіз трьох платформ дозволяє зробити висновок, що кожен сервіс має свої особливості та різний рівень зручності використання. Backstage MUA є найбільш простим і базовим варіантом, який добре підходить для швидкого пошуку візажистів та перегляду портфоліо, однак його функціонал є обмеженим, а відсутність прямих комунікаційних можливостей знижує загальний рівень зручності взаємодії між користувачем і майстром.

1.3. Архітектурна система

Під час проєктування вебзастосунку студії візажистів особливу увагу було приділено створенню надійної та масштабованої архітектури, яка забезпечує ефективну обробку даних і стабільну взаємодію між усіма компонентами системи. В основу програмного комплексу покладено клієнт-серверну архітектуру, що передбачає розподіл функціональних обов'язків між клієнтською та серверною частинами. Такий підхід дозволяє відокремити інтерфейс користувача від бізнес-логіки та процесів роботи з даними, що спрощує подальший супровід і розвиток програмного забезпечення. Для зберігання інформації використовується документоорієнтована база даних MongoDB [7], яка забезпечує гнучке опрацювання даних про користувачів (Reader), візажистів (Writer), записи на консультації, публікації, коментарі та інші сутності системи.

Архітектура вебзастосунку складається з трьох основних рівнів: клієнтського, серверного та рівня зберігання даних. Клієнтський рівень відповідає за взаємодію користувачів із системою через вебінтерфейс. Серверний рівень виконує обробку запитів, реалізує бізнес-логіку та забезпечує взаємодію з базою даних і зовнішніми сервісами. Рівень даних відповідає за довготривале зберігання інформації та швидкий доступ до неї.

Серверна частина реалізована на платформі Node.js [4] із використанням фреймворку Express [5] та мови програмування TypeScript [6]. Така технологічна комбінація забезпечує високу продуктивність, типізацію коду та зручність підтримки проєкту. Архітектура бекенду побудована за модульним принципом і містить маршрути, контролери, сервіси та моделі даних. Маршрути приймають HTTP-запити та перенаправляють їх до відповідних контролерів. Контролери здійснюють первинну перевірку даних і формують відповіді у форматі JSON. Основна бізнес-логіка реалізована на рівні сервісів, що дозволяє ізолювати функціональні процеси від деталей HTTP-взаємодії.

Для роботи з MongoDB використовується бібліотека Mongoose [8], яка

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

реалізує механізм об'єктно-документного відображення. Кожна сутність системи описується окремою схемою, що визначає структуру документів, типи полів та правила валідації. Завдяки цьому забезпечується цілісність даних і спрощується виконання операцій створення, пошуку, оновлення та видалення інформації.

Клієнтська частина розроблена за допомогою бібліотеки React [3] та мови TypeScript [6]. Використання компонентного підходу дозволяє створити зручний, швидкий та масштабований інтерфейс користувача. Через вебінтерфейс відвідувачі можуть переглядати каталог візажистів (Writer), портфоліо робіт, інформацію про послуги та публікації. Зареєстровані користувачі отримують додатковий функціонал, зокрема можливість редагувати власний профіль, додавати візажистів (Writer) до списку обраних, здійснювати онлайн-запис на консультацію та користуватися AI-консультантом.

Для користувачів із роллю візажиста (Writer) реалізовано інструменти керування власним профілем, послугами та портфоліо робіт. Візажист (Writer) може додавати нові фотографії, редагувати інформацію про послуги та переглядати заявки клієнтів. Адміністратор (Admin) системи має розширені права доступу, які дозволяють здійснювати модерацію контенту, керування, обробку скарг та контроль роботи всієї платформи.

Важливим елементом клієнтської частини є використання фреймворку Tailwind CSS [21], який забезпечує швидке створення адаптивного інтерфейсу. Завдяки цьому вебзастосунок коректно відображається на персональних комп'ютерах, планшетах і мобільних пристроях без втрати функціональності. Для зберігання фотографій портфоліо та інших медіафайлів використовується хмарний сервіс Cloudinary [12]. Під час завантаження зображення передаються до хмарного сховища, де проходять автоматичну оптимізацію та зберігаються на віддалених серверах. У базі даних зберігаються лише посилання на відповідні файли. Такий підхід дозволяє зменшити навантаження на сервер та прискорити завантаження медіаконтенту для користувачів.

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

Обмін даними між клієнтською та серверною частинами реалізовано через REST API [9] із використанням протоколу HTTP [13] та формату JSON. Це забезпечує стандартизовану взаємодію між компонентами системи та спрощує можливість подальшого розширення функціоналу платформи. Для коректної взаємодії між різними доменами використовується механізм CORS [25].

Однією з ключових функціональних особливостей системи є інтеграція AI-консультанта. Його робота базується на використанні API сервісу OpenAI [11]. Консультант дозволяє користувачам отримувати інформацію про послуги студії, рекомендації щодо вибору макіяжу та відповіді на типові запитання. Для збереження історії діалогів у системі використовується окрема колекція chatSessions, що містить інформацію про повідомлення користувачів та відповіді штучного інтелекту.

Особлива увага під час розробки приділялася питанням безпеки. Для автентифікації користувачів використовується технологія JSON Web Token (JWT) [10]. Після успішного входу система генерує токени доступу, які використовуються для перевірки особи користувача під час виконання захищених запитів. Доступ до функціональних можливостей системи обмежується відповідно до ролей користувачів за допомогою спеціалізованих middleware-модулів.

Додатково в системі реалізовано механізми валідації вхідних даних, обробки помилок та контролю доступу до ресурсів. Для відправлення електронних повідомлень використовується бібліотека Nodemailer [14], що забезпечує можливість надсилання службових сповіщень користувачам. Розроблена архітектура вебзастосунка забезпечує ефективну взаємодію між усіма компонентами системи, підтримує високий рівень продуктивності та безпеки. Використання сучасних вебтехнологій, хмарних сервісів та інструментів штучного інтелекту дозволило створити програмне рішення, орієнтоване на автоматизацію процесів обслуговування клієнтів і підвищення якості роботи студії візажистів.

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

1.4. Обґрунтування вибору засобів реалізації

Для технічної реалізації вебзастосунку було обрано сучасний та збалансований стек технологій, який забезпечує високу продуктивність, зручність розробки, гнучке масштабування та відповідність актуальним стандартам безпеки вебзастосунків. Вибір програмних засобів здійснювався на основі аналізу функціональних вимог системи, зокрема необхідності побудови надійної багаторольової моделі користувачів, ефективного керування великими обсягами асинхронних запитів, динамічного рендерингу складних інтерфейсів, обробки мультимедійного контенту портфоліо та інтеграції сервісів штучного інтелекту.

Клієнтську частину вебзастосунку реалізовано як односторінковий застосунок (SPA) на базі інструменту збірки Vite та бібліотеки React [3]. Основною перевагою React є компонентний підхід до розробки інтерфейсів, який дозволяє розділити графічну оболонку системи на незалежні функціональні модулі, що суттєво спрощує підтримку коду, повторне використання інтерфейсних елементів та пришвидшує подальше розширення платформи. Завдяки використанню технології Virtual DOM бібліотека забезпечує миттєве оновлення сторінок без необхідності їх повного перезавантаження, що критично важливо для динамічних розділів, таких як стрічка, інтерактивні форми запису на послуги та вікно асинхронного AI-чату.

Для стилізації інтерфейсу платформи було обрано утилітарний CSS-фреймворк Tailwind CSS [21], що дозволило відмовитися від написання громіздких окремих файлів стилів і перенести логіку візуального оформлення безпосередньо у React-компоненти. Це рішення дозволило створити адаптивний, адаптований під світлу та темну теми інтерфейс, який коректно відображається на будь-яких пристроях (мобільних телефонах, планшетах, десктопах), суттєво скорочуючи обсяг фінального CSS-коду. Текстовий контент публікацій та відгуків, який підтримує складне форматування, обробляється та рендериться на фронтенді за допомогою спеціалізованого компонента React Markdown [23].

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

Як єдину мову програмування для клієнтської та серверної частин було обрано TypeScript [6]. На відміну від класичного JavaScript, TypeScript реалізує сувору статичну типізацію, що дозволяє виявляти логічні помилки, невідповідності структур даних та друкарські огріхи ще на етапі написання коду, до моменту компіляції застосунку. Використання типізації стало вирішальним фактором для забезпечення стабільності платформи «GlowHub», оскільки в ній одночасно взаємодіють складні моделі профілів візажистів (Writer), вкладені структури коментарів, сесії записів та кастомні API-запити, які вимагають суворого контролю типів. Для організації роботи з формами, збору даних користувачів та їхнього первинного валідування на фронтенді застосовано бібліотеку React Hook Form [22], яка мінімізує кількість зайвих рендерів компонентів.

Керування станом клієнтського застосунку розподілено на два ізольовані рівні для оптимізації архітектури. Локальний та глобальний клієнтський стан, такий як збереження теми інтерфейсу, інформація про поточну сесію користувача та його рольові права доступу, покладено на легку та продуктивну бібліотеку Zustand [17], яка є сучасною альтернативою важким архітектурам на кшталт Redux. Водночас для керування серверним станом, синхронізації даних із базою та ефективного кешування застосовано бібліотеку React Query [15]. Вона автоматично контролює актуальність даних, фоново оновлює списки постів або статус запису на прийом, усуває надлишкові запити до сервера та дозволяє впровадити оптимістичний UI для AI-консультанта, створюючи ілюзію миттєвої відповіді інтерфейсу.

Серверну частину системи побудовано на базі середовища виконання Node.js [4], що дозволило використовувати JavaScript/TypeScript стек на всіх рівнях розробки. Вибір Node.js обумовлений її подієво-орієнтованою асинхронною моделлю введення-виведення без блокування потоків, що гарантує високу пропускну здатність сервера при одночасній обробці сотень запитів від відвідувачів платформи. Для спрощення розробки серверної архітектури використано мінімалістичний фреймворк Express [5], що надає потужний набір

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

засобів для організації маршрутизації, створення проміжних шарів обробки запитів (middlewares) та побудови гнучкого REST API [9].

Спілкування між клієнтом та сервером базується на стандартах специфікації HTTP/1.1 [13] та загальних вебстандартах MDN [16] із передачею структурованих даних у форматі JSON. Для безпечного обміну даними між різними доменами фронтенду та бекенду на рівні Express інтегровано інструмент CORS [25].

Для усунення рутинного опису клієнтських запитів та досягнення повної наскрізної типізації було впроваджено зв'язку Swagger (OpenAPI Specification) [18] та кодогенератора Orval [20]. На стороні бекенду документація Swagger автоматично формує точний інтерактивний опис усіх доступних ендпоінтів, схем запитів та відповідей. Утиліта Orval аналізує згенерований Swagger JSON-файл та автоматично створює готові React Query хуки та TypeScript-інтерфейси для фронтенду. Це архітектурне рішення повністю нівелює людський фактор: будь-яка зміна структури даних у базі або на сервері миттєво відображається у вигляді помилок типізації на клієнті під час збірки проєкту.

Інформаційним фундаментом для збереження даних обрано нереляційну документо-орієнтовану систему керування базами даних MongoDB [7]. Оскільки платформа «GlowHub» суміщає текстовий контент статей, метадані зображень, гнучкі профілі майстрів та історію діалогів із III, NoSQL архітектура MongoDB, що зберігає дані у вигляді гнучких BSON-документів, виявилася набагато ефективнішою за жорсткі реляційні таблиці. Для взаємодії із сервером баз даних використано ODM-бібліотеку Mongoose [8], яка дозволяє декларувати чіткі схеми документів, автоматично валідувати типи полів перед збереженням та структурувати код сервісного шару бекенду.

Особливу увагу приділено безпеці користувацьких даних та розмежуванню доступу, що реалізовано за допомогою двотокенної схеми автентифікації на основі стандарту JSON Web Tokens (JWT) [10]. Короткоживучі Access-токени захищають кожен окремий REST-ендпоінт від несанкціонованого доступу, а довгоживучі Refresh-токени передаються в куки-файлах із прапорцями

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

httpOnly та Secure відповідно до специфікації RFC 6265 [26], що повністю захищає сесії користувачів від перехоплення за допомогою XSS або CSRF атак. Для взаємодії з апаратними та хмарними ресурсами інтегровано кілька зовнішніх сервісів. Завантаження медіафайлів (світлин робіт візажистів та аватарів) реалізовано через серверний middleware Multer [24], який передає бінарні дані до хмарного сховища Cloudinary [12]. Cloudinary виступає в ролі CDN, оптимізуючи розміри зображень «на льоту» та пришвидшуючи завантаження сторінок платформи, що також позитивно впливає на загальну пошукову оптимізацію (SEO) згідно з гайдлайнами Google Search Central [19]. Автоматизацію тригерних та транзакційних сповіщень (наприклад, повідомлення майстра про нову клієнтську заявку) покладено на бібліотеку Nodemailer [14], яка здійснює захищену доставку листів через SMTP-протокол пошти Gmail.

Інтелектуальне ядро платформи контекстний AI-консультант побудований на базі офіційного OpenAI API [11]. Інтеграція моделі штучного інтелекту дозволяє обробляти запити користувачів природною мовою, аналізувати контекст попередніх повідомлень у діалозі та видавати релевантні індивідуальні рекомендації щодо підбору б'юті-послуг чи специфіки роботи платформи, виводячи користувацький досвід UX на рівень сучасних інтелектуальних систем.

Висновки до першого розділу

У першому розділі кваліфікаційної роботи було здійснено детальний аналіз предметної області та проведено комплексне дослідження сучасних підходів до автоматизації процесів у б'юті-індустрії. Виявлено, що цифровізація діяльності студій візажу потребує створення єдиного інтегрованого простору, який поєднує функції персонального брендингу майстра, інтерактивної взаємодії з клієнтами та автоматизації адміністративних задач. На основі критичного аналізу існуючих закордонних аналогів обґрунтовано необхідність розробки вітчизняної сервісної платформи «GlowHub», яка, на відміну від комерційних рішень із закритим кодом, пропонує безкоштовний розширений інструментарій для взаємодії авторів та автентифікованих користувачів, а також інтегрує сучасні інтелектуальні сервіси обробки природно-мовної інформації.

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

На основі виявлених бізнес-вимог та специфіки багаторівневої рольової моделі користувачів було сформовано та теоретично обґрунтовано архітектурну структуру майбутнього програмного продукту. Платформу запропоновано реалізувати на базі розподіленої клієнт-серверної архітектури з чітким розділенням відповідальності, що забезпечує повну ізоляцію процесів представлення інтерфейсу, виконання обчислювальної бізнес-логіки та персистентного збереження даних. Такий підхід дозволив розмежувати взаємодію між окремими програмними доменами, гарантуючи високу відмовостійкість системи, мінімальну латентність під час обробки асинхронних запитів, а також заклав надійний фундамент для подальшого функціонального розширення та вертикального і горизонтального масштабування екосистеми вебзастосунку.

Окрему увагу в межах розділу приділено науково-технічному обґрунтуванню вибору сучасного та ефективного технологічного стеку для реалізації всіх рівнів вебзастосунку. Для шару представлення даних (Frontend) обрано компонентну бібліотеку React та суворо типізовану мову TypeScript, та інструментами автоматичної кодогенерації API-клієнтів Orval забезпечує миттєвий відгук інтерфейсу односторінкового застосунку (SPA). Центральним обчислювальним ядром (Backend) визначено платформу Node.js та фреймворк Express.js, подійно-орієнтована асинхронна архітектура яких дозволяє ефективно обробляти велику кількість одночасних користувацьких запитів без блокування потоків. Довготривале персистентне збереження інформації покладено на документоорієнтовану СКБД MongoDB та об'єктно-документне моделювання Mongoose, гнучка структура яких дозволяє оптимально оперувати динамічними даними, профілів майстрів та журналів онлайн-записів.

У результаті аналізу архітектурних рішень було обґрунтовано доцільність децентралізації збереження важкого медіа-контенту та інтеграції зовнішніх інтелектуальних мікросервісів через програмні інтерфейси. Винесення графічних портфоліо візажистів (Writer) у хмарне об'єктне сховище Cloudinary дозволило оптимізувати навантаження на основний сервер та прискорити

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

роздачу зображень кінцевим користувачам через мережу CDN. Інноваційним ядром системи визначено автономний модуль AI-консультанта на базі OpenAI API, який у режимі реального часу забезпечує первинну комунікацію, навігацію та експертну підтримку клієнтів. Побудована таким чином архітектурно-технологічна схема платформи «GlowHub» повністю задовольняє критеріям безпеки (завдяки двотокенній схемі JWT та безпечним cookies RFC 6265) і є теоретично готовою до етапу практичної програмної реалізації.

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

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Вимоги до системи

Аналітичний контур цього підрозділу спрямований на декомпозицію функціональних зв'язків та формування об'єктної моделі вебзастосунку GlowHub, що функціонує як інтегроване середовище для б'юті-індустрії з підтримкою інтелектуального асистента. Головний вектор проєктування полягає у створенні консолідованого цифрового простору, який спрощує комунікацію між аудиторією та майстрами макіяжу. Спроектowana архітектура забезпечує оперативний підбір фахівців, демонстрацію експертних кейсів у портфолію, координацію сеансів онлайн-бронювання, а також автоматизацію первинного консультування відвідувачів шляхом впровадження інструментів штучного інтелекту.

Практична цінність розробки полягає в оптимізації ключових операційних циклів, притаманних сфері надання б'юті-послуг. Програмний комплекс виконує роль централізованого репозиторію даних, де накопичуються відомості про кваліфікацію візажистів (Writer), прейскуранти та візуальні приклади робіт, надаючи користувачам інтуїтивні фільтри для пошуку виконавця. Вагомим технологічним компонентом виступає вбудований AI-модуль, який нівелює потребу в залученні живих операторів на етапі ознайомлення з сервісом, надаючи миттєві відповіді щодо специфіки б'юті-процедур та алгоритмів взаємодії з сайтом. Програмна архітектура платформи базується на компонентному розподілі і складається з кількох автономних, але синхронізованих підсистем. Візуалізацію інтерфейсу, навігацію каталогом майстрів та надсилання запитів на візити забезпечує фронтенд-модуль взаємодії з клієнтами. Робота з персональними даними експертів, оновлення контактів та завантаження фотоматеріалів координується модулем менеджменту профілів. Для ведення інформаційного та маркетингового контенту інтегровано підсистему публікацій, тоді як логіка обробки природної мови та генерація

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

відповідей покладена на ізольований модуль AI-консультанта, який функціонує через інтеграцію з хмарними штучними нейромережами [11].

Рольова структура системи передбачає диференціацію рівнів доступу для забезпечення безпеки та персоналізації інтерфейсу. Гості платформи, які не пройшли автентифікацію, обмежені ознайомчим сценарієм: їм доступний загальний огляд сторінок, пошукова видача майстрів, перегляди портфоліо та читання статей у блозі. Користувачі з підтвердженими обліковими записами переходять на рівень персональної взаємодії - вони оперують інструментарієм особистого кабінету, можуть формувати списки обраних фахівців та ініціювати сесії онлайн-запису на процедури. Найвищим пріоритетом у системі володіє адміністратор (Admin), чиї розширені повноваження дозволяють здійснювати глобальний моніторинг і модерацію всього інформаційного наповнення, аудит профілів б'юті-майстрів, редагування гілок та оперативне керування клієнтськими заявками.

Об'єктно-орієнтоване ядро системи спроектовано на основі взаємопов'язаних сутностей, які детально моделюють предметну область студії візажу. До базового пулу класів належать профілі відвідувачів, картки спеціалістів, номенклатура послуг, реєстраційні бланки записів, статті, стрічки обговорень, лог-файли діалогів з нейромережею та медіафайли галереї робіт. Кожен клас володіє набором унікальних параметрів (атрибутів) та функцій (методів), що відповідають за проходження бізнес-процесу від початкової валідації форми у документній базі даних. Клас reader виступає фундаментом для ідентифікації суб'єктів у системі, інкапсулюючи паспортні дані облікового запису, канали зв'язку та маркер поточної ролі. Сутність б'юті-майстра розширює базові параметри, акумулюючи специфічні професійні відомості: опис авторських технік, прив'язку до категорій портфоліо, географічну локацію робочого місця та інші атрибути, необхідні для позиціонування фахівця на ринку. Для фіксації та контролю угод між клієнтом і візажистом застосовується сутність заявки, яка відстежує часові маркери, етапи виконання послуги та керує зміною статусів модерації.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

Розвиток контент-маркетингу та інформаційної підтримки всередині екосистеми забезпечується сутністю публікацій, яка структурує текстові гайди, тренди б'юті-сфери та професійні рекомендації. Наостанок, для підтримки безперервного циклу інтелектуального консультування впроваджено об'єкт повідомлення AI-асистента - цей клас відповідає за персистентне зберігання діалогових сесій, накопичення контексту бесіди та стабільну роботу сервісу автоматизованих б'юті-консультацій.

Для наочного представлення спроектованих архітектурних рішень нижче наведено структурований перелік ключових вимог до системи (табл. 2.1).

Таблиця 2.1

Функціональні вимоги

Етап	Призначення
Користувач (Reader)	Зберігання облікових даних користувача, контактної інформації та ролі в системі
Візажист (Writer)	Зберігання інформації про спеціаліста, його досвід, послуги, портфоліо та контактні дані
Послуга (Service)	Містить інформацію про доступні послуги, їх опис та вартість
Портфоліо (Portfolio)	Зберігає фотографії виконаних робіт візажистів та їх опис
Запис на консультацію (bookings)	Відповідає за збереження інформації про заявки користувачів на запис до візажиста
Обрані візажисти (Favorites)	Забезпечує збереження списку вподобаних спеціалістів для користувача

Публікація (Post)	Використовується для зберігання статей, новин та інформаційних матеріалів
Коментар (Comment)	Містить відгуки та коментарі користувачів до публікацій
AI-консультант (AI Chat)	Забезпечує взаємодію користувача з системою штучного інтелекту та зберігає історію повідомлень
Адміністратор (Admin)	Керує контентом платформи, користувачами, заявками та налаштуваннями системи

Функціонування вебзастосунку студії візажистів GlowHub із вбудованим інтелектуальним асистентом базується на гнучких алгоритмах взаємодії акторів із програмними шлюзами та суворому розмежуванні рівнів доступу. Успішне проходження сесії авторизації відкриває для відвідувача повний спектр інтерактивних можливостей, серед яких вивчення персональних сторінок б'юті-майстрів, формування списку обраних експертів, надсилання онлайн-заявок та ведення діалогу з AI-консультантом. У той же час, адміністративний контур володіє монопольним правом на глобальне керування контентом, верифікацію облікових записів фахівців, модерацію статей та координацію клієнтського потоку записів на консультацію. У межах операційної логіки системи дії користувача (Reader) зосереджені навколо ключових функціональних вузлів, які охоплюють каталог візажистів, модуль електронного запису, стрічку публікацій та інтелектуального помічника. Вебзастосунок виконує роль єдиної бази даних, де систематизуються відомості про кваліфікацію майстрів, графічні матеріали портфоліо робіт, що дозволяє аудиторії здійснювати цільовий пошук за індивідуальними критеріями. Додатково впроваджено інструмент кешування та

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

збереження вподобаних профілів, який мінімізує час на повторний пошук потрібного спеціаліста.

Група нефункціональних вимог описує якісні параметри розробленого програмного забезпечення, що гарантують його стабільність, швидкодію, захищеність та зручність експлуатації (UX/UI). Незважаючи на те, що ці критерії не впливають на безпосереднє виконання бізнес-завдань, саме вони визначають відмовостійкість платформи під час пікових навантажень у реальних умовах та формують позитивний досвід взаємодії з боку користувача (Reader).

Надійність вебзастосунку виражається у безперебійному функціонуванні основних сервісів та ізоляції потенційних системних збоїв під час транзакцій до бази даних, хмарних репозиторіїв та зовнішніх API-інтерфейсів. Показники продуктивності системи гарантують високу швидкість рендерингу сторінок, оперативну обробку HTTP-запитів та збереження стабільного відгуку інтерфейсу навіть за умови високої щільності одночасних сесій відвідувачів. Важливою архітектурною перевагою системи є її високий потенціал до масштабування. Модульна структура коду дозволяє нарощувати функціональну потужність, збільшувати пропускну здатність для нових користувачів та підключати сторонні сервіси без проведення радикального рефакторингу програмного забезпечення. Це закладає надійний фундамент для подальшої модернізації платформи та інтеграції нових маркетингових інструментів. Для роботи з вебзастосунку користувачу (Reader) потрібен лише сучасний браузер із підтримкою HTML5 та JavaScript, а також стабільний інтернет. Застосунок є кросплатформним, тому однаково добре працює на комп'ютерах з операційними системами Windows, macOS чи Linux.

Серверний логічний шар платформи побудовано на базі програмної платформи Node.js із використанням мінімалістичного фреймворку Express.js [4, 5], а управління структурованими даними покладено на документоорієнтовану СКБД MongoDB [7]. Зберігання та динамічна оптимізація мультимедійного контенту делеговані хмарному медіа-менеджеру Cloudinary [12], що дозволяє

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

прискорити завантаження зображень портфоліо та суттєво розвантажити обчислювальні потужності власного сервера.

Інтеграція з обчислювальними потужностями OpenAI API [11] дозволила розгорнути функціонал інтелектуального б'юті-консультанта. Цей модуль повністю автоматизує процес первинного інформування клієнтів, допомагаючи їм оперативно отримувати відповіді на запитання щодо спектру послуг та особливостей навігації сайтом. Впровадження когнітивних технологій штучного інтелекту виводить клієнтський сервіс на новий рівень та підвищує загальну лояльність аудиторії до платформи.

Концептуальна схема взаємодії, наведена у вигляді діаграми варіантів використання, ілюструє базові сценарії оперування системою та відображає архітектурні зв'язки між рольовими моделями та функціональними модулями (Додаток А). Ця UML-модель дозволяє наочно проаналізувати траєкторію дій відвідувачів та адміністратора (Admin) під час їхньої комунікації з вебсервісом.

На схемі деталізовано ключові прецеденти (Use Cases), що охоплюють навігацію каталогом візажистів, операції з профілями спеціалістів, оформлення онлайн-записів на сеанси макіяжу, діалогові сесії з AI-асистентом, вивчення матеріалів та адміністрування контентного шару.

Взаємодія користувачів (Reader) із вебзастосунком починається з вивчення каталогу візажистів та їхніх портфоліо. Цей сценарій дозволяє відвідувачам переглядати галереї робіт, оцінювати стиль та професійний рівень майстрів. Ознайомлення з портфоліо сприяє формуванню довіри до спеціаліста та полегшує процес вибору візажиста (Writer) для майбутнього запису. Іншим важливим сценарієм є взаємодія з платформою, де публікуються корисні матеріали, статті про тренди в макіяжі, кейси та поради від експертів. Перегляд платформи дозволяє користувачам (Reader) отримувати додаткову інформацію, знайомитися з підходами візажистів (Writer) до роботи та прикладами реалізованих образів.

Процес оформлення заявки на онлайн-консультацію є одним із ключових сценаріїв взаємодії клієнта з системою. У межах цього функціоналу користувач

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

(Reader) обирає бажаного візажиста (Writer) та заповнює спеціалізовану форму, де вказує орієнтовний тип макіяжу, що його цікавить, бажану локацію для потенційної послуги та свої специфічні побажання. Цей етап не передбачає миттєвого бронювання дати, а функціонує як інструмент первинної комунікації для подальшого узгодження деталей.

Важливим інструментом підтримки користувачів (Reader) є AI-консультант, який функціонує на базі OpenAI API. Використання технологій штучного інтелекту покращує взаємодію користувачів (Reader) із системою, забезпечує швидку підтримку в режимі реального часу та підвищує загальний рівень сервісу. Адміністративна частина платформи охоплює керування профілями візажистів (Writer), що включає повний цикл створення, редагування та верифікації облікових записів майстрів. Адміністратор (Admin) має можливість моделювати інформацію про спеціалістів, перевіряти завантажені медіаматеріали портфолію, додавати описи послуг та контролювати актуальність прейскурантів. Це забезпечує структуроване та якісне представлення інформації про б'юті-експертів на платформі.

Наступним елементом є керування, яке передбачає створення, редагування та публікацію статей, кейсів і новинної інформації. Адміністратор (Admin) також здійснює управління глобальними налаштуваннями системи, контактними даними, посиланнями на соціальні мережі та SEO-описами, що впливає на видимість платформи в пошукових системах та залучення цільової аудиторії.

Ключовим операційним процесом для адміністратора (Admin) є обробка статей. У цьому модулі здійснюється аналіз вхідних запитів, перевірка коректності даних та підтвердження статті. Адміністратор (Admin) контролює весь життєвий цикл запису, переводить його між етапами та забезпечувати високу якість обслуговування клієнтів.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Розробка бази даних системи

Ефективне функціонування б'юті-платформи «GlowHub» безпосередньо залежить від організації її інформаційного контуру, де база даних виступає фундаментальним компонентом для агрегації, структурування та реплікації інформаційних потоків. Зважаючи на високу динаміку медіаконтенту та необхідність оперативного збереження неструктурованих діалогів із вбудованими інтелектуальними сервісами, як системне сховище було впроваджено документо-орієнтовану СКБД MongoDB. Ця NoSQL-технологія забезпечує динамічне розширення схем документів без зупинки працездатності сервера, гарантує низьку затримку при високій інтенсивності запитів на читання та дозволяє легко інтегрувати нові типи об'єктів.

Усі інформаційні сутності платформи розподілені за ізольованими колекціями, операційна взаємодія з якими на рівні серверного програмного коду реалізована за допомогою об'єктно-документного мапера Mongoose. На основі поточної конфігурації сховища MyNewBlog логічна структура бази даних містить вісім базових колекцій: users, bookings, chatSessions, gallery, posts, comments, complaints та subscriptions. Центральним вузлом облікових записів системи є колекція users, де зберігаються унікальні ідентифікатори, логіни, адреси електронної пошти, зашифровані хеші паролів та вектори ролей, таких як гість (Guest), користувач (Reader), візажист (Writer) або адмін (Admin). На основі значень цієї колекції здійснюється автентифікація користувачів (Reader) та розмежування прав доступу до функціоналу платформи.

За реєстрацію та обробку заявок на онлайн-консультації відповідає колекція bookings. У цих документах фіксуються зв'язки між ідентифікатором замовника та обраним візажистом (Writer), зазначаються текстові побажання щодо стилістики макіяжу, локація (студія чи виїзд), дата створення запиту та поточний статус модерації заявки адміністратором платформи (Admin).

Для забезпечення роботи інтелектуального асистента інтегровано колекцію chatSessions, яка виконує функцію довготривалої пам'яті для OpenAI

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

API. Вона логує зв'язки між клієнтом та обраним контекстом майстра, а масив повідомлень містить структуровані об'єкти: роль мовця текст репліки та часову мітку, що дозволяє алгоритмам штучного інтелекту утримувати контекст бесіди під час підбору косметики. Медіа-ядро платформи представлене колекцією gallery, яка призначена для збереження структурованих б'юті-портфолію візажистів (Writer). Замість важких бінарних файлів у документах фіксуються метадані та прямі URL-посилання на оптимізовані зображення робіт, які депонуються у хмарному сховищі Cloudinary, що забезпечує миттєве завантаження сторінок майстрів.

Інформаційне сховище у реалізоване через колекцію posts. Документи містять текстовий контент статей, новин індустрії краси та оригінальних кейсів, де обов'язковими атрибутами є ідентифікатор автора з правами адміністратора (Admin), заголовок, дата публікації та масив тегів для внутрішньої пошукової фільтрації та SEO-оптимізації платформи.

Інструментом зворотного зв'язку виступає колекція comments, що зберігає текстові відгуки та коментарі користувачів (Reader) під публікаціями у блозі. Кожен документ пов'язаний з унікальним ключем статті з колекції posts та ідентифікатором автора з колекції users, що забезпечує цілісність ланцюжків обговорення та підтримку комунікації між відвідувачами платформи.

Для забезпечення модерації та безпеки контенту впроваджено колекцію complaints. Сюди надходять та записуються скарги користувачів (Reader) на некоректні коментарі, публікації або порушення правил платформи, що дозволяє адміністратору (Admin) оперативно реагувати, перевіряти вимоги, а також блокувати або видаляти шкідливий вміст без втручання в код. Менеджмент користувацьких підписок здійснюється за допомогою колекції subscriptions. Вона фіксує дані клієнтів, які підписалися на оновлення, нові роботи конкретних візажистів або системні розсилки, забезпечуючи маркетингову взаємодію з аудиторією та автоматичну генерацію сповіщень.

Взаємозв'язки між автономними колекціями реалізовані за допомогою механізму нормалізованих посилань за унікальними ключами `_id`. Такий підхід

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

повністю виключає надлишкове дублювання інформації в документах, гарантує цілісність даних на системному рівні та оптимізує швидкість виконання запитів під час генерації динамічних сторінок вебзастосунку «GlowHub». Спроектowana схема повністю охоплює бізнес-логіку системи та готова до подальшого масштабування.

У межах документо-орієнтованої структури вона консолідує облікові записи всіх категорій суб'єктів платформи: клієнтів, професійних візажистів (Writer) та адміністраторів (Admin). Використання єдиної колекції для різних ролей дозволяє оптимізувати процедуру верифікації сесій. Для забезпечення захисту конфіденційної інформації поле пароля піддається обов'язковому асинхронному криптографічному хешуванню перед записом у сховище. Атрибут спеціалізації є динамічним масивом, який ініціалізується виключно для профілів майстрів з метою їхньої подальшої класифікації в каталозі.

Таблиця 2.2

Основні колекції бази даних

Колекція	Призначення	Основні поля
Users	Зберігання інформації про користувачів, візажистів (Writer) та адміністраторів (Admin)	_id, username, email, password, role, specializations
Posts	Зберігання статей та інформаційних матеріалів	_id, slug, title, excerpt, content, imageUrl, status, author, date
Comments	Зберігання коментарів до публікацій	_id, content, author, post, createdAt
Complaints	Зберігання скарг на публікації	_id, post, type, description, status, createdAt

Gallery	Зберігання фотографій портфоліо візажистів	_id, url, author, createdAt
Bookings	Зберігання заявок на консультації та послуги	_id, name, email, message, artist, status, createdAt
ChatSessions	Зберігання історії взаємодії користувачів з AI- консультантом	_id, user, messages, createdAt
Subscriptions	Зберігання підписок користувачів на візажистів	_id, reader, artist, createdAt

Центральним компонентом підсистеми автентифікації, авторизації та керування правами доступу є колекція users. У межах документо-орієнтованої структури вона консолідує облікові записи всіх категорій суб'єктів платформи: клієнтів, професійних візажистів (Writer) та адміністраторів (Admin). Використання єдиної колекції для різних ролей дозволяє оптимізувати процедуру верифікації сесій. Для забезпечення захисту конфіденційної інформації поле пароля піддається обов'язковому асинхронному криптографічному хешуванню перед записом у сховище. Атрибут спеціалізації є динамічним масивом, який ініціалізується виключно для профілів майстрів з метою їхньої подальшої класифікації в каталозі.

Таблиця 2.3

Опис таблиці users

Поле	Тип	Призначення
_id	ObjectId	Унікальний системний ідентифікатор документа (генерація MongoDB)
username	String	Унікальне екранне ім'я користувача (логін)
email	String	Електронна пошта користувача (унікальне значення для входу)

password	String	Хеш пароля для безпечної автентифікації
role	String	Системна роль користувача (‘client’, ‘visagist’, ‘admin’)
specializations	Array(String)	Масив категорій макіяжу (заповнюється лише для візажистів)

Інформаційне наповнення та контент-маркетинговий інструментарій платформи базуються на колекції posts. Кожен документ цієї колекції є цілісним інформаційним об’єктом, що містить як безпосередній текстовий вміст статті, так і набір метаданих, необхідних для пошукової оптимізації (SEO). Завдяки використанню унікальних людино-зрозумілих ідентифікаторів (slug) забезпечується коректне формування статичних URL-адрес сторінок. Поле статусу дозволяє реалізувати гнучкий життєвий цикл публікації – від чернетки до офіційного релізу, а зв’язок із колекцією користувачів (Reader) чітко визначає авторство матеріалу

Таблиця 2.4

Опис таблиці posts

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор публікації
slug	String	Унікальний текстовий ідентифікатор для формування ЧПУ-посилань
title	String	Заголовок статті або інформаційного матеріалу
excerpt	String	Короткий анонс статті для картки попереднього перегляду
content	String	Повний текстовий зміст статті (тіло публікації)
imageUrl	String	Посилання на головне зображення статті в Cloudinary
status	String	Поточний стан публікації (‘draft’, ‘published’)

author	ObjectId	Посилання на ідентифікатор автора публікації з колекції Users
date	Date	Офіційна дата та час публікації матеріалу на платформі

Для забезпечення інтерактивної взаємодії між відвідувачами платформи та авторами публікацій спроектовано колекцію comments. Її архітектура побудована за принципом нормалізованих посилань, де кожен документ зберігає унікальні ключі цільової статті та автора репліки. Такий підхід дозволяє уникнути надлишкового дублювання персональних даних у коментарях і забезпечує динамічне підтягування актуальних профілів користувачів (Reader). Автоматична фіксація часу створення запису дозволяє вибудовувати хронологічні ланцюжки обговорень на клієнтській стороні застосунку.

Таблиця 2.5

Опис таблиці comments

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор коментаря
content	String	Текстовий вміст коментаря користувача (Reader)
author	ObjectId	Посилання на автора коментаря з колекції Users
post	ObjectId	Посилання на цільову статтю з колекції Posts
createdAt	Date	Дата та час створення коментаря (автоматична мітка)

Модуль безпеки інформаційного простору та модерації контенту оперує даними з колекції complaints. Вона слугує для реєстрації та типізації звернень користувачів (Reader) щодо виявлених порушень, некоректної поведінки або спаму у публікаціях чи коментарях. Завдяки системі статусів обробки (status), адміністратор (Admin) платформи має можливість ефективно відстежувати життєвий цикл кожної скарги в реальному часі, фіксувати факт розгляду

інциденту та приймати рішення щодо модерації контенту без необхідності безпосереднього втручання в програмний код сервера.

Таблиця 2.6

Опис таблиці complaints

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор скарги
post	ObjectId	Посилання на статтю з колекції Posts, на яку надійшла скарга
type	String	Категорія або причина порушення (напр., спам, некоректна поведінка)
description	String	Детальний опис суті скарги від користувача (Reader)
status	String	Поточний стан обробки скарги модератором ('pending', 'confirmed', 'rejected')
createdAt	Date	Дата та час реєстрації скарги в системі

Візуальна презентація професійного рівня б'юті-майстрів реалізована за допомогою колекції gallery. Вона виконує роль цифрового каталогу виконаних робіт (портфоліо). З метою оптимізації дискового простору сервера бази даних і підвищення швидкості завантаження інтерфейсів, колекція зберігає не самі бінарні медіафайли, а їхні оптимізовані URL-адреси, згенеровані хмарним сховищем Cloudinary. Кожен елемент галереї жорстко прив'язаний до ідентифікатора візажиста (Writer), що забезпечує коректну агрегацію зображень на його персональній сторінці.

Таблиця 2.7

Опис колекції Gallery

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор елемента портфоліо

url	String	Пряме URL-посилання на завантажене медіазображення в хмарі Cloudinary
author	ObjectId	Посилання на візажиста-власника роботи з колекції Users
createdAt	Date	Дата та час завантаження фотографії до галереї

(Заявки на онлайн-консультації) Процес первинної комунікації між клієнтом та б'юті-експертом формалізовано через колекцію bookings. Вона призначена для фіксації вхідних заявок на онлайн-консультації. Структура документа дозволяє користувачеві (Reader) делегувати системі свої контактні дані, ім'я та деталізований опис побажань щодо майбутнього образу чи макіяжу. Зв'язок із конкретним майстром реалізовано через ідентифікатор автора. Поточний стан запиту контролюється системним статусом, що дозволяє візажистам (Writer) ефективно керувати чергою вхідних запитів.

Таблиця 2.8

Опис колекції Bookings

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор заявки на консультацію
name	String	Ім'я клієнта, вказане при оформленні запиту
email	String	Контактна електронна адреса клієнта для зворотного зв'язку
message	String	Текстовий опис побажань, специфіки макіяжу або запитань
artist	ObjectId	Посилання на обраного візажиста з колекції Users
status	String	Поточний системний статус запиту ('pending', 'confirmed')
createdAt	Date	Дата та час відправки заявки користувачем (Reader)

Функціонування інтелектуального модуля підтримки на базі штучного інтелекту забезпечується колекцією chatSessions. Вона виступає в ролі репозиторію довготривалої пам'яті для інтеграції з OpenAI API. Кожен документ ініціалізує унікальну сесію спілкування конкретного користувача (Reader) з ШІ-асистентом. Всередині документа зберігається динамічний масив об'єктів-повідомлень, що містить повний текст реплік та ролі мовців. Це дозволяє алгоритмам нейромережі аналізувати контекст попередньої розмови та генерувати точні рекомендації щодо стилів макіяжу.

Таблиця 2.9

Опис колекції ChatSessions

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор сесії чату
user	ObjectId	Посилання на користувача-ініціатора діалогу з колекції Users
messages	Array(Object)	Масив об'єктів реплік (роль мовця, текст повідомлення, timestamp)
createdAt	Date	Дата та час ініціалізації сесії взаємодії з ШІ-консультантом

Для підтримки лояльності аудиторії та автоматизації процесів інформування користувачів (Reader) про оновлення контенту інтегровано колекцію subscriptions. Вона фіксує стійкі інформаційні зв'язки всередині платформи, поєднуючи ідентифікатор користувача з ідентифікатором цільового візажиста (Writer). На основі записів цієї колекції бекенд-додаток ініціює тригери для генерації сповіщень про появу нових статей або нових робіт у портфоліо майстра, на якого підписаний користувач (Reader).

Опис колекції Subscriptions

Поле	Тип	Призначення
_id	ObjectId	Унікальний ідентифікатор підписки
reader	ObjectId	Посилання на користувача-підписника з колекції Users
artist	ObjectId	Посилання на цільового візажиста з колекції Users
createdAt	Date	Дата та час оформлення підписки

На основі аналізу предметної області та функціональних вимог до платформи було спроектовано логічну структуру бази даних. Оскільки для реалізації проєкту обрано документо-орієнтовану систему керування базами даних MongoDB, моделювання архітектури даних має свої особливості порівняно з класичними реляційними моделями. Логічну схему розгорнутої бази даних та взаємозв'язки між її сутностями представлено на рис. 2.1.

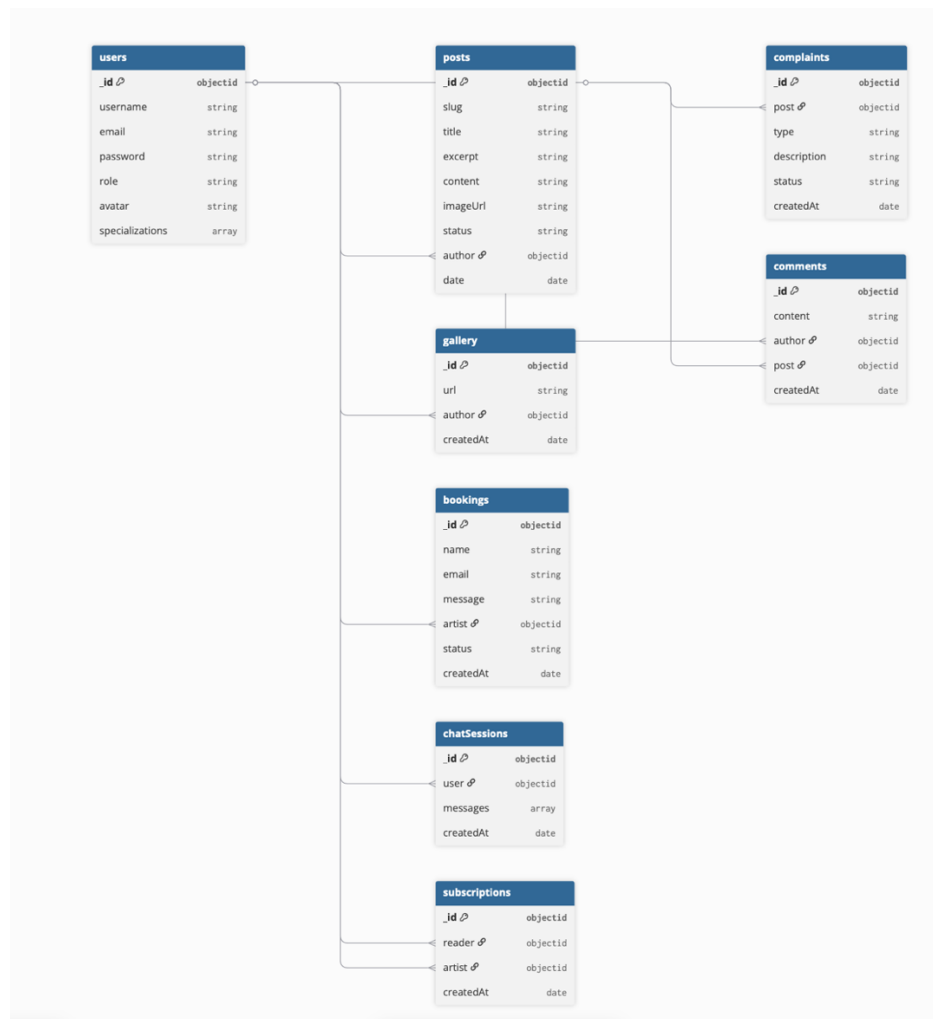


Рис. 2.1. Діаграма бази даних

Основним структурним компонентом бази даних є колекція users, яка призначена для збереження облікових записів користувачів (Reader). Вона містить поля автентифікації, рольової моделі та посилання на медіа-файли профілю (avatar). Колекція posts відповідає за сутність платформи і містить текстовий контент, метадані, унікальний slug для SEO-оптимізації посилань та зв'язок із колекцією користувачів (Reader) через поле author. Взаємодія користувачів (Reader) із контентом реалізована через колекції comments та complaints. Вони виконують функції коментування публікацій та забезпечення системи модерації скарг на контент. Обидві сутності жорстко прив'язані до ідентифікаторів постів та авторів. Медіа-контент платформи зберігається окремо у колекції gallery, яка містить прямі посилання на завантажені зображення користувачів (Reader). Функціональна частина

платформи підтримується модулем бронювання у колекції bookings, що фіксує клієнтські заявки, текстові повідомлення та ідентифікатор виконавця. Для забезпечення збереження текстових діалогів створено сутність chatSessions, де масив messages динамічно акумулює історію повідомлень користувача (Reader). Зв'язки між користувачами (Reader) типу реалізовано через колекцію subscriptions, яка відстежує підписки користувачів на оновлення конкретних авторів.

2.3. Проєктування та реалізація алгоритмів роботи системи

Ефективність функціонування розробленого вебзастосунку студії візажистів значною мірою залежить від правильності побудови алгоритмів, які забезпечують обробку даних, взаємодію між користувачами (Reader) та виконання бізнес-процесів системи. Проєктування алгоритмічного забезпечення дозволяє формалізувати послідовність дій під час виконання основних операцій вебзастосунку та визначити механізми взаємодії між клієнтською частиною, сервером і базою даних. У межах розробленої системи особливу увагу приділено алгоритмам реєстрації та авторизації користувачів (Reader), створення та публікації контенту, обробки заявок на консультації, а також роботі AI-консультанта.

Одним із ключових алгоритмів системи є алгоритм реєстрації користувачів (Reader). Його основне призначення полягає у створенні нового облікового запису та забезпеченні безпечного збереження персональних даних. Робота алгоритму розпочинається з введення користувачем (Reader) реєстраційних даних через вебінтерфейс. Після надсилання форми сервер отримує інформацію та виконує перевірку коректності введених значень. На цьому етапі перевіряється правильність формату електронної пошти, довжина пароля та заповнення обов'язкових полів. У разі виявлення помилок користувачу (Reader) повертається відповідне повідомлення.

Якщо дані відповідають встановленим вимогам, система виконує перевірку унікальності електронної адреси в колекції користувачів (Reader) бази даних

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

MongoDB. У випадку, якщо користувач (Reader) із такою адресою вже існує, реєстрація блокується. Після успішного проходження перевірок пароль шифрується за допомогою алгоритму bcrypt, що дозволяє уникнути зберігання паролів у відкритому вигляді. Далі створюється новий документ користувача (Reader) та записується до колекції users. Після завершення операції система повідомляє користувача (Reader) про успішне створення облікового запису.

Не менш важливим є алгоритм авторизації, який забезпечує автентифікацію користувачів (Reader) та контроль доступу до функціональних можливостей платформи. Під час входу користувач (Reader) вводить електронну пошту та пароль, після чого сервер виконує пошук відповідного облікового запису в базі даних. Якщо користувач (Reader) знайдений, здійснюється порівняння введеного пароля з його зашифрованою версією, що зберігається в системі. У разі успішної перевірки генерується JWT-токен доступу, який містить ідентифікатор користувача (Reader) та його роль.

Згенерований токен передається клієнтській частині та використовується для підтвердження особи користувача (Reader) під час виконання подальших запитів до сервера. Завдяки використанню JWT реалізовано безсесійний механізм автентифікації, який дозволяє зменшити навантаження на сервер та підвищити масштабованість системи. Крім того, роль користувача (Reader), що міститься в токені, використовується для реалізації механізму розмежування доступу до окремих функцій вебзастосунку.

Наступним важливим компонентом є алгоритм створення та публікації контенту. Він використовується для додавання нових статей та розміщення інформаційних матеріалів на платформі. Перед початком виконання операції система перевіряє наявність дійсного JWT-токена та визначає права поточного користувача (Reader). Лише користувачі з відповідними ролями можуть створювати або редагувати публікації. Після проходження перевірки виконується обробка введених даних. Текстові поля проходять валідацію та очищення від потенційно небезпечного вмісту. Якщо до публікації додається зображення, воно передається до хмарного сервісу Cloudinary. Після успішного

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

завантаження система отримує URL-адресу файлу та формує новий документ статті, який містить заголовок, короткий опис, основний текст, зображення та інформацію про автора. Після збереження документа в колекції posts нова стаття автоматично стає доступною для перегляду користувачами (Reader).

Окреме місце в архітектурі системи займає алгоритм обробки коментарів. Його призначення полягає в організації взаємодії між відвідувачами платформи та авторами публікацій. Після введення тексту коментаря система перевіряє, чи авторизований користувач (Reader). Далі здійснюється перевірка коректності введених даних та створюється новий документ у колекції comments. Коментар автоматично прив'язується до відповідної статті та користувача (Reader), який його залишив. Завдяки цьому забезпечується можливість обговорення матеріалів та підвищення активності користувачів (Reader) на платформі.

Важливим бізнес-процесом системи є алгоритм створення заявки на консультацію. Він забезпечує взаємодію між клієнтом та обраним візажистом (Writer). Після заповнення форми користувачем (Reader) сервер отримує дані заявки, перевіряє їхню коректність та створює новий запис у колекції bookings. На початковому етапі заявці автоматично присвоюється статус «pending», який означає очікування розгляду.

Після створення заявки інформація стає доступною відповідному візажисту (Writer). Виконавець може переглянути отриманий запит та прийняти рішення щодо його підтвердження або відхилення. У разі підтвердження статус змінюється на «confirmed», а у випадку відмови - на «rejected». Усі зміни автоматично зберігаються в базі даних та відображаються користувачеві (Reader) в інтерфейсі системи. Це дозволяє забезпечити прозорий процес взаємодії між сторонами та зменшити кількість додаткових комунікацій. Однією з основних особливостей розробленої платформи є інтеграція AI-консультанта. Для його роботи реалізовано окремий алгоритм обробки повідомлень користувачів (Reader). Після надсилання повідомлення текст запиту передається на сервер, де формується звернення до API OpenAI. До запиту додається контекст поточного

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

діалогу та службові інструкції, які визначають поведінку віртуального консультанта.

Отримана від AI-моделі відповідь повертається клієнту та відображається у вікні чату. Для підтримки контексту історія повідомлень зберігається у колекції `chatSessions` та використовується під час формування наступних запитів. Це дозволяє AI-консультанту надавати більш точні та персоналізовані рекомендації щодо вибору послуг, типів макіяжу та особливостей роботи студії. Розроблені алгоритми забезпечують коректне функціонування основних модулів вебзастосунку студії візажистів. Їх використання дозволяє автоматизувати процеси реєстрації, авторизації, публікації контенту, обробки заявок та роботи AI-консультанта. Комплексне застосування зазначених алгоритмів підвищує продуктивність системи, забезпечує цілісність даних і створює умови для подальшого розвитку програмного продукту.

2.4. Реалізація функціоналу платформи

Розроблений вебзастосунок студії візажистів побудований за сучасною архітектурою Frontend/Backend, яка передбачає відокремлення клієнтської та серверної частин системи. Такий підхід забезпечує незалежний розвиток окремих компонентів програмного забезпечення, підвищує масштабованість системи та спрощує її супровід. Клієнтська частина відповідає за взаємодію з користувачем (Reader) і відображення інформації, тоді як серверна частина реалізує бізнес-логіку, обробляє запити та забезпечує доступ до даних.

Серверна частина вебзастосунку реалізована на платформі Node.js із використанням фреймворку Express.js. Вибір даної технології обумовлений її високою продуктивністю, підтримкою асинхронної обробки запитів та широкими можливостями інтеграції із зовнішніми сервісами. Сервер виконує функції обробки HTTP-запитів, маршрутизації, взаємодії з базою даних MongoDB та реалізації механізмів автентифікації користувачів (Reader).

Архітектура серверної частини побудована за модульним принципом. Кожен функціональний модуль відповідає за окрему групу операцій та має

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

власний набір маршрутів, контролерів і моделей даних. Зокрема, у системі реалізовано модулі користувачів (Reader), публікацій, галереї, коментарів, підписок, бронювань, скарг та AI-консультанта. Така структура дозволяє спростити підтримку програмного коду та забезпечує можливість подальшого розширення функціоналу без внесення значних змін до існуючих компонентів.

Для зберігання інформації використовується документоорієнтована база даних MongoDB. У базі даних створено окремі колекції для користувачів (Reader), публікацій, галереї, коментарів, заявок на консультації, підписок, чатів AI-консультанта та скарг.

Для взаємодії із базою даних використовується бібліотека Mongoose, яка реалізує механізм об'єктно-документного відображення. Кожна сутність системи представлена окремою схемою з визначеним набором атрибутів та правил валідації. Завдяки цьому забезпечується цілісність даних і контроль правильності інформації, яка надходить до бази даних.

Клієнтська частина вебзастосунку створена за допомогою бібліотеки React. Використання React дозволяє реалізувати сучасний односторінковий інтерфейс із швидким оновленням даних без повного перезавантаження сторінки. Усі елементи інтерфейсу реалізовані у вигляді окремих компонентів, що підвищує повторне використання коду та спрощує підтримку проєкту.

Через клієнтський інтерфейс користувачі (Reader) можуть переглядати каталог візажистів, ознайомлюватися з портфоліо робіт, переглядати статті, здійснювати запис на консультації та отримувати рекомендації від AI-консультанта. Для зареєстрованих користувачів (Reader) додатково реалізовано особистий кабінет, де можна редагувати профіль та керувати обраними візажистами (Writer).

Одним із ключових модулів системи є модуль автентифікації та авторизації користувачів (Reader). Для реєстрації нового користувача (Reader) використовується маршрут /auth/register. Після надсилання реєстраційної форми сервер виконує перевірку коректності даних, перевіряє унікальність електронної адреси та зберігає нового користувача (Reader) у базі даних. Паролі користувачів

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

перед записом до бази даних проходять процедуру хешування, що дозволяє підвищити рівень безпеки системи.

Вхід до системи реалізовано через маршрут `/auth/login`. Після успішної перевірки облікових даних сервер генерує JWT-токен, який використовується для підтвердження особи користувача (Reader) під час подальших запитів. На клієнтській стороні інформація про авторизацію зберігається глобально та використовується для відображення відповідного функціоналу залежно від ролі користувача (Reader).

У системі передбачено декілька ролей користувачів: гість (Guest), зареєстрований користувач (Reader), візажист (Writer) та адміністратор (Admin). Кожна роль має власний набір дозволів. Наприклад, адміністратор (Admin) отримує доступ до інструментів модерації контенту та обробки скарг, тоді як візажист (Writer) може керувати власним профілем, портфоліо та послугами.

Для зберігання медіаконтенту використовується хмарний сервіс Cloudinary. Під час завантаження фотографій клієнтський інтерфейс передає файл на сервер, де він обробляється за допомогою бібліотеки Multer. Після цього файл автоматично завантажується до Cloudinary, а система отримує пряме посилання на нього.

Отримана URL-адреса записується до відповідного документа MongoDB. Такий підхід дозволяє суттєво зменшити навантаження на сервер та базу даних, оскільки безпосередньо медіафайли не зберігаються всередині системи. Крім того, використання Cloudinary забезпечує швидку доставку контенту користувачам (Reader) завдяки розподіленій мережі серверів.

Функціонал платформи реалізований через окремий модуль публікацій. Авторизовані візажисти (Writer) можуть створювати нові статті, редагувати вже наявні матеріали та додавати до них ілюстрації. Кожна публікація містить заголовок, короткий опис, основний текст, зображення та інформацію про автора. Для зручності пошуку автоматично формується унікальний slug, який використовується в URL-адресах сторінок. Для забезпечення взаємодії між користувачами (Reader) реалізовано систему коментарів. Коментарі

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

зберігаються в окремій колекції та пов'язані з відповідними публікаціями через ідентифікатори документів. Під час завантаження сторінки статті сервер автоматично отримує інформацію про авторів коментарів із колекції користувачів (Reader).

Для реалізації такого механізму використовується функція Populate бібліотеки Mongoose. Вона дозволяє автоматично підвантажувати пов'язані документи без необхідності виконання додаткових запитів до бази даних. Аналогічний підхід використовується для реалізації системи підписок та заявок на консультації.

Важливим компонентом вебзастосунку є модуль онлайн-запису до візажистів (Writer). Після заповнення користувачем (Reader) форми система створює новий документ у колекції bookings. Заявка містить інформацію про клієнта, вибраного візажиста (Writer), текст повідомлення та поточний статус. Візажист (Writer) має можливість переглядати отримані заявки та змінювати їхній статус залежно від результату розгляду.

Для підвищення якості обслуговування клієнтів у вебзастосунок інтегровано AI-консультанта. Він дозволяє користувачам (Reader) отримувати відповіді на поширені запитання щодо послуг студії, рекомендації з вибору макіяжу та іншу довідкову інформацію. Історія діалогів зберігається в окремій колекції chatSessions, що дозволяє підтримувати контекст спілкування та формувати більш точні відповіді. Усі модулі системи працюють в асинхронному режимі із застосуванням можливостей Node.js та конструкцій async/await. Завдяки цьому сервер здатний одночасно обробляти значну кількість запитів без блокування основного потоку виконання. Поєднання React, Express.js, MongoDB та Cloudinary забезпечує високу продуктивність, гнучкість і масштабованість розробленого вебзастосунку, а також створює основу для його подальшого розвитку та розширення функціональних можливостей.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до другого розділу

У другому розділі роботи було здійснено повний цикл проектування, алгоритмізації та програмної реалізації платформи «GlowHub». На основі аналізу предметної області сформовано оптимальну інфологічну модель даних та розроблено архітектуру вебдодатка. Спроектовано та реалізовано гнучку NoSQL-структуру бази даних у середовищі MongoDB, яка складається з 8 взаємопов'язаних колекцій: users, posts, comments, complaints, gallery, bookings, chatSessions та subscriptions. Розроблено логічні алгоритми для ключових функціональних модулів системи, зокрема: сервісу автентифікації та авторизації на базі токенів JWT, модуля публікації контенту з інтеграцією хмарного сховища Cloudinary, а також асинхронного алгоритму обробки заявок на бронювання. Усі алгоритми побудовані з урахуванням асинхронної специфіки платформи Node.js, що мінімізує затримки при обробці запитів користувачів (Reader).

Проведено програмну реалізацію платформи за сучасною архітектурою розподілення клієнтської та серверної частин (Frontend/Backend). Серверний функціонал успішно розгорнуто на базі фреймворку Express.js, що гарантує високу швидкість маршрутизації, а реактивний клієнтський інтерфейс створено за допомогою бібліотеки React.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ

3.1. Порядок встановлення та налаштування параметрів системи

Оскільки початковий етап розробки та фіксації змін відбувався на платформі GitHub, першим архітектурним кроком є повне перенесення та синхронізація кодової бази з цільовим репозиторієм на GitLab. Для цього локальний Git-клієнт налаштовується на роботу з двома віддаленими серверами (remotes). За допомогою командного рядка до поточного локального репозиторію додається нова адреса призначення, після чого виконується передавання усієї історії комітів та наявних гілок. Це забезпечує повну ідентичність коду на новій платформі GitLab, яка надалі виступатиме головним сховищем проєкту для контролю версій та розгортання.

Архітектура застосунку вимагає ізольованого керування залежностями для серверної та клієнтської частин. У системі обов'язково має бути встановлене середовище виконання Node.js. Розгортання починається з переходу в кореневу директорію бекенду та запуску команди `npm install`, яка зчитує файл `package.json` і завантажує ядро Express, TypeScript-інструменти (ts-node), ORM-бібліотеку Mongoose та безпекові пакети. Наступним кроком є перехід у директорію фронтенду, де аналогічна команда розгортає середовище збірки Vite, React-компоненти, менеджер стану Zustand та інструмент автоматичної генерації API-клієнта Orval. Це створює локальний фундамент, де кожна частина має власну ізольовану теку `node_modules`.

Серверна частина вебзастосунку GlowHub побудована на основі Node.js та Express і використовує зовнішні сервіси для зберігання даних та мультимедійного контенту. Налаштування підключення до цих сервісів здійснюється за допомогою конфігураційного файлу `.env`, який розташований у кореневій директорії серверної частини проєкту. Для роботи з документоорієнтованою базою даних MongoDB використовується змінна середовища `MONGODB_URI`, що містить рядок підключення до локального екземпляра бази даних або хмарного кластера MongoDB Atlas. На основі цього

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

підключення бібліотека Mongoose забезпечує взаємодію з колекціями користувачів (Reader), публікацій, коментарів та інших сутностей системи.

Для зберігання фотографій портфоліо візажистів (Writer), аватарів користувачів (Reader) та іншого мультимедійного контенту використовується хмарний сервіс Cloudinary. Параметри доступу до сховища задаються через змінні середовища CLOUDINARY_CLOUD_NAME, CLOUDINARY_API_KEY та CLOUDINARY_API_SECRET. Завантаження файлів на сервер здійснюється за допомогою проміжного програмного забезпечення Multer, після чого зображення автоматично передаються до Cloudinary для подальшого зберігання та обробки. Такий підхід дозволяє зменшити навантаження на сервер, підвищити швидкість доступу до медіафайлів та забезпечити надійне зберігання контенту.

Для захисту персональних даних та авторизації запитів в архітектуру системи закладено механізм двокомпонентних JWT-токенів. У файлі оточення .env необхідно прописати змінні ACCESS_SECRET та REFRESH_SECRET із випадковими складними рядками для шифрування підписів токенів. Також виставляється час їхнього життя: ACCESS_EXPIRATION=15m (короткочасний токен доступу, що зберігається в пам'яті клієнта через Zustand) та REFRESH_EXPIRATION=7d (довготривалий токен для оновлення сесії, що передається в захищених Cookies за допомогою cookie-parser). Оскільки фронтенд і бекенд працюють на різних портах, для запобігання блокування запитів браузером налаштовується політика CORS: у змінну FRONTEND_URL чітко прописується адреса клієнта (за замовчуванням http://localhost:5173), відкриваючи легітимний шлях для обміну даними між доменами.

Важливою функціональною частиною архітектури GlowHub є асинхронне інформування користувачів (Reader) та майстрів про бізнес-події. Модуль розсилки базується на протоколі SMTP і потребує налаштування поштових шлюзів у .env. Конфігурація включає: SMTP_HOST=smtp.gmail.com, SMTP_PORT=465 та активацію шифрування SMTP_SECURE=true. У змінну SMTP_USER вноситься реальна електронна адреса, а у SMTP_PASS спеціальний 16-значний «пароль додатка», згенерований у налаштуваннях

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

безпеки Google-акаунта (двофакторна автентифікація). Рядок SMTP_FROM визначає канонічне ім'я відправника, яке бачитиме клієнт у своїй поштовій скриньці.

Вбудований у платформу AI-консультант реалізовано шляхом інтеграції з неймережевими моделями OpenAI через офіційний API. Для активації функціональності штучного інтелекту в чаті системи у файлі середовища визначається змінна OPENAI_API_KEY, у якій зберігається секретний токен, отриманий у кабінеті розробника OpenAI. Генерація відповідей здійснюється через API-запити, що обробляються у хмарній інфраструктурі OpenAI та повертають результат у вигляді текстових відповідей на основі переданого контексту.

Фінальний етап розгортання передбачає одночасний запуск двох частин системи в режимі розробки та первинне наповнення бази даних тестовим контентом. У терміналі бекенду виконується команда `npm run dev`, яка за допомогою `ts-node` запускає HTTP-сервер на базі Express, відкриває API-ендпоїнти та забезпечує підключення до MongoDB і Cloudinary. Паралельно у окремому терміналі запускається фронтенд за допомогою команди `npm run dev`, що піднімає локальний сервер Vite для клієнтської частини застосунку.

Після запуску системи виконується ручна ініціалізація середовища: створюються тестові облікові записи з різними ролями користувачів (Admin, Writer, Reader). Це необхідно для перевірки коректності реалізації рольової моделі доступу, функціоналу завантаження медіафайлів у галерею, роботи системи публікацій, а також базових сценаріїв взаємодії користувачів із платформою.

Далі проводиться комплексне тестування основних функціональних модулів системи, зокрема реєстрації та авторизації користувачів, роботи AI-консультанта, механізму онлайн-запису, завантаження медіаконтенту та системи керування даними. Успішне проходження тестування підтверджує коректність конфігурації програмного середовища та готовність вебзастосунку до подальшого розгортання у виробничому середовищі.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Структура інтерфейсу

Інтерфейс вебзастосунку студії візажистів з AI-консультантом розроблено з урахуванням принципів зручності використання, логічної навігації та адаптивності для різних типів пристроїв. Структура інтерфейсу забезпечує швидкий доступ до основних функцій системи та сприяє комфортній взаємодії користувачів (Reader) із платформою. Як показано на прикладі головної сторінки (рис. 3.1), першим візуальним елементом, який зустрічає користувача (Reader), є мінімалістичний та естетичний бренд-блок GlowHub.

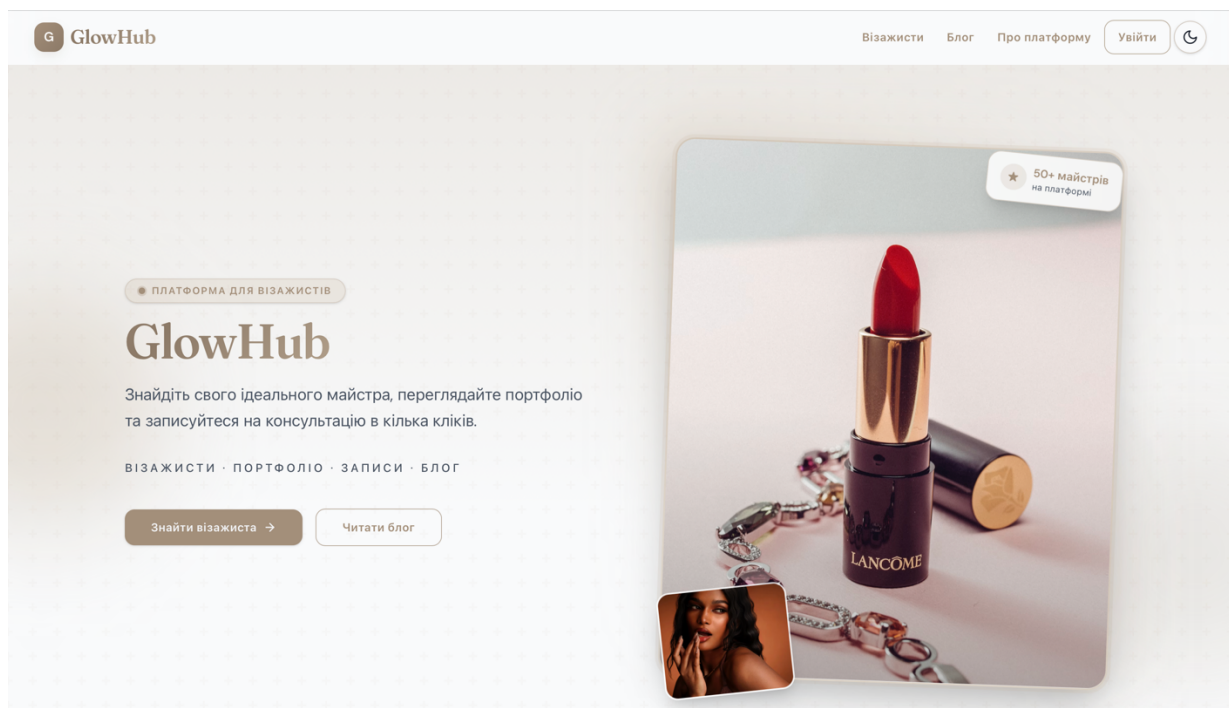


Рис. 3.1. Головна сторінка

Ліва частина екрана відведена під текстове позиціонування платформи, де чітко окреслено її головне призначення пошук ідеального майстра, перегляд портфоліо та швидкий запис на консультацію. Для спрощення взаємодії тут розміщено основні елементи заклику до дії у вигляді двох контрастних кнопок, яка веде до каталогу майстрів, та переходу до експертних статей. Права частина інтерфейсу збалансована великим трендовим зображенням б'юті-тематики із декоративними паралакс-ефектами, що створює атмосферу професійної студії.

Якщо прокрутити головну сторінку трохи нижче, то перед користувачем (Reader) відкривається наступний важливий контентний блок, який детально

					ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

продемонстровано на прикладі інтерфейсу сторінки цінностей та історії (рис. 3.2). Цей розділ візуально розділений на дві великі частини, які допомагають користувачеві (Reader) ближче познайомитися з платформою, її місією та етапами розвитку. Ліва частина екрана присвячена історичному таймлайну. Вона виконана у вигляді вертикальної хронологічної лінії з круглими маркерами, де просто і зрозуміло розписано шлях трансформації проєкту від звичайного контентного сайту до великого сервісу.



Рис. 3.2. Сторінка цінностей та історії

Права частина інтерфейсу, яку ми бачимо на рис. 3.2, повністю відведена під інформаційний блок, що надає довідкову інформацію про б'юті-платформу та сприяє формуванню довіри користувачів. Ці цінності оформлені у вигляді вертикального списку з трьох окремих заокруглених карток, кожна з яких має лаконічну іконку з лівого боку та чітке текстове пояснення. Під час дефіле у цьому блоці розміщено фінальну акцентну плашку з цитатою команди GlowHub, яка підкреслює головну місію сервісу віру в те, що кожен заслуговує знайти свого майстра, а кожен майстер своїх клієнтів.

При переході користувача (Reader) до розділу статей перед ним розгортається повноцінний каталог публікацій, детально представлений на прикладі сторінки, що зображений на рис. 3.3. Верхня частина цього інтерфейсу

виділена під інтерактивну панель фільтрації та пошуку контенту, яка оформлена у вигляді великого закругленого блоку. Тут розміщено головне поле із підказкою, що працює з вбудованим механізмом debounce для уникнення надлишкових запитів до сервера під час введення тексту. Трохи нижче інтегровано додаткові інструменти сортування та відбору: текстове поле для швидкого пошуку за іменем конкретного візажиста (Writer), розкривний список «Сортування», де за замовчуванням виставлено параметр «Нові спочатку», а також функціональна кнопка «Скинути» для миттєвого очищення всіх застосованих користувачем (Reader) конфігурацій.

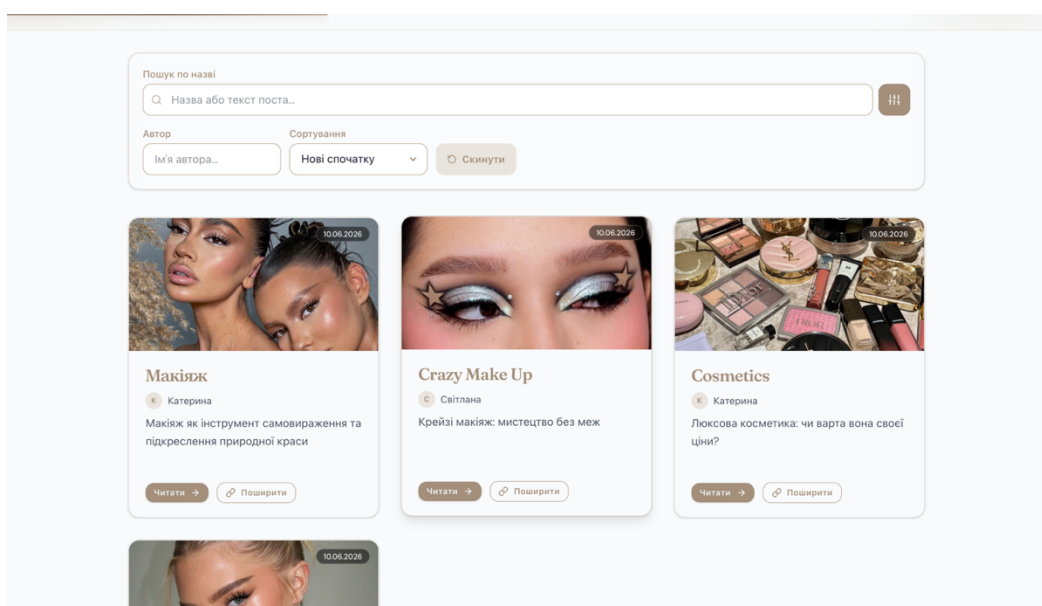


Рис. 3.3. Сторінка статей

Основну площу екрана на рис. 3.3 займає адаптивна контентна сітка, що складається з карток публікацій, які візуально відображають творчий доробок майстрів. На зображенні чітко видно картки верхнього ряду та початок наступної статті. Кожна картка спроектована за єдиним ергономічним стандартом: зверху розміщено якісну тематичну обкладинку з б'юті-індустрії, у правому кутку якої закріплено темний індикатор із датою публікації. Під фотографією виводиться великий заголовок статті, рядок з круглою іконкою автора та його іменем, а також короткий текстовий опис-уринок, який розкриває головну суть матеріалу.

У самому низу кожної такої картки симетрично розташовані два головні інтерактивні елементи керування: акцентна кнопка (читати) зі стрілкою для

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

переходу на сторінку повного тексту статті та кнопка (поширити) з іконкою ланцюжка, яка дозволяє миттєво скопіювати пряме посилання на публікацію в буфер обміну для подальшого надсилання.

Для нових користувачів (Reader), які прагнуть отримати доступ до розширеного рольового функціоналу системи, передбачено окрему сторінку реєстрації, детальну структуру якої представлено на рис. 3.4. Інтерфейс цієї форми виконано у вигляді чистої, центрально орієнтованої картки з легким б'юті-дизайном, що відповідає загальній стилістиці платформи.

The image shows a mobile app registration screen. At the top, there's a back arrow and the text 'На головну'. Below that is a user icon and the title 'Створіть акаунт' with the subtitle 'Зареєструйтесь, щоб почати'. The form contains four input fields: 'Ім'я' (with placeholder 'Ваше ім'я'), 'Пошта' (with placeholder 'you@example.com'), 'Пароль' (with placeholder 'Мінімум 6 символів' and a visibility icon), and 'Підтвердження пароля' (with placeholder 'Повторіть пароль' and a visibility icon). Below these is a toggle switch for 'Зареєструватися як автор' with the text 'Автори можуть публікувати пости в блозі'. At the bottom of the form is a large 'Зареєструватися' button. Below the button, there are links: 'Вже маєте акаунт? Увійти' and 'Продовжити як гість'. At the very bottom, there is a copyright notice: '© 2026 Glonin'lyb. Краса, яка надихає.'

Рис. 3.4. Форма реєстрації нового користувача (Reader)

Основна частина інтерфейсу складається з послідовних текстових полів введення, кожне з яких супроводжується зрозумілим підписом та внутрішньою текст-підказкою для спрощення процесу заповнення. Перше поле містить іконку користувача (Reader) та підказку, наступне поле обладнане іконкою конверта та дефолтним прикладом. Блоки для роботи з паролями (із підказкою «Мінімум 6 символів») та підтвердження пароля (Повторіть пароль) додатково оснащені графічними іконками перекресленого ока з правого боку, що дозволяє користувачеві (Reader) одним кліком перемикає видимість символів для запобігання помилок при введенні. Особливістю архітектури реєстраційної форми є інтегрований інтерактивний перемикач у блоці зареєструватися як автор з додатковим можливостями, а саме автоматичним розширенням рольової моделі

користувача (Reader) безпосередньо в момент створення облікового запису. Замість стандартного статусу звичайного користувача, активація цього перемикача автоматично присвоює акаунту роль візажиста, що повністю змінює доступний функціонал системи після успішної авторизації. Це відкриває користувачеві (Reader) прямий доступ до створення нових публікацій, які після проходження модерації з'являються у загальній стрічці (рис. 3.3).

Окрім цього, для новоспеченого автора автоматично генерується персональна сторінка-профіль куди інтегрується інтерактивна галерея для ведення власного портфоліо робіт та табличний модуль для прийому й обробки клієнтських заявок. Цей елемент дозволяє користувачеві (Reader) самостійно обрати свій початковий статус у системі: залишитися у ролі звичайного користувача або надіслати запит на активацію ролі візажиста (Writer). Наступним на огляді персональний акаунт автора детальну структуру якого представлено на рис. 3.5. Інтерфейс цієї зони спроектовано максимально просто та зрозуміло, розділяючи інформаційну частину та візуальний блок робіт.

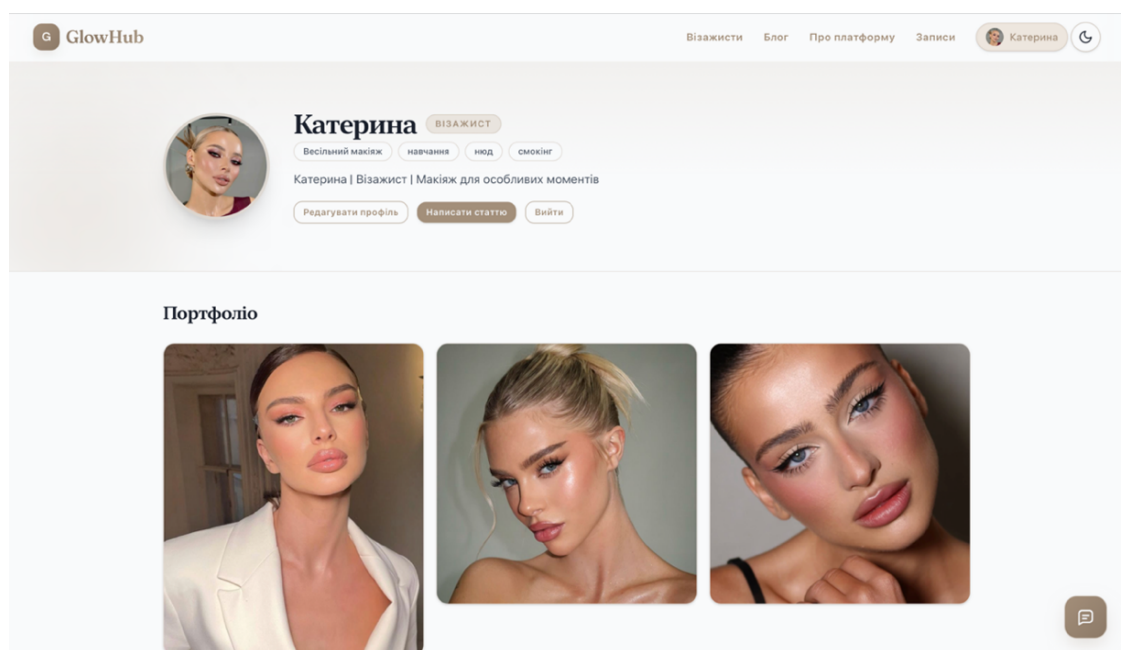


Рис. 3.5. Персональна сторінка-профіль

У кожного автора наявний детальний блок самопрезентації та інструменти взаємодії з аудиторією. Тут відображається інформаційна картка майстра з його аватаром, іменем, тегами робочих спеціалізацій та кнопками керування

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

профілем. Також на цій сторінці обов'язково присутня адаптивна галерея портфоліо для демонстрації найкращих б'юті-робіт клієнтам та функціональна кнопка для швидкого створення нових публікацій.

Основну нижню площу екрана займає великий блок з портфоліо. Він реалізований у вигляді адаптивної горизонтальної сітки, де відображаються якісні фотографії виконаних макіяжів, що демонструють професійний рівень майстра потенційним клієнтам. На рисунку у файлі чітко видно три великі закруглені картки з різними б'юті-роботами автора. Додатково у правому нижньому кутку інтерфейсу закріплено круглу плаваючу кнопку AI-консультанта з іконкою повідомлення, яка завжди залишається під рукою для швидкого виклику інтелектуального помічника.

При натисканні на цю кнопку перед користувачем (Reader) відкривається інтерактивний інтелектуальний помічник, розгорнутий вигляд якого продемонстровано на рис. 3.6. Інтерфейс цього модуля спроектований у вигляді вертикального спливаючого вікна, що імітує сучасний месенджер, і містить чіткий підзаголовок



Рис. 3.6. Вікно з наявним AI-консультантом

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

У верхній панелі чату з правого боку розміщено круглу кнопку з хрестиком для швидкого згортання віджета. Центральну частину екрана займає робоча зона діалогу, де репліки користувача (Reader) підсвічуються кольоровим блоком із зазначенням часу відправки, розгорнуті текстові відповіді штучного інтелекту відображаються у вигляді структурованих списків із покроковими інструкціями навігації сайтом. Нижня частина інтерфейсу повністю відведена під інструменти введення та надсилання повідомлень. Безпосередньо під вікном введення з лівого боку вбудовано динамічний лічильник символів, який допомагає контролювати обсяг повідомлення, а з правого боку симетрично розташовані два текстові контролери дії: функціональна кнопка очистити для миттєвого видалення поточної історії листування та акцентна кнопка надіслати з іконкою паперового літачка.

Важливою складовою підсистеми керування користувацькими даними вебзастосунку є інтерфейс персоналізації профілю для користувачів із роллю візажиста (Writer). При натисканні автором на кнопку (редагувати профіль) поверх основного вмісту сторінки розгортається модальне вікно редагування особистого профілю автора, детальний вигляд якого продемонстровано на рис. 3.7.

Редагувати профіль

Аватар
☐ Вибрати фото

Про себе
 Катерина | Візажист | Макіяж для особливих моментів

Спеціалізації
 Весільний макіяж, навчання, нюд, смокінг

Введіть через кому

Instagram
<https://instagram.com/yourusername>

Facebook
<https://facebook.com/yourpage>

TikTok
<https://tiktok.com/@yourusername>

Зберегти Скасувати

Рис. 3.7. Вікно редагування особистого профілю автора

Інтерфейс цього вікна спроектований у вигляді чіткої та структурованої форми з напівпрозорою підкладкою, яка затіняє задній план. Основна частина форми на містить послідовні інтерактивні поля для оновлення професійної інформації про автора. Поле «Про себе» реалізовано у вигляді великого текстового вікна, де введено актуальне опис «Катерина | Візажист | Макіяж для особливих моментів». Нижче розташоване поле, у якому автор може вказати поточні напрями роботи. Для інтеграції з соціальними мережами в інтерфейсі передбачено три окремі поля введення посилань, що дозволяє майстру підтримувати зв'язок із клієнтами на різних майданчиках.

Наступним на огляді сторінка з написанням статті, при натисканні автором на акцентну кнопку система перенаправляє користувача (Reader) до робочого простору публікацій, вигляд якого детально продемонстровано на рис. 3.7. Інтерфейс під назвою «Створити новий пост» організований у вигляді двоколонкового макета, де ліва частина відведена під форму заповнення даних, а права під інтерактивне прев'ю картки статті. Ліва панель форми містить усі необхідні поля для конструювання майбутньої публікації.

Рис. 3.8. Сторінка зі створенням нового поста

Для створення публікації в лівій частині форми розробники передбачили кілька спеціалізованих текстових полів введення та інтерактивних зон. Спочатку йде поле «Заголовок» із підказкою, під яким розташоване вікно «Короткий опис»

для додавання тексту-прев'ю та SEO-метаданих. Наступним елементом є технічний блок із заповнювачем поля auto-generated-from-title, що відповідає за автоматичне формування людинозрозумілого посилання на статтю. Процес завантаження медіафайлів реалізовано через область «Обкладинка» з пунктирним трафаретом та інструкцією щодо форматів (PNG, JPG, WebP до 10MB), а для набору основного матеріалу виділено масштабну зону «Контент (Markdown)» з підтримкою полегшеної розмітки тексту. Права частина інтерфейсу на рис. 3.8 містить динамічний блок «ПРЕВ'Ю КАРТКИ», який у реальному часі відображає вигляд майбутнього поста у загальній стрічці. На завантаженій обкладинці з косметичним приладдям автоматично закріплюється індикатор із поточною датою.

Також у вебзастосунку наявна вбудована темна тема інтерфейсу, яка адаптує всі графічні елементи сайту для комфортного використання в умовах низького освітлення, як детально продемонстровано на прикладі сторінки (рис. 3.9). Перемикання між колірними режимами здійснюється в один клік за допомогою круглої кнопки із зображенням сонця або місяця, що розташована у крайньому правому кутку верхньої панелі навігації поруч із профілем користувача (Reader).

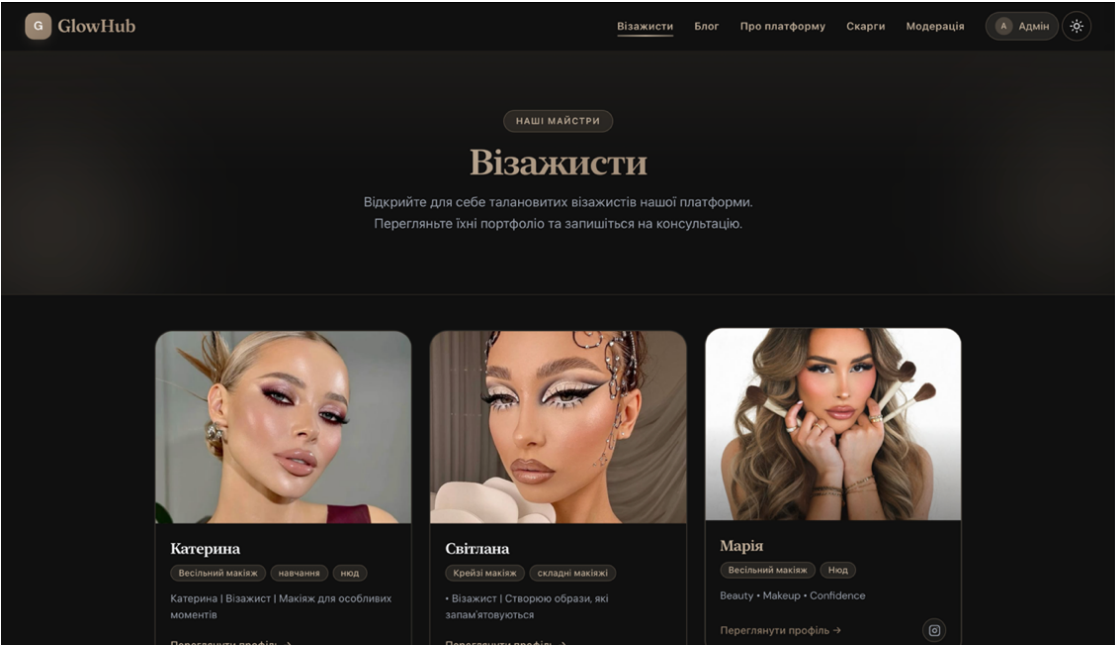


Рис. 3.9. Темна тема інтерфейсу

При активації нічного режиму, світле тло сайту змінюється на глибокий чорний колір, а текстові блоки та заголовки автоматично інвертуються у контрастні світлі відтінки.

Наступним ми розглянемо сторінку звичайного користувача детальну структуру якої представлено на рис. 3.10. На відміну від кабінету автора, інтерфейс користувача з роллю користувача (Reader) є більш лаконічним і зосередженим на відстеженні контенту. На цій сторінці наявна інформаційна картка профілю, де користувач може завантажити чи оновити власну аватарку, а також переглядати список улюблених візажистів, на яких він підписався.

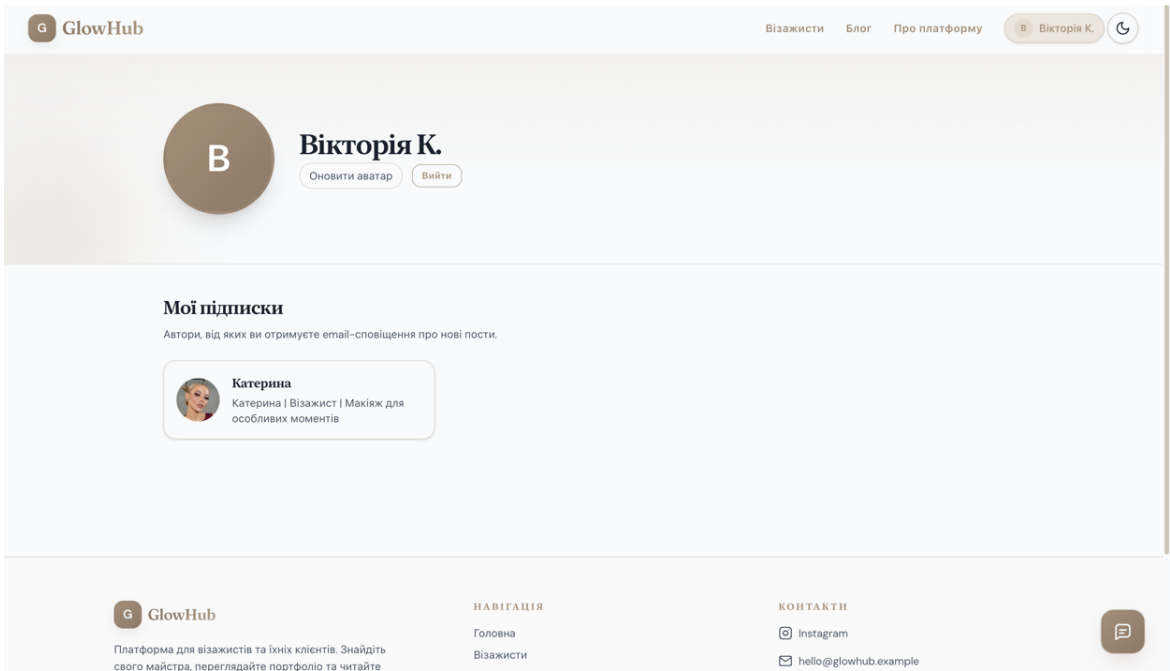


Рис. 3.10. Персональна сторінка користувача

Основну частину сторінки займає унікальний блок «Мої підписки» при підписці на автора (Writer), користувач (Reader) буде отримувати email-сповіщення про нові пости автора, на якого підписаний. У цій зоні відображаються картки відстежуваних майстрів, що дозволяє користувачеві швидко переходити до їхніх профілів.

Також користувач має змогу записатися до будь-якого обраного майстра. Переходячи на персональну сторінку візажиста (Writer), користувач бачить його портфоліо та інтерактивну форму з можливістю поставити запитання або надіслати заявку на консультацію (рис. 3.11).

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

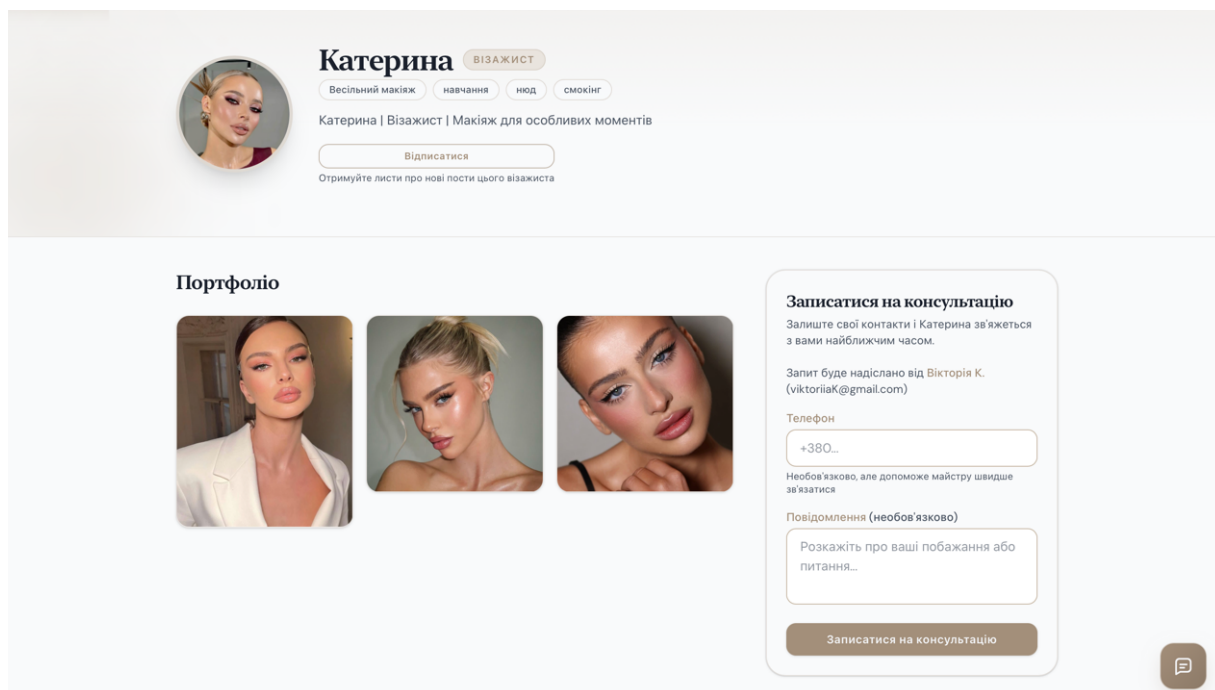


Рис. 3.11. Сторінка профілю майстра з формою запису на консультацію

Основна робоча зона розділена на два блоки: ліворуч представлена галерея портфоліо для демонстрації б'юті-робіт, а праворуч модуль «Записатися на консультацію». У цьому модулі автоматично відображаються дані поточного користувача (Reader), а також наявні поля для введення номера телефону та супровідного повідомлення.

Зі сторони візажиста (Writer) процес обробки та менеджменту отриманих заявок виглядає як окремий робочий простір особистого кабінету, детальний інтерфейс якого представлено на рис. 3.12.

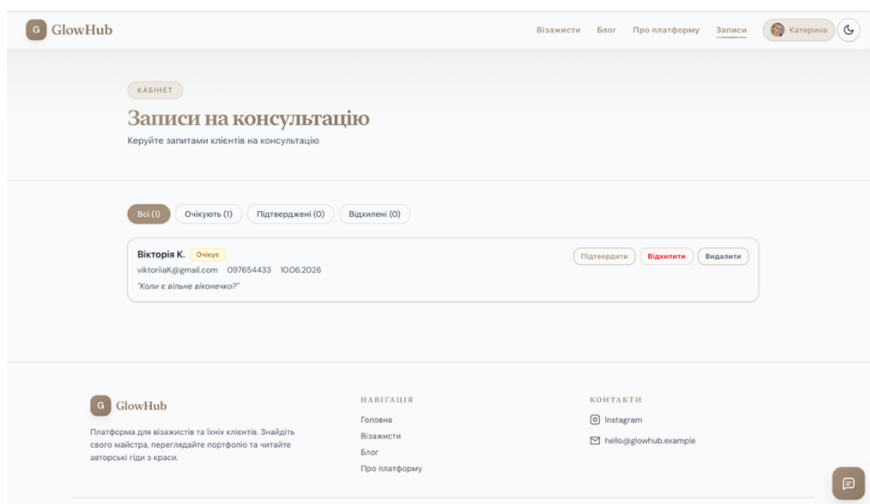


Рис. 3.12. Інтерфейс керування записами на консультацію в кабінеті візажиста

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

Ця сторінка під назвою «Записи на консультацію» розроблена для систематизації клієнтських запитів і містить чіткий орієнтир. Для оптимізації роботи з великою кількістю клієнтів у верхній частині робочої зони на рис. 3.12 інтегровано панель горизонтальних табів-фільтрів, які відображають кількісні показники заявок за їхнім поточним статусом у системі: «Всі (1)», «Очікують (1)», «Підтверджені (0)» та «Відхилені (0)».

Авторизований користувач (Reader) вебзастосунку має змогу залишати коментарі під постами майстрів, як продемонстровано на прикладі інтерфейсу сторінки (рис. 3.13). Наявний інтерактивний блок «Коментарі», де відображається загальна кількість залишених відгуків та стрічка відповідей від користувачів. Кожне повідомлення містить круглу іконку профілю, ім'я автора коментаря, дату публікації та сам текст відгуку.

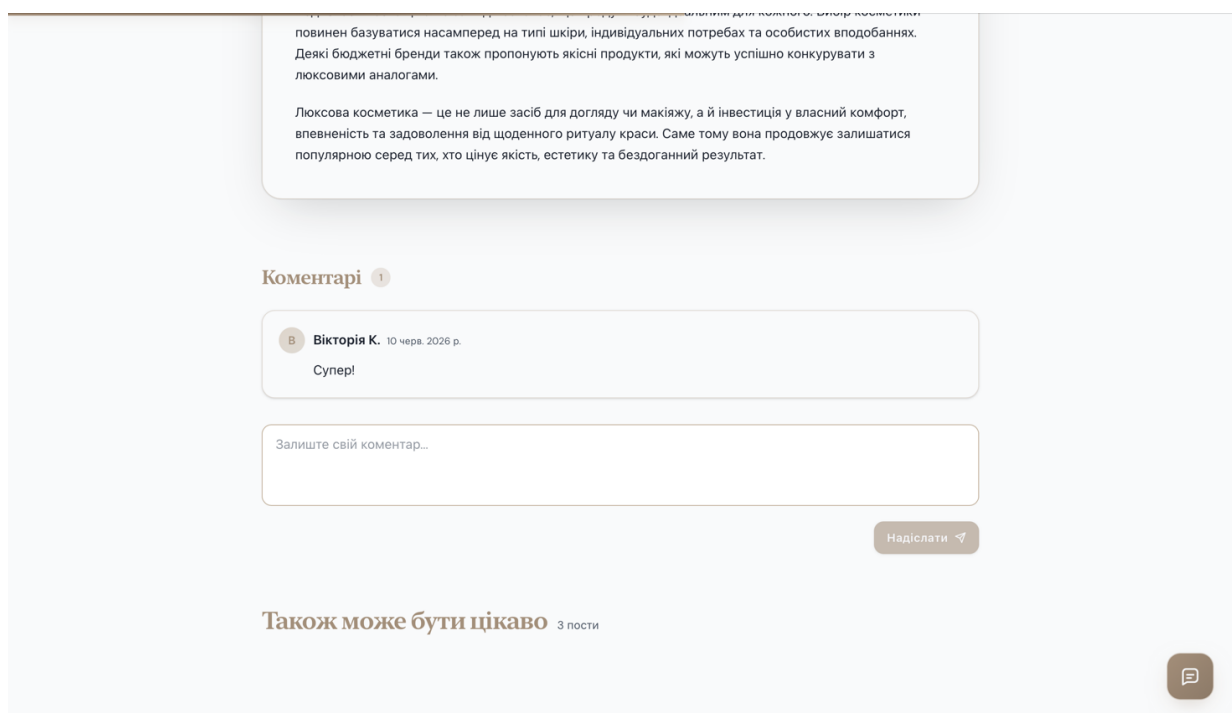


Рис. 3.13. Блок коментарів на сторінці статті

Для користувачів, які пройшли авторизацію в системі, доступна спеціальна форма зворотного зв'язку, що складається з текстового поля та активної кнопки «Надіслати» для миттєвого додавання запису. Якщо користувач (Reader) не авторизований у системі, то така можливість відсутня, замінююється повідомленням із пропозицією увійти до свого облікового запису. Завершує

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

сторінку рекомендаційний блок «Також може бути цікаво», який пропонує інші пости для перегляду.

Користувач (Reader) має змогу не лише підтримати автора коментарем, але й поскаржитись на публікацію в разі виявлення порушень правил платформи. Ця функція реалізована за допомогою інтерактивного модального вікна, яке накладається поверх сторінки статті, як показано на рис. 3.14.

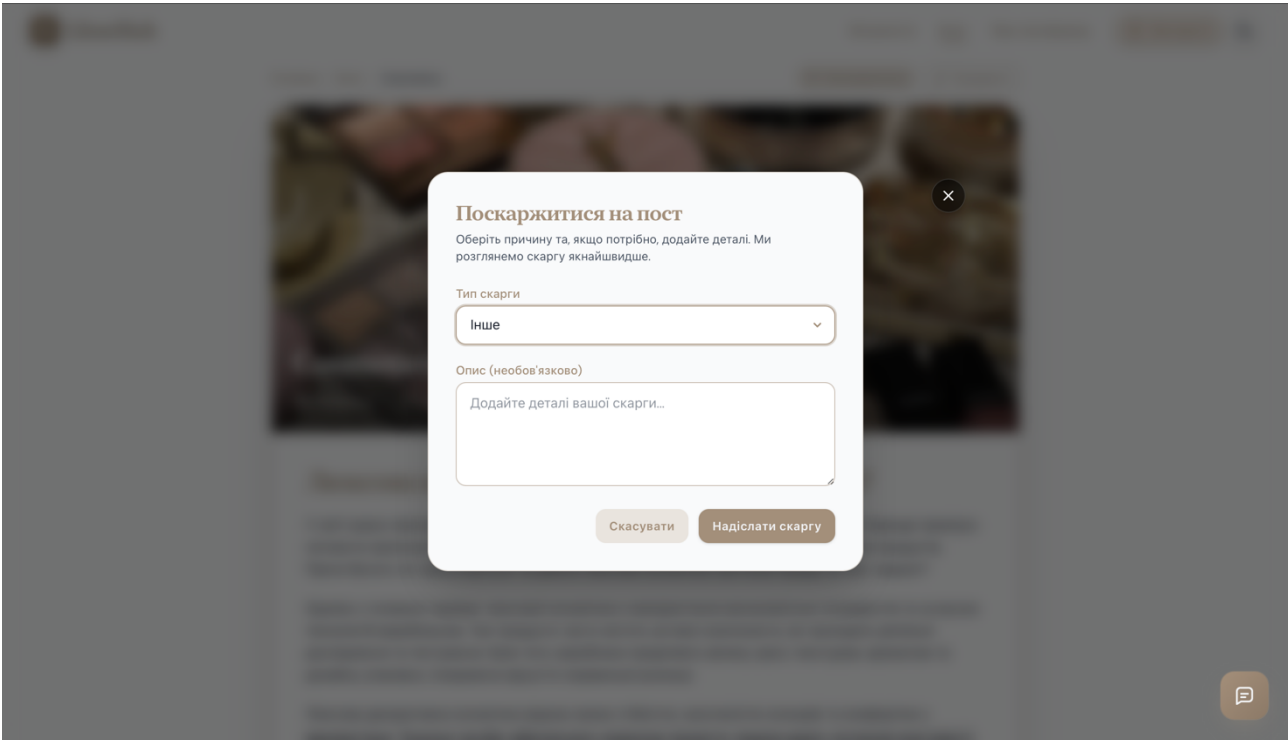


Рис. 3.14. Інтерфейс форми модерації та зворотного зв'язку користувача (Reader)

Форма має чіткий заголовок «Поскаржитися на пост» та містить коротку інструкцію, яка закликає користувача обрати причину й додати деталі для швидкого розгляду звернення командою модераторів. Для категоризації проблеми в інтерфейсі передбачено розкривний список «Тип скарги», де користувач (Reader) може обрати відповідний пункт. Нижче розташоване додаткове поле «Опис (необов'язково)» із плейсхолдером «Додайте деталі вашої скарги...», яке дозволяє користувачеві (Reader) розгорнуто пояснити причину свого невдоволення.

Процес взаємодії з цим вікном завершується за допомогою двох кнопок у нижній частині: «Скасувати» для повернення до читання або «Надіслати скаргу»

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

для її фіксації в системі. Також вікно можна миттєво закрити, натиснувши на кнопку-хрестик у верхньому правому кутку форми.

Коли користувач (Reader) надсилає скаргу на публікацію, вона автоматично надходить до особистого кабінету адміністратора платформи (Admin). Інтерфейс цієї робочої панелі для опрацювання звернень представлено на сторінці «Адмін-панель / Скарги» на рис.3.15. У адміністратора (Admin) реалізовано блок фільтрації для швидкого пошуку потрібних заявок. Він складається з випадаючого списку «Статус» та текстового поля «Slug поста», яке дозволяє фільтрувати скарги на конкретну статтю за її унікальною адресою.

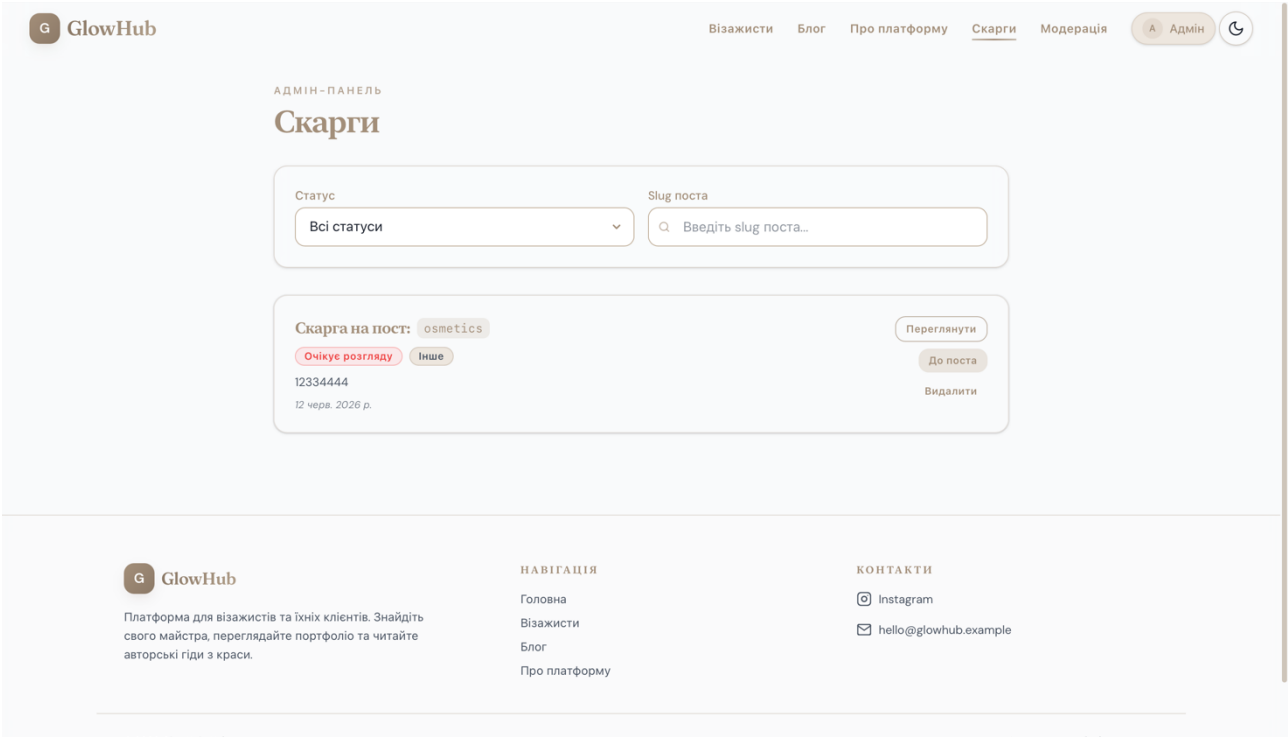


Рис. 3.15. Робоче вікно адміністратора платформи з модулем модерації контенту

Нижче розташовується безпосередньо стрічка з картками отриманих скарг. Кожна така картка містить детальну інформацію про інцидент, зокрема назву елемента «Скарга на пост:». Усередині картки виводиться опис порушення, залишений користувачем (Reader), та дата фіксації скарги в системі. Праворуч передбачено блок дій для модерації, де адміністратор (Admin) може натиснути «Переглянути» для детального аналізу, скористатися кнопкою щоб впевнитися у наявності порушень та прийняти остаточне рішення щодо модерації контенту.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо факт недотримання правил підтверджується, адміністратор (Admin) має всі необхідні інструменти для оперативного реагування та видалення матеріалу з платформи або вибрати опцію «Видалити» для архівації чи відхилення запиту.

Окрім опрацювання скарг, адміністратор (Admin) платформи також виконує функцію премодерації нового контенту. Інтерфейс робочого простору для перевірки публікацій перед їхнім оприлюдненням представлено на рис. 3.16. Цей розділ супроводжується чітким системним поясненням. Такий підхід дозволяє підтримувати високу якість матеріалів на платформі та запобігати публікації спаму чи невідповідного контенту.

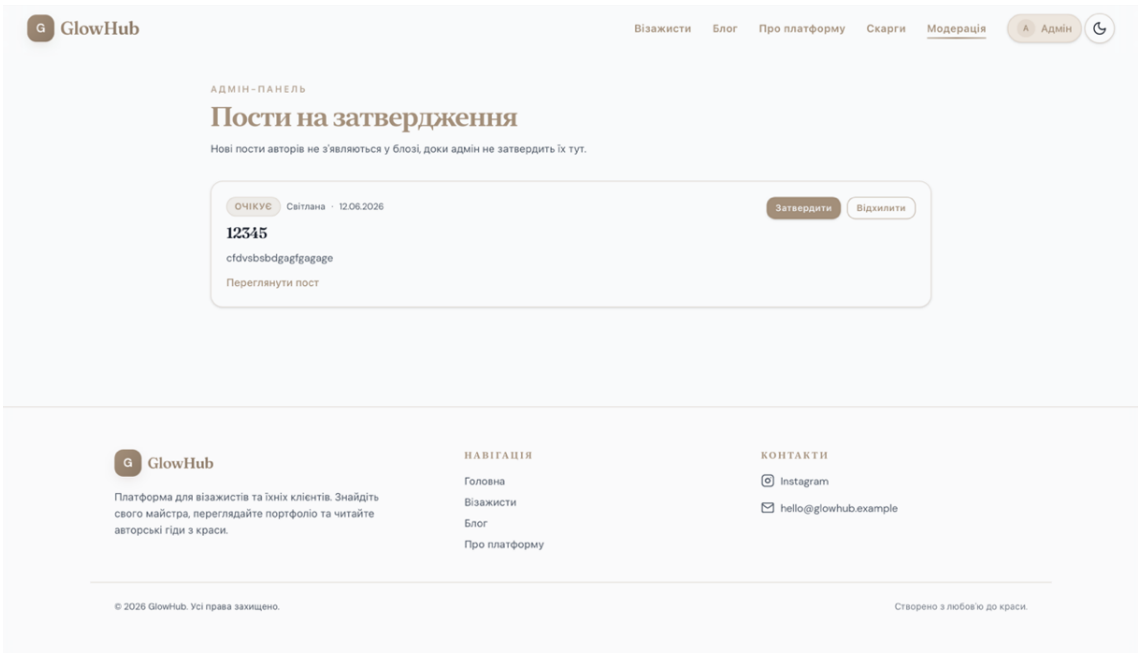


Рис. 3.16. Інтерфейс робочого простору для перевірки публікацій

Безпосередньо на робочій панелі відображається список карток статей, що очікують на перевірку. У структурі кожної картки виводяться метадані, такі як сірий маркер-індикатор «Очікує», ім'я автора публікації та дата створення запиту. Поруч відображаються заголовок, текстовий фрагмент статті та інтерактивне посилання «Переглянути пост» для повного ознайомлення з матеріалом. Адміністратор (Admin) може натиснути кнопку «Затвердити», після чого стаття миттєво публікується в загальному блозі платформи, або скористатися кнопкою «Відхилити» для скасування публікації та повернення її автору.

3.3 Тестування роботи вебзастосунку

Після завершення розробки вебзастосунку «GlowHub» було проведено функціональне тестування основних модулів системи з метою перевірки коректності роботи реалізованого функціоналу, стабільності взаємодії між клієнтською та серверною частинами, а також правильності обробки даних.

Тестування виконувалося шляхом моделювання реальних сценаріїв використання системи користувачами різних ролей: гостя, користувача (Reader), візажиста (Writer) та адміністратора (Admin). Під час перевірки оцінювалася коректність роботи механізмів реєстрації та авторизації, створення записів на консультацію, завантаження зображень, взаємодії з AI-консультантом та системи керування контентом. Додатково було перевірено роботу рольової моделі доступу, коректність відображення інтерфейсів для різних категорій користувачів та правильність обробки помилок під час введення некоректних даних. Особлива увага приділялася перевірці взаємодії із зовнішніми сервісами, зокрема Cloudinary для зберігання медіафайлів та OpenAI API для забезпечення роботи AI-консультанта.

Таблиця 3.1

Результати функціонального тестування вебзастосунку

№	Тестовий сценарій	Очікуваний результат	Результат
1	Реєстрація нового користувача	Створення облікового запису	Успішно
2	Авторизація користувача	Успішний вхід до системи	Успішно
3	Перегляд каталогу візажистів	Відображення списку візажистів	Успішно
4	Перегляд портфоліо візажиста	Відображення робіт майстра	Успішно

5	Онлайн-запис на консультацію	Створення заявки в системі	Успішно
6	Додавання візажиста в обране	Створення запису в списку обраних	Успішно
7	Додавання нової роботи до галереї	Збереження запису та фото	Успішно
8	Створення публікації в блозі	Відображення нової статті	Успішно
9	Додавання коментаря до статті	Збереження коментаря	Успішно
10	Надсилання повідомлення AI-консультанту	Отримання відповіді від AI	Успішно
11	Створення скарги на контент	Збереження скарги в БД	Успішно
12	Підписка на візажиста	Створення запису в subscriptions	Успішно

Результати функціонального тестування, наведені в таблиці 3.1, підтверджують коректну роботу основних модулів вебзастосунку. Усі перевірені сценарії були виконані успішно, що свідчить про працездатність механізмів реєстрації та авторизації користувачів, роботи з контентом, онлайн-запису на консультації, взаємодії з AI-консультантом та інших функціональних можливостей системи. Для більш детальної оцінки якості програмного продукту додатково було проведено тестування окремих характеристик вебзастосунку, результати якого наведено в наступній таблиці.

Таблиця 3.2

Результати тестування безпеки та розмежування доступу

№	Тестовий сценарій	Очікуваний результат	Результат
1	Авторизація з невірним паролем	Відмова у вході	Успішно

2	Доступ гостя до особистого кабінету	Заборона доступу	Успішно
3	Доступ користувача до панелі адміністратора	Заборона доступу	Успішно
4	Доступ користувача до керування галереєю візажиста	Заборона доступу	Успішно
5	Доступ візажиста до адміністративних функцій	Заборона доступу	Успішно
6	Використання простроченого JWT-токена	Відмова в доступі	Успішно
7	Модерація скарги адміністратором	Зміна статусу скарги	Успішно
8	Додавання публікації авторизованим візажистом	Успішне створення запису	Успішно
9	Оновлення профілю власником облікового запису	Дані успішно змінено	Успішно

Під час тестування критичних помилок, що перешкоджали б роботі системи, виявлено не було. Усі функціональні модулі коректно взаємодіють між собою та забезпечують виконання передбачених бізнес-процесів. Особливу увагу було приділено перевірці роботи механізму авторизації на основі JWT-токенів та взаємодії із зовнішніми сервісами Cloudinary і OpenAI API.

За результатами проведеного тестування встановлено, що вебзастосунок відповідає визначеним функціональним вимогам, забезпечує стабільну роботу основних модулів та може бути використаний для автоматизації діяльності студії візажистів.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до третього розділу

У третьому розділі роботи було детально спроектовано, розроблено та практично реалізовано клієнтську (Reader), авторську (Writer) та адміністративну (Admin) частини вебзастосунку GlowHub. Під час розробки особливу увагу приділено організації зручного користувацького інтерфейсу, розмежуванню прав доступу та реалізації функціональних можливостей для різних категорій користувачів.

У межах вебзастосунку реалізовано систему реєстрації та авторизації користувачів (Reader), каталог візажистів (Writer) із портфоліо робіт, механізм онлайн-запису на консультації, статті із можливістю коментування публікацій, систему підписок, а також AI-консультанта для надання інформаційної підтримки користувачам. Для майстрів (Writer) передбачено функції керування власним профілем, послугами та галереєю робіт, тоді як адміністратор (Admin) отримує інструменти модерації контенту, опрацювання скарг і контролю роботи системи.

Для забезпечення безпеки та якості інформаційного наповнення впроваджено механізм модерації контенту та систему подання скарг. Користувачі (Reader) можуть повідомляти про порушення правил платформи, а адміністратор (Admin) має можливість переглядати звернення та приймати рішення щодо подальшого опрацювання контенту.

Окрему увагу було приділено перевірці працездатності розробленого програмного забезпечення. Проведене функціональне тестування охопило основні сценарії використання вебзастосунку, зокрема реєстрацію та авторизацію користувачів (Reader), роботу з публікаціями, онлайн-запис на консультації, взаємодію з AI-консультантом, створення коментарів, підписок і скарг.

Результати тестування підтвердили коректну роботу всіх перевірених модулів та відповідність реалізованого функціоналу поставленим вимогам. У результаті виконання третього розділу було реалізовано повнофункціональний вебзастосунок студії візажистів з інтегрованим AI-консультантом, який

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

забезпечує зручну взаємодію між користувачами (Reader) та майстрами (Writer), підтримує ефективне керування контентом і записами на консультації, а також відповідає вимогам сучасних вебтехнологій щодо функціональності, безпеки та зручності використання.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було успішно вирішено актуальне науково-практичне завдання, що полягає у проєктуванні, розробці та тестуванні вебзастосунку студії візажистів з інтегрованим AI-консультантом «GlowHub». Створений програмний комплекс поєднує в собі сучасні інтернет-технології з інструментами штучного інтелекту, що дозволяє автоматизувати рутинні бізнес-процеси в індустрії краси, покращити взаємодію з клієнтами та оптимізувати менеджмент б'юті-студії.

На основі детального аналізу предметної області та існуючих програмних рішень-аналогів було виявлено їхні ключові недоліки, такі як обмеженість функціоналу прямої комунікації, територіальні рамки та слабка персоналізація сервісів. Це дозволило обґрунтувати необхідність розробки власного гнучкого вебрішення. Для реалізації платформи було обрано архітектурний патерн розподілення клієнтської та серверної частин (Frontend/Backend) та впроваджено сучасний стек технологій: бібліотеку React з інструментом збірки Vite, середовище Node.js із фреймворком Express.js, а також мову TypeScript на всіх рівнях розробки для забезпечення суворої статичної типізації коду.

У межах інфологічного моделювання було розроблено гнучку NoSQL-структуру бази даних у середовищі MongoDB. Структура складається з 8 взаємопов'язаних колекцій (users, posts, comments, complaints, gallery, bookings, chatSessions, subscriptions), що дозволило оптимально організувати збереження інформації про користувачів (Reader), медіа-портфоліо майстрів, онлайн-записи та текстові сесії. Інтеграція сторонніх мікросервісів, таких як хмарне сховище Cloudinary для оптимізації важкого медіаконтенту та бібліотека Nodemailer для автоматичної SMTP-доставки email-сповіщень, дозволила суттєво знизити навантаження на сервер.

Інноваційним ядром платформи стала реалізація контекстного AI-консультанта на базі офіційного OpenAI API. Інтегрований інтелектуальний

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

помічник функціонує в режимі реального часу, обробляє запити природною мовою, враховує контекст діалогу та надає клієнтам персоналізовані рекомендації щодо б'юті-послуг і навігації платформою. Безпека системи та розмежування прав доступу забезпечується багаторівневою рольовою моделлю користувачів та двотокенною схемою автентифікації на базі стандарту JWT із використанням захищених файлів cookies. Створений адаптивний інтерфейс користувача (Reader) з підтримкою світлої та темної тем, а також впроваджена адмін-панель для премодерації контенту й опрацювання скарг повністю задовольняють вимоги зручності та безпеки. Проведене тестування розробленого програмного комплексу підтвердило його повну відповідність усім поставленим функціональним та нефункціональним вимогам, високу швидкість відгуку, надійність обробки асинхронних запитів.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тарасюк М.О., Чижмотря О.В. Вебплатформа студії візажистів з AI-консультантом. Тези Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених «Тенденції розвитку технологій в автоматизації, електроенергетиці, приладобудуванні та робототехніці» (присвяченої Дню науки), м. Житомир, 13-15 травня 2026 р. – Житомир: Житомирська політехніка, 2026.
2. Тарасюк М.О., Чижмотря О.В. Автоматизація онлайн-бронювання у вебсервісах сфери beauty-послуг. Тези Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених «Тенденції розвитку технологій в автоматизації, електроенергетиці, приладобудуванні та робототехніці» (присвяченої Дню науки), м. Житомир, 13-15 травня 2026 р. – Житомир: Житомирська політехніка, 2026.
3. React Documentation. Офіційна документація React. URL: <https://react.dev> (дата звернення: 08.06.2026).
4. Node.js Documentation. Офіційна документація Node.js. URL: <https://nodejs.org> (дата звернення: 08.06.2026).
5. Express.js Guide. Офіційна документація Express.js. URL: <https://expressjs.com> (дата звернення: 09.06.2026).
6. TypeScript Handbook. Офіційна документація TypeScript. URL: <https://www.typescriptlang.org> (дата звернення: 08.06.2026).
7. MongoDB Manual. Офіційна документація MongoDB. URL: <https://www.mongodb.com/docs> (дата звернення: 08.06.2026).
8. Mongoose Documentation. URL: <https://mongoosejs.com/docs> (дата звернення: 08.06.2026).
9. REST API Tutorial. URL: <https://restfulapi.net> (дата звернення: 08.06.2026).
10. JWT Introduction. URL: <https://jwt.io/introduction> (дата звернення: 08.06.2026).
11. OpenAI API Documentation. URL: <https://platform.openai.com/docs> (дата звернення: 08.06.2026).

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

12. Cloudinary Documentation. URL: <https://cloudinary.com/documentation> (дата звернення: 08.06.2026).
13. HTTP/1.1 Specification (RFC 9110). URL: <https://www.rfc-editor.org/rfc/rfc9110> (дата звернення: 08.06.2026).
14. Nodemailer Documentation URL: <https://nodemailer.com/about> (дата звернення: 08.06.2026).
15. React Query Documentation. URL: <https://tanstack.com/query/latest> (дата звернення: 08.06.2026).
16. MDN Web Docs. JavaScript, HTTP, API. URL: <https://developer.mozilla.org> (дата звернення: 08.06.2026).
17. Zustand Documentation. URL: <https://zustand-demo.pmnd.rs> (дата звернення: 08.06.2026).
18. Swagger / OpenAPI Specification. URL: <https://swagger.io/specification> (дата звернення: 08.06.2026).
19. SEO Starter Guide. Google Search Central. URL: <https://developers.google.com/search/docs> (дата звернення: 08.06.2026).
20. Orval API Client Generator Documentation. URL: <https://orval.dev> (дата звернення: 08.06.2026).
21. Tailwind CSS. Utility-first CSS framework for rapid UI development. URL: <https://tailwindcss.com/docs> (дата звернення: 08.06.2026).
22. React Hook Form Documentation. URL: <https://react-hook-form.com> (дата звернення: 08.06.2026).
23. React Markdown Component Reference. URL: <https://github.com/remarkjs/react-markdown> (дата звернення: 08.06.2026).
24. Multer Middleware for Express (Handling Multipart/Form-Data). URL: <https://github.com/expressjs/multer> (дата звернення: 08.06.2026).
25. Cross-Origin Resource Sharing (CORS) W3C Recommendation. URL: <https://www.w3.org/TR/cors> (дата звернення: 08.06.2026).
26. RFC 6265: HTTP State Management Mechanism (HTTP Cookies spec). URL: <https://www.rfc-editor.org/rfc/rfc6265>

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Vite Documentation. URL: <https://vitejs.dev/guide/> (дата звернення: 08.06.2026).
28. bcrypt Documentation. URL: <https://github.com/kelektiv/node.bcrypt.js> (дата звернення: 08.06.2026).
29. Cookie Parser Middleware Documentation. URL: <https://expressjs.com/en/resources/middleware/cookie-parser.html> (дата звернення: 08.06.2026).
30. MongoDB Atlas Documentation. URL: <https://www.mongodb.com/atlas/database> (дата звернення: 08.06.2026).
31. JSON Web Tokens Best Practices. Auth0 Documentation. URL: <https://auth0.com/docs/secure/tokens/json-web-tokens> (дата звернення: 08.06.2026).
32. Cloudinary Image Upload API Reference. URL: https://cloudinary.com/documentation/image_upload_api_reference (дата звернення: 08.06.2026).

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграма варіантів використання

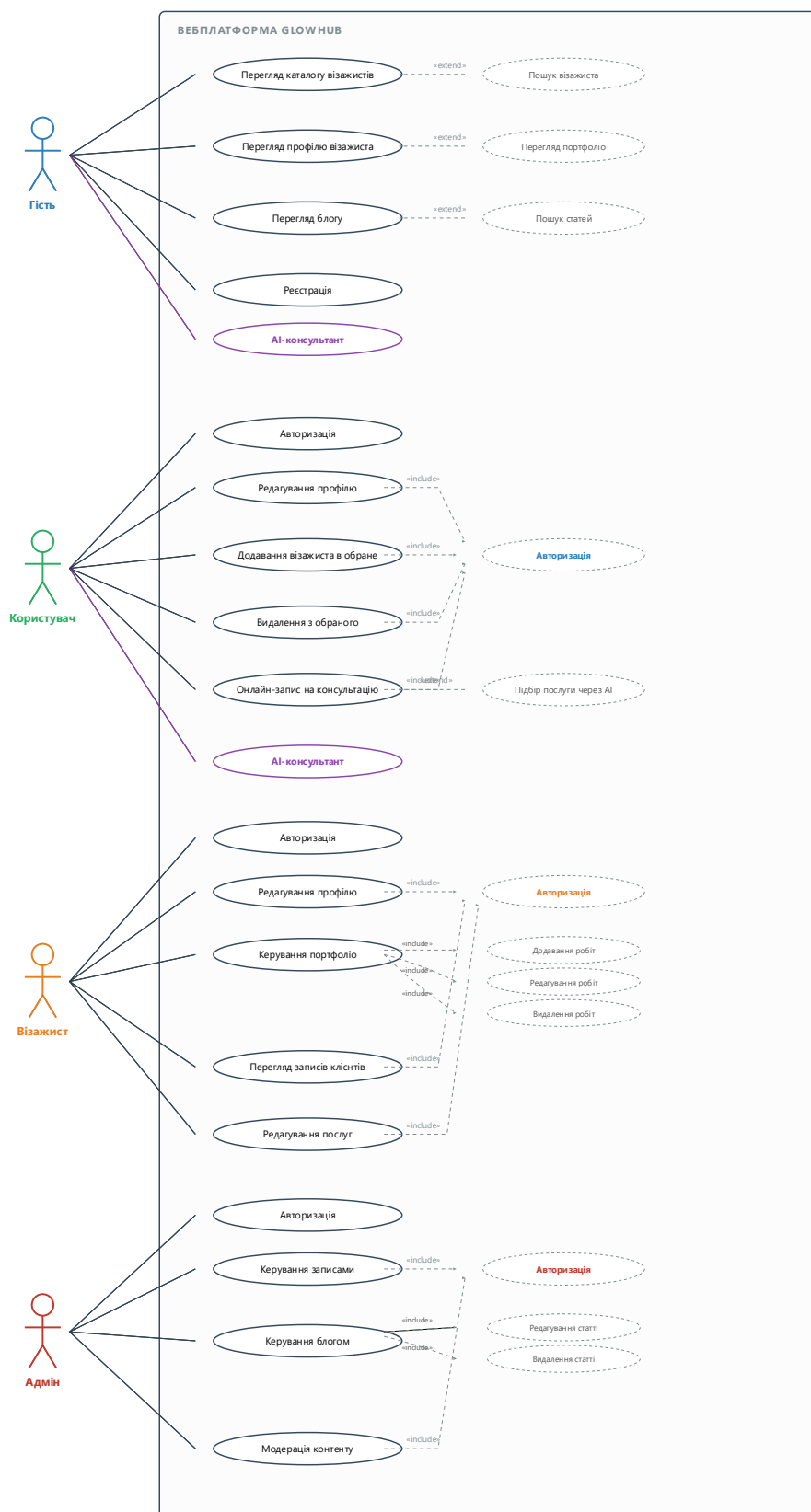


Рис. А.1. Діаграма варіантів використання

ЛІСТИНГ: userModel.ts

```

import mongoose, { Schema, Document } from "mongoose";
import { applyBaseSchemaConfig } from "../utils/baseSchemaConfig";

export interface IUser extends Document {
  username: string;
  email: string;
  password: string;
  role: 'admin' | 'writer' | 'reader';
  bio?: string;
  avatar?: string;
  specializations?: string[];
  socialLinks?: {
    instagram?: string;
    facebook?: string;
    tiktok?: string;
  };
};

const userSchema: Schema = new Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: {
    type: String,
    enum: ['admin', 'writer', 'reader'],
    default: 'reader',
    required: true,
  },
  bio: { type: String },
  avatar: { type: String },
  specializations: [{ type: String }],
  socialLinks: {
    instagram: { type: String },
    facebook: { type: String },
    tiktok: { type: String },
  },
});

applyBaseSchemaConfig(userSchema);

export default mongoose.model<IUser>("User", userSchema);

import mongoose, { Schema, Document } from "mongoose";
import { applyBaseSchemaConfig } from "../utils/baseSchemaConfig";

export interface IBooking extends Document {
  name: string;

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

email: string;
phone?: string;
message?: string;
artist: mongoose.Types.ObjectId;
status: 'pending' | 'confirmed' | 'rejected';
createdAt: Date;
updatedAt: Date;
}

const bookingSchema: Schema = new Schema(
{
  name: { type: String, required: true },
  email: { type: String, required: true },
  phone: { type: String },
  message: { type: String },
  artist: { type: Schema.Types.ObjectId, ref: 'User', required: true },
  status: {
    type: String,
    enum: ['pending', 'confirmed', 'rejected'],
    default: 'pending',
    required: true,
  },
},
{ timestamps: true }
);

applyBaseSchemaConfig(bookingSchema);

export default mongoose.model<IBooking>("Booking", bookingSchema);

```

ЛІСТИНГ: bookingModel.ts

```

import mongoose, { Schema, Document } from "mongoose";
import { applyBaseSchemaConfig } from "../utils/baseSchemaConfig";

export type PostStatus = "pending" | "approved" | "rejected";

export interface IPost extends Document {
  slug: string;
  title: string;
  excerpt: string;
  date: Date;
  imageUrl?: string;
  content: string;
  comments: mongoose.Types.ObjectId[];
  author: mongoose.Types.ObjectId;
  status: PostStatus;
  subscribersNotifiedAt?: Date;
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const postSchema: Schema = new Schema({
  slug: { type: String, required: true, unique: true },
  title: { type: String, required: true },
  excerpt: { type: String, required: true },
  date: { type: Date, default: Date.now, required: true },
  imageUrl: { type: String, required: true },
  content: { type: String, required: true },
  comments: [{ type: mongoose.Schema.Types.ObjectId, ref: "Comment" }],
  status: {
    type: String,
    enum: ["pending", "approved", "rejected"],
    default: "pending",
    required: true,
  },
  subscribersNotifiedAt: { type: Date },
  author: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true,
  },
});

applyBaseSchemaConfig(postSchema);

export default mongoose.model<IPost>("Post", postSchema);

```

ЛІСТИНГ: postModel.ts

```

import { Request, Response } from "express";
import ChatService from "../services/chatService";

class ChatController {
  constructor(private chatService: ChatService) {}

  async getChat(req: Request, res: Response) {
    try {
      const userJwt = req.user as { id?: string };
      const session = await this.chatService.getSession(userJwt.id!);

      res.status(200).json(session);
    } catch (error) {
      this.handleError(error, res);
    }
  }

  async sendMessage(req: Request, res: Response) {
    try {
      const userJwt = req.user as { id?: string };
      const { content } = req.body;

      if (typeof content !== "string" || !content.trim()) {
        return res.status(400).json({ message: "content is required" });
      }
    }
  }
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    const session = await this.chatService.sendMessage(userJwt.id!, content);
    const assistantMessage = [...session.messages]
      .reverse()
      .find((message) => message.role === "assistant");

    res.status(201).json({
      message: assistantMessage,
      messages: session.messages,
    });
  } catch (error) {
    this.handleError(error, res);
  }
}

async clearChat(req: Request, res: Response) {
  try {
    const userJwt = req.user as { id?: string };
    const session = await this.chatService.clearSession(userJwt.id!);

    res.status(200).json(session);
  } catch (error) {
    this.handleError(error, res);
  }
}

private handleError(error: unknown, res: Response) {
  if (!(error instanceof Error)) {
    return res.status(500).json({ message: "An unknown error occurred" });
  }

  if (
    error.message === "Message content is required" ||
    error.message === "Message is too long" ||
    error.message === "Invalid user ID"
  ) {
    return res.status(400).json({ message: error.message });
  }

  if (error.message === "OPENAI_API_KEY is not configured") {
    return res.status(503).json({
      message: "AI assistant is not configured yet",
    });
  }

  return res.status(500).json({ message: error.message });
}

export default ChatController;

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						97
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг: chatController.ts

```
import { Request, Response, NextFunction } from "express";
import jwt from "jsonwebtoken";
import { AuthService } from "../services/authService";

declare global {
  namespace Express {
    interface Request {
      user?: string | object;
    }
  }
}

const authService = new AuthService();

export const requireRole = (roles: Array<"admin" | "writer" | "reader">) => {
  return (req: Request, res: Response, next: NextFunction) => {
    const user = req.user as { role?: string };
    if (
      !user ||
      !user.role ||
      !roles.includes(user.role as "admin" | "writer" | "reader")
    ) {
      return res
        .status(403)
        .json({ message: "Access denied: insufficient permissions" });
    }
    next();
  };
};

export const verifyAccessToken = (
  req: Request,
  res: Response,
  next: NextFunction
) => {
  const authHeader = req.headers.authorization;
  const token = authHeader && authHeader.split(" ")[1];

  if (!token) {
    return res
      .status(401)
      .json({ message: "Access token is missing or invalid" });
  }

  const decoded = authService.verifyToken(
    token,
    process.env.ACCESS_SECRET || "access_secret"
  );
  if (!decoded) {
    return res.status(403).json({ message: "Invalid or expired access token" });
  }
};
```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    req.user = decoded;
    next();
  };

export const optionalVerifyAccessToken = (
  req: Request,
  _res: Response,
  next: NextFunction
) => {
  const authHeader = req.headers.authorization;
  const token = authHeader && authHeader.split(" ")[1];

  if (!token) {
    return next();
  }

  const decoded = authService.verifyToken(
    token,
    process.env.ACCESS_SECRET || "access_secret"
  );

  if (decoded) {
    req.user = decoded;
  }

  next();
};

export const handleRefreshToken = (
  req: Request,
  res: Response,
  next: NextFunction
) => {
  const refreshToken = authService.extractRefreshToken(req);

  if (!refreshToken) {
    return res.status(401).json({ message: "Refresh token is missing" });
  }

  const decoded = authService.verifyToken(
    refreshToken,
    process.env.REFRESH_SECRET || "refresh_secret"
  );

  if (!decoded) {
    return res
      .status(403)
      .json({ message: "Invalid or expired refresh token" });
  }

  req.user = decoded;

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						99
Змн.	Арк.	№ докум.	Підпис	Дата		

```
next();
};
```

ЛІСТИНГ: AiChatWidget.tsx

```
import { useEffect, useMemo, useRef, useState } from "react";
import type { FormEvent, KeyboardEvent } from "react";
import axios from "axios";
import { useMutation, useQuery, useQueryClient } from "@tanstack/react-query";
import { useLocation } from "react-router-dom";
import ReactMarkdown from "react-markdown";
import remarkGfm from "remark-gfm";
import type { Components } from "react-markdown";
import toast from "react-hot-toast";
import { clearAiChat, getAiChat, sendAiChatMessage } from "../../api/chat/chat";
import type { AiChatMessage } from "../../api/chat/chat";
import { UserRole } from "../../api/blogAPI.schemas";
import { useGetApiUserMe } from "../../api/users/users";
import { useAuthStore } from "../../store/auth";
import Button from "../../ui/Button";
import Textarea from "../../ui/Textarea";

const CHAT_QUERY_KEY = ["ai-chat"];
const MAX_MESSAGE_LENGTH = 2000;

const getErrorMessage = (error: unknown) => {
  if (axios.isAxiosError<{ message?: string }>(error)) {
    return error.response?.data?.message || "Не вдалося отримати відповідь";
  }
}

return "Не вдалося отримати відповідь";
};

const formatMessageTime = (date?: string) => {
  if (!date) return "";

  return new Date(date).toLocaleTimeString("uk-UA", {
    hour: "2-digit",
    minute: "2-digit",
  });
};

const markdownComponents: Partial<Components> = {
  a: ({ children, node: _node, ...props }) => (
    <a
      {...props}
      className="font-medium text-primary underline decoration-primary/50 underline-offset-2
transition-colors hover:text-primary-hover dark:text-primary-light dark:decoration-primary-light/50
dark:hover:text-primary"
      rel="noopener noreferrer"
      target="_blank"
    >

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {children}
  </a>
),
};

const MessageBubble = ({ message }: { message: AiChatMessage }) => {
  const isUser = message.role === "user";

  return (
    <div className={`flex ${isUser ? "justify-end" : "justify-start"}`>
      <div
        className={`max-w-[85%] min-w-0 rounded-2xl px-4 py-3 text-sm leading-relaxed shadow-sm ${
          isUser
            ? "rounded-br-md bg-primary text-text-light"
            : "rounded-bl-md border border-border/25 bg-surface-elevated text-text dark:border-border-dark/35 dark:bg-surface-elevated-dark dark:text-text-light"
          }`}
        >
        {isUser ? (
          <p className="whitespace-pre-wrap">{message.content}</p>
        ) : (
          <div className="prose prose-sm min-w-0 max-w-none overflow-x-auto [overflow-wrap:anywhere] text-text [word-break:break-word] dark:text-text-light prose-headings:font-display prose-h1:mb-2 prose-h2:mb-1.5 prose-h3:mb-1 prose-p:mb-1.5 prose-p:mt-0 prose-p:leading-relaxed first:prose-p:mt-0 last:prose-p:mb-0 prose-p:text-inherit prose-li:mb-0.5 prose-li:mt-0.5 prose-ul:mb-1.5 prose-ol:mb-1.5 prose-ul:mt-0 prose-ol:mt-0 first:prose-ul:mt-0 first:prose-ol:mt-0 dark:prose-li:marker:text-text-light prose-a:text-inherit prose-headings:font-semibold prose-headings:text-primary prose-headings:leading-tight dark:prose-headings:text-primary-light prose-h1:text-base prose-h2:text-base prose-h3:text-sm prose-blockquote:my-2 prose-blockquote:border-l-primary prose-blockquote:pl-3 prose-blockquote:text-text-muted dark:prose-blockquote:text-text-muted-dark prose-hr:my-3 prose-hr:border-border/30 dark:prose-hr:border-border-dark/50 prose-table:block prose-table:overflow-x-auto prose-th:border prose-th:px-2 prose-th:px-2 prose-th:py-1 prose-th:py-1 dark:prose-th:text-text-light dark:prose-th:text-text-light prose-code:rounded-md prose-code:bg-secondary/50 prose-code:px-1.5 prose-code:py-0.5 prose-code:font-mono prose-code:text-[0.85em] prose-code:before:content-none prose-code:after:content-none dark:prose-code:bg-primary/20 dark:prose-code:text-text-light prose-pre:my-2 prose-pre:overflow-x-auto prose-pre:rounded-lg prose-pre:border prose-pre:border-border/20 prose-pre:bg-[#181818] prose-pre:p-2.5 prose-pre:text-xs prose-pre:leading-normal prose-pre:text-text-light dark:prose-pre:border-border-dark/30 dark:prose-strong:text-text-light">
            <ReactMarkdown
              components={markdownComponents}
              remarkPlugins={[remarkGfm]}
              skipHtml
            >
              {message.content}
            </ReactMarkdown>
          </div>
        )}
        {message.createdAt && (
          <span
            className={`mt-1 block text-[10px] ${

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        isUser ? "text-text-light/75" : "text-text-muted dark:text-text-muted-dark"
      }`
    >
    {formatMessageTime(message.createdAt)}
  </span>
)}
</div>
</div>
);
};

const EmptyState = () => (
  <div className="rounded-2xl border border-dashed border-border/35 bg-secondary/30 p-4 text-sm text-text-muted dark:border-border-dark/45 dark:bg-primary/5 dark:text-text-muted-dark">
    <p className="mb-2 font-semibold text-primary dark:text-primary-light">
      Запитайте AI-консультанта
    </p>
    <p>
      Я допоможу зорієнтуватися на GlowHub, знайти потрібний розділ або підібрати ідею для макіяжу, поки ви очікуєте на реального візажиста.
    </p>
  </div>
);

const AiChatWidget = () => {
  const [isOpen, setIsOpen] = useState(false);
  const [messageInput, setMessageInput] = useState("");
  const [optimisticMessages, setOptimisticMessages] = useState<AiChatMessage[]>([]);
  const messagesEndRef = useRef<HTMLDivElement | null>(null);
  const queryClient = useQueryClient();
  const { pathname } = useLocation();
  const { guestMode } = useAuthStore();
  const isAuthPage = pathname === "/login" || pathname === "/register";

  const { data: userResponse } = useGetApiUserMe({
    query: {
      retry: false,
      refetchOnWindowFocus: false,
      enabled: !guestMode && !isAuthPage,
    },
  });

  const user = userResponse?.data;
  const canUseChat = user?.role === UserRole.reader || user?.role === UserRole.writer;

  const {
    data: chatResponse,
    isLoading: isChatLoading,
    isError: isChatError,
    error: chatError,
  } = useQuery({
    queryKey: CHAT_QUERY_KEY,

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    queryFn: getAiChat,
    enabled: canUseChat,
    refetchOnWindowFocus: false,
  });

  const messages = useMemo(
    () => chatResponse?.data.messages ?? [],
    [chatResponse?.data.messages]
  );
  const visibleMessages = useMemo(
    () => [...messages, ...optimisticMessages],
    [messages, optimisticMessages]
  );

  const sendMutation = useMutation({
    mutationFn: sendAiChatMessage,
    onSuccess: async () => {
      await queryClient.invalidateQueries({ queryKey: CHAT_QUERY_KEY });
      setOptimisticMessages([]);
    },
    onError: (error) => {
      toast.error(getErrorMessage(error));
    },
  });

  const clearMutation = useMutation({
    mutationFn: clearAiChat,
    onSuccess: () => {
      setOptimisticMessages([]);
      queryClient.invalidateQueries({ queryKey: CHAT_QUERY_KEY });
      toast.success("Історію чату очищено");
    },
    onError: () => toast.error("Не вдалося очистити чат"),
  });

  useEffect(() => {
    if (isOpen) {
      messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
    }
  }, [isOpen, visibleMessages.length, sendMutation.isPending]);

  if (isAuthPage || !canUseChat) {
    return null;
  }

  const isSubmitDisabled =
    sendMutation.isPending ||
    !messageInput.trim() ||
    messageInput.length > MAX_MESSAGE_LENGTH;

  const handleSubmit = (event?: FormEvent) => {
    event?.preventDefault();
  }

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const content = messageInput.trim();
if (!content || isSubmitDisabled) return;

setOptimisticMessages([
  {
    role: "user",
    content,
    createdAt: new Date().toISOString(),
  },
]);
setMessageInput("");
sendMutation.mutate(content);
});

const handleKeyDown = (event: KeyboardEvent<HTMLTextAreaElement>) => {
  if (event.key === "Enter" && !event.shiftKey) {
    event.preventDefault();
    handleSubmit();
  }
};

return (
  <div className="fixed bottom-4 right-4 z-50 sm:bottom-6 sm:right-6">
    {isOpen && (
      <section className="mb-4 flex h-[min(680px,calc(100vh-7rem))] w-[calc(100vw-2rem)]
flex-col overflow-hidden rounded-3xl border border-border/30 bg-surface/95 shadow-
[0_24px_80px_-32px_rgba(0,0,0,0.45)] backdrop-blur-xl animate-scale-in dark:border-border-
dark/45 dark:bg-surface-dark/95 sm:w-[420px]">
        <header className="border-b border-border/20 bg-surface-elevated/75 p-4 dark:border-
border-dark/30 dark:bg-surface-elevated-dark/80">
          <div className="flex items-start justify-between gap-3">
            <div>
              <p className="text-xs font-semibold uppercase tracking-[0.22em] text-primary
dark:text-primary-light">
                GlowHub AI
              </p>
              <h2 className="font-display text-xl font-semibold text-text dark:text-text-light">
                AI-консультант
              </h2>
              <p className="mt-1 text-xs text-text-muted dark:text-text-muted-dark">
                Допомога по платформі та базові поради з мейкапу.
              </p>
            </div>
            <button
              type="button"
              className="flex h-9 w-9 shrink-0 cursor-pointer items-center justify-center rounded-full
border border-border/25 text-text-muted transition-colors hover:border-primary/45 hover:text-
primary dark:border-border-dark/35 dark:text-text-muted-dark dark:hover:text-primary-light"
              aria-label="Закрити чат"
              onClick={() => setIsOpen(false)}
            >

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						104
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <svg width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor"
strokeWidth="2.2" strokeLinecap="round" aria-hidden>
          <path d="M18 6 6 18" />
          <path d="m6 6 12 12" />
        </svg>
      </button>
    </div>
  </header>

  <div className="flex-1 space-y-3 overflow-y-auto p-4">
    {isChatLoading && (
      <div className="space-y-3">
        <div className="h-16 rounded-2xl bg-secondary/60 dark:bg-secondary-dark/80" />
        <div className="ml-auto h-14 w-3/4 rounded-2xl bg-primary/20" />
      </div>
    )}

    {isChatError && (
      <div className="rounded-2xl border border-error/30 bg-error/10 p-4 text-sm text-error">
        {getErrorMessage(chatError)}
      </div>
    )}

    {!isChatLoading && !isChatError && visibleMessages.length === 0 && (
      <EmptyState />
    )}

    {visibleMessages.map((message, index) => (
      <MessageBubble
        key={` ${message.role} - ${message.createdAt} || index} - ${index} `}
        message={message}
      />
    ))}

    {sendMutation.isPending && (
      <div className="flex justify-start">
        <div className="rounded-2xl rounded-bl-md border border-border/25 bg-surface-
elevated px-4 py-3 text-sm text-text-muted shadow-sm dark:border-border-dark/35 dark:bg-surface-
elevated-dark dark:text-text-muted-dark">
          AI-консультант друкує...
        </div>
      </div>
    )}
    <div ref={messagesEndRef} />
  </div>

  <form
    className="border-t border-border/20 bg-surface-elevated/75 p-4 dark:border-border-
dark/30 dark:bg-surface-elevated-dark/80"
    onSubmit={handleSubmit}
  >
    <Textarea

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

value={messageInput}
onChange={(event) => setMessageInput(event.target.value)}
onKeyDown={handleKeyDown}
maxLength={MAX_MESSAGE_LENGTH}
placeholder="Напишіть питання про GlowHub або мейкап..."
textareaClassName="max-h-32 min-h-[76px] resize-none text-sm"
disabled={sendMutation.isPending}
/>
<div className="mt-3 flex items-center justify-between gap-3">
  <div className="text-xs text-text-muted dark:text-text-muted-dark">
    {messageInput.length}/{MAX_MESSAGE_LENGTH}
  </div>
  <div className="flex items-center gap-2">
    {visibleMessages.length > 0 && (
      <Button
        variant="ghost"
        size="sm"
        onClick={() => clearMutation.mutate()}
        isLoading={clearMutation.isPending}
        disabled={clearMutation.isPending || sendMutation.isPending}
      >
        ОЧИСТИТИ
      </Button>
    )}
    <Button
      type="submit"
      size="sm"
      disabled={isSubmitDisabled}
      isLoading={sendMutation.isPending}
      rightIcon={
        <svg width="14" height="14" viewBox="0 0 24 24" fill="none" stroke="currentColor"
strokeWidth="2.2" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
          <path d="m22 2-7 20-4-9-9-4Z" />
          <path d="M22 2 11 13" />
        </svg>
      }
    >
      Надіслати
    </Button>
  </div>
</div>
</form>
</section>
)}

<button
  type="button"
  className="group flex h-14 w-14 cursor-pointer items-center justify-center rounded-2xl bg-
gradient-to-br from-primary to-primary-hover text-text-light shadow-glow transition-transform
hover:-translate-y-0.5 focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-primary/45
focus-visible:ring-offset-2 focus-visible:ring-offset-surface dark:focus-visible:ring-offset-surface-
dark"

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

aria-label={isOpen ? "Закрити AI-чат" : "Відкрити AI-чат"}
aria-expanded={isOpen}
onClick={() => setIsOpen((current) => !current)}
>
  {isOpen ? (
    <svg width="22" height="22" viewBox="0 0 24 24" fill="none" stroke="currentColor"
strokeWidth="2.2" strokeLinecap="round" aria-hidden>
      <path d="M18 6 6 18" />
      <path d="m6 6 12 12" />
    </svg>
  ) : (
    <svg width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor"
strokeWidth="2.1" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
      <path d="M21 15a4 4 0 0 1-4 4H8l-5 3V7a4 4 0 0 1 4-4h10a4 4 0 0 1 4 4z" />
      <path d="M8 10h8" />
      <path d="M8 14h5" />
    </svg>
  )}
</button>
</div>
);
};

```

export default AiChatWidget;

ЛІСТИНГ: artit/index.tsx

```

import { useCallback, useState } from "react";
import { useParams, Link } from "react-router-dom";
import { useForm } from "react-hook-form";
import { z } from "zod";
import { zodResolver } from "@hookform/resolvers/zod";
import MainLayout from "../../layouts/main/MainLayout";
import AnimateOnView from "../../components/AnimateOnView";
import Seo from "../../components/Seo";
import Loader from "../../components/ui/Loader";
import Popup from "../../components/ui/Popup";
import Button from "../../components/ui/Button";
import TextField from "../../components/ui/TextField";
import BookingForm from "../../components/BookingForm";
import PostCard from "../../components/PostCard";
import { useGetApiArtistById, usePutApiArtistsMe, getGetApiArtistByIdQueryKey } from
"../../api/artists/artists";
import { useGetApiGallery, usePostApiGalleryUpload, useDeleteApiGalleryId } from
"../../api/gallery/gallery";
import { useGetApiPosts } from "../../api/posts/posts";
import { getGetApiUserMeQueryKey, useGetApiUserMe, usePutApiUserMeAvatar } from
"../../api/users/users";
import {
  getGetApiSubscriptionsArtistIdMeQueryKey,
  useGetApiSubscriptionsMe,
  useDeleteApiSubscriptionsArtistId,

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    useGetApiSubscriptionsArtistIdMe,
    usePostApiSubscriptionsArtistId,
  } from "../api/subscriptions/subscriptions";
import { useAuthStore } from "../store/auth";
import { useLogout } from "../hooks/useLogout";
import { useQueryClient } from "@tanstack/react-query";
import toast from "react-hot-toast";
import { UserRole, type SubscriptionArtist, type User } from "../api/blogAPI.schemas";

const profileSchema = z.object({
  bio: z.string().optional(),
  specializations: z.string().optional(),
  instagram: z.string().url({ message: "Введіть коректне посилання" }).optional().or(z.literal("")),
  facebook: z.string().url({ message: "Введіть коректне посилання" }).optional().or(z.literal("")),
  tiktok: z.string().url({ message: "Введіть коректне посилання" }).optional().or(z.literal("")),
});

type ProfileFormValues = z.infer<typeof profileSchema>;
type ArtistProfilePageProps = {
  mode?: "public" | "me";
};

const getSubscriptionArtist = (artist: SubscriptionArtist | undefined): User | null =>
  artist && typeof artist === "object" ? artist : null;

const PlusIcon = () => (
  <svg width="22" height="22" viewBox="0 0 24 24" fill="none" stroke="currentColor"
  strokeWidth="1.8" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
    <line x1="12" y1="5" x2="12" y2="19" />
    <line x1="5" y1="12" x2="19" y2="12" />
  </svg>
);

const ChevronLeftIcon = () => (
  <svg width="22" height="22" viewBox="0 0 24 24" fill="none" stroke="currentColor"
  strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
    <path d="m15 18-6-6 6-6" />
  </svg>
);

const ChevronRightIcon = () => (
  <svg width="22" height="22" viewBox="0 0 24 24" fill="none" stroke="currentColor"
  strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
    <path d="m9 18 6-6 6-6" />
  </svg>
);

const TrashIcon = () => (
  <svg width="16" height="16" viewBox="0 0 24 24" fill="none" stroke="currentColor"
  strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
    <polyline points="3 6 5 6 21 6" />
    <path d="M19 6l-1 14H6L5 6" />
  </svg>
);

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						108
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <path d="M10 11v6M14 11v6" />
    <path d="M9 6V4h6v2" />
  </svg>
);

const InstagramIcon = () => (
  <svg width="18" height="18" viewBox="0 0 24 24" fill="none" stroke="currentColor"
    strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" aria-hidden>
    <rect x="2" y="2" width="20" height="20" rx="5" ry="5" />
    <path d="M16 11.37A4 4 0 1 1 12.63 8 4 4 0 0 1 16 11.37z" />
    <line x1="17.5" y1="6.5" x2="17.51" y2="6.5" />
  </svg>
);

const ArtistProfilePage = ({ mode = "public" }: ArtistProfilePageProps) => {
  const { id } = useParams<{ id: string }>();
  const queryClient = useQueryClient();
  const { accessToken, guestMode } = useAuthStore();
  const logout = useLogout();
  const isMePage = mode === "me";

  const [popupIndex, setPopupIndex] = useState<number | null>(null);
  const [editOpen, setEditOpen] = useState(false);
  const [avatarFile, setAvatarFile] = useState<File | null>(null);
  const [avatarPreview, setAvatarPreview] = useState<string | null>(null);

  const { data: userData, isLoading: meLoading } = useGetApiUserMe({
    query: { refetchOnWindowFocus: false, enabled: !!accessToken && !guestMode },
  });
  const me = userData?.data;

  const { data: artistData, isLoading: artistLoading } = useGetApiArtistById(id || "", {
    query: { enabled: !isMePage && !!id },
  });
  const artist = isMePage ? me : artistData?.data;
  const artistId = artist?.id;
  const isOwner = isMePage;
  const isArtist = artist?.role === "writer";
  const isReaderAccount = isMePage && me?.role === UserRole.reader;
  const canManageSubscription =
    !!artistId && !isOwner && isArtist && me?.role === UserRole.reader;

  const { data: galleryData, refetch: refetchGallery } = useGetApiGallery(
    artistId ? { author: artistId } : undefined,
    { query: { enabled: !!artistId && isArtist } }
  );
  const images = galleryData?.data || [];

  const { data: postsData } = useGetApiPosts(
    { authorId: artistId, limit: 3, sort: "date-desc" },
    { query: { enabled: !!artistId && isArtist } }
  );
};

```

					ІІІ3.КР.Б – 121 – 26 – ІІ3	Арк.
						109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const posts = postData?.data?.posts || [];

const { data: mySubscriptionsData, isLoading: isSubscriptionsLoading } =
  useGetApiSubscriptionsMe({
    query: {
      enabled: isReaderAccount,
      refetchOnWindowFocus: false,
    },
  });
const mySubscriptions = mySubscriptionsData?.data || [];

const { data: subscriptionStatus, isLoading: isSubscriptionLoading } =
  useGetApiSubscriptionsArtistIdMe(artistId || "", {
    query: {
      enabled: canManageSubscription,
      refetchOnWindowFocus: false,
    },
  });
const isSubscribed = !!subscriptionStatus?.data?.subscribed;

const invalidateSubscriptionStatus = () => {
  if (!artistId) return;

  queryClient.invalidateQueries({
    queryKey: getGetApiSubscriptionsArtistIdMeQueryKey(artistId),
  });
};

const { mutate: subscribe, isPending: isSubscribing } =
  usePostApiSubscriptionsArtistId({
    mutation: {
      onSuccess: () => {
        toast.success("Ви підписалися на оновлення");
        invalidateSubscriptionStatus();
      },
      onError: () => toast.error("Не вдалося підписатися"),
    },
  });

const { mutate: unsubscribe, isPending: isUnsubscribing } =
  useDeleteApiSubscriptionsArtistId({
    mutation: {
      onSuccess: () => {
        toast.success("Підписку скасовано");
        invalidateSubscriptionStatus();
      },
      onError: () => toast.error("Не вдалося скасувати підписку"),
    },
  });

const { mutate: uploadImage, isPending: isUploading } = usePostApiGalleryUpload({
  mutation: {

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

onSuccess: () => {
  toast.success("Фото успішно завантажено");
  refetchGallery();
},
onError: () => toast.error("Помилка при завантаженні фото"),
},
});

const { mutate: deleteImage } = useDeleteApiGalleryId({
  mutation: {
    onSuccess: () => {
      toast.success("Фото видалено");
      refetchGallery();
    },
    onError: () => toast.error("Помилка при видаленні фото"),
  },
});

const { mutate: updateProfile, isPending: isUpdating } = usePutApiArtistsMe({
  mutation: {
    onSuccess: () => {
      toast.success("Профіль оновлено");
      setEditOpen(false);
      setAvatarFile(null);
      setAvatarPreview(null);
      queryClient.invalidateQueries({ queryKey: getGetApiUserMeQueryKey() });
      if (artistId) {
        queryClient.invalidateQueries({ queryKey: getGetApiArtistByIdQueryKey(artistId) });
      }
    },
    onError: () => toast.error("Помилка при оновленні профілю"),
  },
});

const { mutate: updateMyAvatar, isPending: isUpdatingAvatar } =
  usePutApiUserMeAvatar({
    mutation: {
      onSuccess: () => {
        toast.success("Аватар оновлено");
        setAvatarPreview(null);
        queryClient.invalidateQueries({ queryKey: getGetApiUserMeQueryKey() });
      },
      onError: () => {
        toast.error("Не вдалося оновити аватар");
        setAvatarPreview(null);
      },
    },
  });

const {
  register,
  handleSubmit,

```

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

```

    formState: { errors },
    reset,
  } = useForm<ProfileFormValues>({
    resolver: zodResolver(profileSchema),
    defaultValues: {
      bio: artist?.bio || "",
      specializations: artist?.specializations?.join(", ") || "",
      instagram: artist?.socialLinks?.instagram || "",
      facebook: artist?.socialLinks?.facebook || "",
      tiktok: artist?.socialLinks?.tiktok || "",
    },
  });

const handleEditOpen = () => {
  reset({
    bio: artist?.bio || "",
    specializations: artist?.specializations?.join(", ") || "",
    instagram: artist?.socialLinks?.instagram || "",
    facebook: artist?.socialLinks?.facebook || "",
    tiktok: artist?.socialLinks?.tiktok || "",
  });
  setEditOpen(true);
};

const onProfileSubmit = (data: ProfileFormValues) => {
  const specializations = data.specializations
    ? data.specializations.split(",").map((s) => s.trim()).filter(Boolean)
    : [];
  updateProfile({
    data: {
      bio: data.bio,
      specializations,
      socialLinks: {
        instagram: data.instagram || undefined,
        facebook: data.facebook || undefined,
        tiktok: data.tiktok || undefined,
      },
      avatarFile: avatarFile || undefined,
    },
  });
};

const handleAvatarChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  const file = e.target.files?.[0];
  if (file) {
    setAvatarFile(file);
    setAvatarPreview(URL.createObjectURL(file));
  }
};

const handleReaderAvatarChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  const file = e.target.files?.[0];

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (!file) return;

    setAvatarPreview(URL.createObjectURL(file));
    updateMyAvatar({ data: { avatar: file } });
    e.target.value = "";
  };

const handleUploadImage = (e: React.ChangeEvent<HTMLInputElement>) => {
  const file = e.target.files?.[0];
  if (file) uploadImage({ data: { image: file } });
};

const handleSubscriptionToggle = () => {
  if (!artistId) return;

  if (isSubscribed) {
    unsubscribe({ artistId });
  } else {
    subscribe({ artistId });
  }
};

const handlePrev = useCallback(() => {
  setPopupIndex((prev) =>
    prev !== null ? (prev - 1 + images.length) % images.length : null
  );
}, [images.length]);

const handleNext = useCallback(() => {
  setPopupIndex((prev) =>
    prev !== null ? (prev + 1) % images.length : null
  );
}, [images.length]);

if (isMePage ? meLoading : artistLoading) {
  return (
    <MainLayout>
      <div className="flex min-h-[50vh] items-center justify-center">
        <Loader size={40} />
      </div>
    </MainLayout>
  );
}

if (!artist) {
  return (
    <MainLayout>
      <div className="flex min-h-[50vh] flex-col items-center justify-center gap-4 text-center">
        <h1 className="font-display text-3xl font-semibold">Візажиста не знайдено</h1>
        <Link to="/artists">
          <Button variant="outline">До списку візажистів</Button>
        </Link>
      </div>
    </MainLayout>
  );
}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						113
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </div>
  </MainLayout>
);
}

const popupImg = popupIndex !== null ? images[popupIndex]?.url : null;
const profileAvatar = avatarPreview || artist.avatar;

return (
  <MainLayout>
    <Seo
      title={` ${artist.username} | GlowHub` }
      description={
        artist.bio ||
        (isArtist
          ? `Профіль візажиста ${artist.username} на платформі GlowHub. Портфоліо та запис на консультацію.`
          : `Профіль користувача ${artist.username} на платформі GlowHub.`)
        }
      image={artist.avatar}
      url={typeof window !== "undefined" ? window.location.href : undefined}
    />

    <section className="relative left-1/2 w-screen max-w-[100vw] -translate-x-1/2 overflow-hidden border-b border-border/20 bg-surface py-12 dark:border-border-dark/25 dark:bg-surface-dark sm:py-16">
      <div className="page-hero-gradient pointer-events-none absolute inset-0 opacity-60" />
      <div aria-hidden className="pointer-events-none absolute -left-14 top-1/3 h-64 w-64 rounded-full bg-primary-light/25 blur-3xl dark:bg-primary/15" style={{ animation: "blob 18s ease-in-out infinite" }} />
      <div className="relative z-[1] mx-auto max-w-6xl px-4 sm:px-6 lg:px-8">
        <div className="flex flex-col items-start gap-8 sm:flex-row sm:items-center">
          <AnimateOnView delay={0} variant="scale">
            <div className="relative">
              {profileAvatar ? (
                <img
                  src={profileAvatar}
                  alt={artist.username}
                  className="h-28 w-28 rounded-full border-4 border-primary/30 object-cover shadow-xl dark:border-primary-light/30 sm:h-36 sm:w-36"
                />
              ) : (
                <div className="flex h-28 w-28 items-center justify-center rounded-full border-4 border-primary/30 bg-gradient-to-br from-primary to-primary-hover shadow-xl dark:border-primary-light/30 sm:h-36 sm:w-36">
                  <span className="text-4xl font-semibold text-white sm:text-5xl">
                    {artist.username?.slice(0, 1).toUpperCase()}
                  </span>
                </div>
              )}
            </div>
          </AnimateOnView>
        </div>
      </div>
    </section>
  )
);

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="flex-1">
  <AnimateOnView delay={80}>
    <div className="mb-1 flex flex-wrap items-center gap-3">
      <h1 className="font-display text-3xl font-semibold text-text dark:text-text-light
sm:text-4xl">
        {artist.username}
      </h1>
      {isArtist && (
        <span className="rounded-full border border-primary/25 bg-primary-light/25 px-3
py-1 text-xs font-semibold uppercase tracking-widest text-primary dark:border-primary/35 dark:bg-
primary/15 dark:text-primary-light">
          Візажист
        </span>
      )}
    </div>
  </AnimateOnView>

  {isArtist && artist.specializations && artist.specializations.length > 0 && (
    <AnimateOnView delay={120}>
      <div className="mb-3 flex flex-wrap gap-2">
        {artist.specializations.map((spec) => (
          <span
            key={spec}
            className="rounded-full border border-border/30 bg-surface/80 px-3 py-1 text-xs
font-medium text-text-muted dark:border-border-dark/40 dark:bg-surface-dark/80 dark:text-text-
muted-dark"
          >
            {spec}
          </span>
        ))}
      </div>
    </AnimateOnView>
  )}

  {artist.bio && (
    <AnimateOnView delay={160}>
      <p className="mb-4 max-w-2xl leading-relaxed text-text-muted dark:text-text-muted-
dark">
        {artist.bio}
      </p>
    </AnimateOnView>
  )}

  <AnimateOnView delay={200}>
    <div className="flex flex-wrap items-center gap-3">
      {artist.socialLinks?.instagram && (
        <a
          href={artist.socialLinks.instagram}
          target="_blank"
          rel="noopener noreferrer"
          className="flex items-center gap-2 rounded-full border border-border/30 bg-

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						115
Змн.	Арк.	№ докум.	Підпис	Дата		

```

surface/60 px-3 py-1.5 text-sm text-text-muted transition-all hover:border-primary/40 hover:text-
primary dark:border-border-dark/40 dark:bg-surface-dark/60 dark:text-text-muted-dark
dark: hover: text-primary-light"
    >
    <InstagramIcon />
    Instagram
  </a>
)}
{isOwner && isArtist && (
  <
    <Button variant="outline" size="sm" onClick={handleEditOpen}>
      Редагувати профіль
    </Button>
    <Link to="/post/create">
      <Button variant="primary" size="sm">
        Написати статтю
      </Button>
    </Link>
  </>
)}
{isReaderAccount && (
  <label className="cursor-pointer rounded-full border border-border/30 bg-surface/60
px-3 py-1.5 text-sm text-text-muted transition-all hover:border-primary/40 hover:text-primary
dark:border-border-dark/40 dark:bg-surface-dark/60 dark:text-text-muted-dark dark: hover: text-
primary-light">
    {isUpdatingAvatar ? "Оновлення..." : "Оновити аватар"}
    <input
      type="file"
      accept="image/*"
      className="sr-only"
      disabled={isUpdatingAvatar}
      onChange={handleReaderAvatarChange}
    />
  </label>
)}
{isOwner && (
  <Button
    variant="outline"
    size="sm"
    onClick={() => logout.mutate()}
    disabled={logout.status === "pending"}
    isLoading={logout.status === "pending"}
  >
    Вийти
  </Button>
)}
{canManageSubscription && (
  <div className="flex flex-col gap-1">
    <Button
      type="button"
      variant={isSubscribed ? "outline" : "primary"}
      size="sm"

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						116
Змн.	Арк.	№ докум.	Підпис	Дата		


```

</AnimateOnView>
): (
<div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-3">
  {mySubscriptions.map((subscription, index) => {
    const subscribedArtist = getSubscriptionArtist(subscription.artist);

    if (!subscribedArtist?.id) return null;

    return (
      <AnimateOnView
        key={subscription.id || subscribedArtist.id}
        delay={(index % 3) * 70}
      >
        <Link
          to={`'/artist/${subscribedArtist.id}`}
          className="flex h-full items-center gap-4 rounded-2xl border border-border/30 bg-
surface p-4 shadow-sm transition-all hover:-translate-y-0.5 hover:border-primary/35 hover:shadow-
md dark:border-border-dark/40 dark:bg-surface-dark"
        >
          {subscribedArtist.avatar ? (
            <img
              src={subscribedArtist.avatar}
              alt={subscribedArtist.username}
              className="h-14 w-14 shrink-0 rounded-full object-cover"
            />
          ) : (
            <span className="flex h-14 w-14 shrink-0 items-center justify-center rounded-full
bg-primary/15 text-lg font-semibold uppercase text-primary dark:bg-primary/25 dark:text-primary-
light">
              {subscribedArtist.username?.slice(0, 1)}
            </span>
          )}
          <div className="min-w-0">
            <h3 className="truncate font-semibold text-text dark:text-text-light">
              {subscribedArtist.username}
            </h3>
            <p className="mt-1 line-clamp-2 text-sm text-text-muted dark:text-text-muted-
dark">
              {subscribedArtist.bio || "Переглянути профіль автора"}
            </p>
          </div>
        </Link>
      </AnimateOnView>
    );
  })}
</div>
)}
</section>
)}

{isArtist && (
  <div className={`mb-12 grid gap-10 ${isOwner ? "" : "lg:grid-cols-[1fr 360px]"}`} >

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<section>
  <AnimateOnView delay={0}>
    <div className="mb-6 flex items-center justify-between">
      <h2 className="font-display text-2xl font-semibold text-text dark:text-text-light">
        Портфоліо
      </h2>
    </div>
  </AnimateOnView>

  {images.length === 0 && !isOwner ? (
    <AnimateOnView>
      <div className="flex flex-col items-center gap-3 rounded-2xl border border-dashed
border-border/40 py-14 text-center dark:border-border-dark/50">
        <p className="text-text-muted dark:text-text-muted-dark">
          Поки немає робіт у портфоліо
        </p>
      </div>
    </AnimateOnView>
  ) : (
    <div className="grid grid-cols-2 gap-4 sm:grid-cols-3">
      {images.map((image, idx) => (
        <AnimateOnView key={image.id} delay={{idx % 3} * 70} variant="scale">
          <div className="group relative overflow-hidden rounded-2xl border border-
border/25 bg-surface shadow-sm dark:border-border-dark/35 dark:bg-surface-dark"
            style={{ aspectRatio: idx % 5 === 0 ? "1/1.2" : idx % 7 === 0 ? "1/0.85" : "1/1" }}
          >
            <button
              type="button"
              className="absolute inset-0 cursor-zoom-in"
              onClick={() => setPopupIndex(idx)}
              aria-label={`Переглянути фото ${idx + 1}`}
            >
              <img
                src={image.url}
                alt={image.caption || `Робота ${idx + 1}`}
                className="h-full w-full object-cover transition-transform duration-700 ease-out
group-hover:scale-110"
                loading="lazy"
              />
              <div className="absolute inset-0 bg-gradient-to-t from-black/50 via-transparent
to-transparent opacity-0 transition-opacity duration-300 group-hover:opacity-100" />
              <div className="absolute inset-0 flex items-center justify-center opacity-0
transition-opacity duration-300 group-hover:opacity-100">
                <span className="rounded-full border border-white/25 bg-black/40 px-4 py-2
text-sm font-medium text-white backdrop-blur-sm">
                  Переглянути
                </span>
              </div>
            </div>
          </button>
          {isOwner && (
            <button
              type="button"

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        onClick={() => image.id && deleteImage({ id: image.id })}
        className="absolute right-2 top-2 z-10 flex h-8 w-8 items-center justify-center
rounded-full border border-white/20 bg-black/50 text-white opacity-0 transition-all group-
hover:opacity-100 hover:bg-red-600/80 focus-visible:opacity-100 focus-visible:outline-none"
        aria-label="Видалити фото"
      >
        <TrashIcon />
      </button>
    )}
  </div>
</AnimateOnView>
)}}

{isOwner && (
  <AnimateOnView delay={{(images.length % 3) * 70} variant="scale">
    <label
      className="group relative flex cursor-pointer flex-col items-center justify-center
gap-2 rounded-2xl border-2 border-dashed border-border/45 text-center text-text-muted transition-
all duration-300 hover:border-primary/70 hover:bg-primary-light/20 hover:text-primary
dark:border-border-dark/50 dark:text-text-muted-dark dark:hover:border-primary-light
dark:hover:bg-primary/15 dark:hover:text-primary-light"
      style={{ aspectRatio: "1/1" }}
    >
      {isUploading ? (
        <Loader />
      ) : (
        <span className="flex h-12 w-12 items-center justify-center rounded-full bg-
primary-light/30 text-primary transition-transform duration-300 group-hover:-rotate-6 group-
hover:scale-110 dark:bg-primary/20 dark:text-primary-light">
          <PlusIcon />
        </span>
        <span className="text-sm font-medium">Додати фото</span>
      </>
    )}
    <input
      type="file"
      accept="image/*"
      className="sr-only"
      onChange={handleUploadImage}
      aria-label="Завантажити нове фото"
    />
  </label>
</AnimateOnView>
)}}
</div>
)}}
</section>

{!isOwner && (
  <aside className="lg:sticky lg:top-24 lg:self-start">
    <AnimateOnView delay={100}>

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="rounded-3xl border border-border/25 bg-surface p-6 shadow-sm ring-
1 ring-black/5 dark:border-border-dark/35 dark:bg-surface-dark dark:ring-white/5">
  <h2 className="mb-1 font-display text-xl font-semibold text-text dark:text-text-
light">
    Записатися на консультацію
  </h2>
  <p className="mb-5 text-sm text-text-muted dark:text-text-muted-dark">
    Залиште свої контакти і {artist.username} зв'яжеться з вами найближчим часом.
  </p>
  <BookingForm
    artistId={artist.id || ""}
    artistName={artist.username || ""}
  />
</div>
</AnimateOnView>
</aside>
)}
</div>
)}

{posts.length > 0 && (
  <section>
    <AnimateOnView delay={0}>
      <div className="mb-6 flex items-center justify-between">
        <h2 className="font-display text-2xl font-semibold text-text dark:text-text-light">
          Останні статті
        </h2>
        <Link
          to={` /blog?authorId=${artistId}`}
          className="text-sm font-medium text-primary transition-colors hover:text-primary-
hover dark:text-primary-light"
        >
          Всі статті →
        </Link>
      </div>
    </AnimateOnView>
    <div className="grid gap-5 sm:grid-cols-2 lg:grid-cols-3">
      {posts.map((post, idx) => (
        <AnimateOnView key={post.id} delay={idx * 80}>
          <PostCard
            title={post.title || "Без назви"}
            excerpt={post.excerpt || "Опис відсутній"}
            author={post.author?.username || "Автор невідомий"}
            href={` /post/${post.slug}`}
            imageUrl={post.imageUrl}
            date={post.date ? new Date(post.date).toLocaleDateString("uk-UA") : ""}
          />
        </AnimateOnView>
      ))}
    </div>
  </section>
)}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</div>

<Popup open={popupIndex !== null} onClose={() => setPopupIndex(null)} wide>
  {popupImg && (
    <div className="flex w-full min-w-0 flex-col items-stretch gap-3 pt-2 md:flex-row
md:items-center md:justify-center md:gap-2 md:pt-0">
      <button
        type="button"
        onClick={handlePrev}
        className="hidden h-10 w-10 shrink-0 cursor-pointer items-center justify-center rounded-
full border border-white/20 bg-black/50 text-white backdrop-blur-sm transition-all hover:scale-105
hover:bg-black/70 focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-primary/70
md:flex"
        aria-label="Попереднє фото"
      >
        <ChevronLeftIcon />
      </button>
      <div className="relative w-full min-w-0 max-w-full flex-1">
        <img
          src={popupImg}
          alt="Велике фото"
          className="mx-auto max-h-[min(78dvh,85svh)] w-full object-contain rounded-2xl
shadow-2xl md:max-h-[80vh]"
        />
        {popupIndex !== null && (
          <div className="absolute bottom-3 left-1/2 hidden -translate-x-1/2 rounded-full border
border-white/20 bg-black/50 px-3 py-1 text-xs text-white backdrop-blur-sm md:block">
            {popupIndex + 1} / {images.length}
          </div>
        )}
      </div>
      <button
        type="button"
        onClick={handleNext}
        className="hidden h-10 w-10 shrink-0 cursor-pointer items-center justify-center rounded-
full border border-white/20 bg-black/50 text-white backdrop-blur-sm transition-all hover:scale-105
hover:bg-black/70 focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-primary/70
md:flex"
        aria-label="Наступне фото"
      >
        <ChevronRightIcon />
      </button>
      {popupIndex !== null && (
        <div className="flex w-full max-w-sm items-center justify-between gap-3 self-center
md:hidden">
          <button
            type="button"
            onClick={handlePrev}
            className="flex h-11 w-11 shrink-0 cursor-pointer items-center justify-center rounded-
full border border-white/20 bg-black/50 text-white backdrop-blur-sm transition active:scale-95"
            aria-label="Попереднє фото"
          >

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <ChevronLeftIcon />
      </button>
      <span className="shrink-0 text-sm text-white/90 tabular-nums">
        {popupIndex + 1} / {images.length}
      </span>
      <button
        type="button"
        onClick={handleNext}
        className="flex h-11 w-11 shrink-0 cursor-pointer items-center justify-center rounded-
full border border-white/20 bg-black/50 text-white backdrop-blur-sm transition active:scale-95"
        aria-label="Наступне фото"
      >
        <ChevronRightIcon />
      </button>
    </div>
  )}
</div>
)}
</Popup>

<Popup open={editOpen} onClose={() => setEditOpen(false)}>
  <div className="w-full max-w-lg rounded-2xl border border-border/25 bg-surface p-6
shadow-xl dark:border-border-dark/35 dark:bg-surface-dark">
    <h2 className="mb-5 font-display text-xl font-semibold text-text dark:text-text-light">
      Редагувати профіль
    </h2>
    <form onSubmit={handleSubmit(onProfileSubmit)} className="flex flex-col gap-4">
      <div>
        <label className="mb-1 block text-sm font-medium text-primary dark:text-primary-
light">
          Аватар
        </label>
        <div className="flex items-center gap-4">
          {(avatarPreview || artist.avatar) && (
            <img
              src={avatarPreview || artist.avatar}
              alt="Аватар"
              className="h-16 w-16 rounded-full object-cover border-2 border-primary/30"
            />
          )}
          <label className="cursor-pointer rounded-xl border border-dashed border-border/50 px-
4 py-2 text-sm text-text-muted transition-all hover:border-primary/50 hover:text-primary
dark:border-border-dark/50 dark:text-text-muted-dark dark:hover:text-primary-light">
            Вибрати фото
          <input
            type="file"
            accept="image/*"
            className="sr-only"
            onChange={handleAvatarChange}
          />
          </label>
        </div>
      </div>
    </form>
  </div>

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</div>

<div>
  <label htmlFor="edit-bio" className="mb-1 block text-sm font-medium text-primary
dark:text-primary-light">
    Про себе
  </label>
  <textarea
    id="edit-bio"
    rows={3}
    placeholder="Розкажіть про себе, ваш досвід та підхід до роботи..."
    className="w-full resize-none rounded-xl border border-border-input bg-bg-input px-4
py-2.5 text-text transition-[border-color,box-shadow] duration-200 placeholder:text-text-muted/60
focus:border-primary focus:outline-none focus:ring-2 focus:ring-primary/35 dark:border-border-
input-dark dark:bg-bg-input-dark dark:text-text-light dark:placeholder:text-text-muted-dark/70"
    {...register("bio")}
  />
</div>

<TextField
  label="Спеціалізації"
  placeholder="Весільний макіяж, Вечірній макіяж, Навчання"
  hint="Введіть через кому"
  error={errors.specializations?.message}
  {...register("specializations")}
/>

<TextField
  label="Instagram"
  placeholder="https://instagram.com/yourusername"
  error={errors.instagram?.message}
  {...register("instagram")}
/>

<TextField
  label="Facebook"
  placeholder="https://facebook.com/yourpage"
  error={errors.facebook?.message}
  {...register("facebook")}
/>

<TextField
  label="TikTok"
  placeholder="https://tiktok.com/@yourusername"
  error={errors.tiktok?.message}
  {...register("tiktok")}
/>

<div className="mt-2 flex gap-3">
  <Button type="submit" isLoading={isUpdating} disabled={isUpdating} className="flex-
1 justify-center">
    Зберегти
  </Button>
  <Button type="button" variant="outline" onClick={() => setEditOpen(false)}

```

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						124
Змн.	Арк.	№ докум.	Підпис	Дата		

```

className="flex-1 justify-center">
    Скасувати
  </Button>
</div>
</form>
</div>
</Popup>
</MainLayout>
);
};

```

```
export default ArtistProfilePage;
```

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						125
Змн.	Арк.	№ докум.	Підпис	Дата		