НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення для створення та управління портфоліо фахівця»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми к.т.н., доцент	_____ (підпис)	**Ганна ВАЙГАНГ**
Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Максим НЕДЬОШЕВ**
Керівник бакалаврської кваліфікаційної роботи д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Виконав	_____ (підпис)	**Дмитрій КУЗЬМЕНКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Кузьменко Дмитрію Олеговичу
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення для створення та управління портфоліо фахівця»
затверджена наказом від «19» грудня 2025 р. № 3196 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи:

Перелік питань, що підлягають дослідженню:
Аналіз предметної області та постановка завдання, проектування інформаційного забезпечення, розробка та впровадження програмного забезпечення

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Максим НЕДЬОШЕВ**

Консультант бакалаврської кваліфікаційної роботи _____ (підпис) **Роман РУДЕНСЬКИЙ**
д.е.н., професор

Завдання взяв до виконання _____ (підпис) **Дмитрій КУЗЬМЕНКО**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

ІШІ – Інструменти штучного інтелекту

ПЗ – Програмне забезпечення

СУБД – Система управління базами даних

API (Application Programming Interface) – Інтерфейс прикладного програмування

BaaS (Backend-as-a-Service) – Бекенд як послуга

BSON (Binary JSON) – Бінарний формат серіалізації даних

CDN (Content Delivery Network) – Мережа доставки контенту

CLI (Command Line Interface) – Інтерфейс командного рядка

CRUD (Create, Read, Update, Delete) – Базові функції управління даними

DOM (Document Object Model) – Об'єктна модель документа

ER-діаграма (Entity-Relationship) – Діаграма «сутність-зв'язок»

FK (Foreign Key) – Зовнішній ключ

FPS (Frames Per Second) – Кількість кадрів на секунду

GPU (Graphics Processing Unit) – Графічний процесор

HMR (Hot Module Replacement) – Механізм гарячої заміни модулів

HTTPS (HyperText Transfer Protocol Secure) – Захищений протокол передачі гіпертексту

LTS (Long Term Support) – Версія програмного забезпечення з довгостроковою підтримкою

NoSQL (Not Only SQL) – Нереляційний підхід до управління базами даних

PBR (Physically Based Rendering) – Фізично коректний рендеринг

PK (Primary Key) – Первинний ключ

SPA (Single Page Application) – Односторінковий вебдодаток

SSL (Secure Sockets Layer) – Криптографічний протокол безпеки зв'язку

UI/UX (User Interface / User Experience) – Інтерфейс користувача / Досвід користувача

UML (Unified Modeling Language) – Уніфікована мова моделювання

WebGL (Web Graphics Library) – Програмна бібліотека для рендерингу 3D-графіки у браузері

WSS (WebSocket Secure) – Безпечний протокол передачі даних у реальному часі

ВСТУП

Актуальність теми. Стрімкий розвиток цифрових технологій, креативних індустрій, індустрії розробки ігор (GameDev) та сфери тривимірної комп'ютерної графіки ставить перед сучасними спеціалістами нові, значно складніші виклики у питанні професійної самопрезентації на глобальному ринку праці. Зростання конкуренції вимагає від 3D-художників, аніматорів, концепт-артистів та UI/UX дизайнерів наявності не просто статичного текстового резюме, а високотехнологічного інтерактивного цифрового портфоліо. Особливо гостро ця проблема проявляється у сфері тривимірного моделювання, де класичні статичні зображення або відеодемонстрації не здатні повною мірою передати реальну якість полігональної сітки, особливості топології, налаштування PBR-матеріалів та деталізацію текстур. Це суттєво ускладнене процесом об'єктивної технічної оцінки кандидатів з боку рекрутерів та артдиректорів, змушуючи їх витрачати час на завантаження важких вихідних файлів та їх відкриття у десктопному програмному забезпеченні. Існуючі загальні професійні соціальні мережі не підтримують апаратний вебрендеринг тривимірної графіки, а спеціалізовані 3D-галереї позбавлені вбудованого інструментарію для прямої комунікації та професійного нетворкінгу в режимі реального часу, що призводить до фрагментації процесів найму.

Метою розробки програмного додатку є створення спеціалізованої інтерактивної вебплатформи для публікації та управління портфоліо фахівців креативних індустрій, яка на архітектурному рівні об'єднує передові технології апаратного рендерингу тривимірного контенту безпосередньо у вікні браузера з інструментами прямої real-time комунікації між роботодавцями та кандидатами. Актуальність досягнення цієї мети полягає у забезпеченні ринку праці єдиним оптимізованим середовищем, яке автоматизує технічний аудит цифрових робіт фахівців, підвищує ефективність рекрутингу у сфері комп'ютерної графіки,

скорочує час на прийняття рішень про найм та забезпечує надійний захист інтелектуальної власності авторів.

Методи та технології розробки. Методологічною основою роботи є принципи об’єктно-орієнтованого проєктування, концепція побудови реактивних інтерфейсів та безсерверний архітектурний підхід Backend-as-a-Service (BaaS). Для реалізації клієнтської частини прикладного програмного забезпечення (Single Page Application) використано прогресивний фреймворк Vue.js 3 з підходом Composition API та сучасний інструмент модульної збірки Vite. Інтерактивний 3D-в'ювер та діагностичний модуль інспектування матеріалів розроблено на базі високорівневої JavaScript-бібліотеки Three.js, що працює поверх апаратного кросбраузерного інтерфейсу WebGL. Інфраструктура серверної частини, автентифікація користувачів, нереляційне сховище даних реального часу та збереження мультимедійних файлів реалізовані за допомогою хмарної екосистеми Google Firebase (сервіси Firebase Authentication, Cloud Firestore та Cloud Storage).

Апробація результатів розробки. Основні теоретичні положення, архітектурні рішення та практичні результати розробки програмного додатку були представлені, обговорені та отримали позитивну оцінку на Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених «Інформаційні технології та комп'ютерна інженерія». Матеріали та тези доповіді, присвячені принципам побудови вебсистем інтерактивного моніторингу та інспектування тривимірного контенту, опубліковані у збірнику наукових праць конференції.

Структура записки у логічній послідовності відповідно до етапів інженерного життєвого циклу розробки програмного забезпечення:

- У першому розділі виконано системний аналіз предметної області цифрового рекрутингу в креативних індустріях. Проведено детальний порівняльний огляд функціональних та архітектурних особливостей платформ-аналогів. За допомогою інструментарію UML (діаграми прецедентів,

послідовності та активності) здійснено бізнес-моделювання процесів взаємодії та сформовано комплекс функціональних, нефункціональних та технічних вимог до системи.

- У другому розділі здійснено проєктування інформаційного забезпечення платформи. Розроблено концептуальну та логічну ER-модель даних у нотації Crow’s Foot. Наведено інженерне обґрунтування використання документо-орієнтованої NoSQL СУБД Firebase Cloud Firestore, детально описано фізичну структуру хмарних колекцій і документів, а також розроблено декларативні правила безпеки Firestore Security Rules для захисту даних користувачів.
- У третьому розділі представлено архітектурне та об'єктно-орієнтоване проєктування прикладного програмного забезпечення. За допомогою UML-діаграм класів, пакетів та компонентів деталізовано внутрішню структуру SPA-додатка, описано застосування патернів програмування (зокрема поліморфізму для різних типів контенту) та наведено алгоритми роботи графічного рушія Three.js.
- У четвертому розділі висвітлено питання практичного впровадження, розгортання та експлуатації вебплатформи. Описано конфігурування операційного оточення Node.js та процес деплою статичного бандлу на хмарний хостинг Firebase Hosting із залученням CDN. Наведено опис рольового адміністрування та представлено детальну візуальну демонстрацію роботи графічного інтерфейсу системи, модуля 3D-інспектора, 2D-галереї, відеоплеєра та real-time месенджера.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз процесу створення та управління портфоліо фахівця як предметної області

В умовах сучасної діджиталізації та зростаючої конкуренції на ринку праці, наявність якісного цифрового портфоліо є критичною вимогою для фахівців креативних індустрій. Процес створення портфоліо включає систематизацію виконаних проєктів, їх візуальну презентацію та надання технічної інформації про навички автора. Еволюція вебтехнологій призвела до переходу від статичних форматів презентації (PDF-документи, набори JPEG-зображень) до інтерактивних вебрішень, що вимагає від спеціалізованих платформ підтримки складних мультимедійних форматів.

Для спеціалістів у сфері комп'ютерної графіки (3D-моделерів, концепт-артистів, UI/UX дизайнерів) критично важливим є не просто статичне зображення результату (рендер), а можливість повноцінно продемонструвати об'ємну структуру моделі (mesh), топологію сітки, текстурні карти високої роздільної здатності та анімацію. Управління таким портфоліо передбачає постійне оновлення контенту, взаємодію з аудиторією через коментарі та систему оцінок, а також можливість оперативного зв'язку з потенційними замовниками або рекрутерами. З іншого боку, рекрутери та артдиректори потребують інструментів, які дозволяють швидко та об'єктивно оцінити технічний рівень кандидата без необхідності завантажувати важкі файли та відкривати їх у спеціалізованому програмному забезпеченні (наприклад, Blender, Maya чи ZBrush).

Аналіз сучасних наукових публікацій щодо проблем цифрового рекрутингу [1], а також дослідження ринку існуючих інформаційних платформ для

креативних індустрій [2, 3, 4] дозволяють виділити ряд суттєвих недоліків у наявних інструментах. Зокрема, спираючись на огляд спеціалізованих джерел та функціональних можливостей сучасних вебсервісів, можна стверджувати, що основними проблемами у цій предметній області є:

- складність інтеграції інтерактивного 3D-контенту у стандартні вебплатформи, що не спеціалізуються на комп'ютерній графіці, через відсутність нативної підтримки WebGL-контексту;
- відсутність інструментів прямого нетворкінгу та професійної комунікації в реальному часі всередині спеціалізованих галерейних сервісів, що змушує користувачів переходити в інші месенджери;
- необхідність використання декількох несумісних ресурсів для демонстрації різних типів робіт (окремо для відео, 2D-арту та 3D-моделей), що суттєво ускладнює роботу рекрутерам під час комплексної оцінки навичок кандидата;
- проблеми із захистом інтелектуальної власності авторів, оскільки багато платформ не надають гнучких налаштувань для обмеження прямого завантаження вихідних файлів моделей чи зображень.

1.2 Моделювання предметної області

Для формалізації вимог та опису логіки поведінки розроблюваної програмної системи на етапі системного аналізу використано методологію об’єктно-орієнтованого моделювання за допомогою уніфікованої мови моделювання (UML) [5]. Використання UML дозволяє стандартизувати архітектурні рішення, зменшити ризики на етапі програмування та створити єдину термінологічну базу. Для повного розкриття специфіки предметної області побудовано комплекс моделей, що включає діаграми прецедентів, послідовності та активності.

Базовим етапом моделювання є побудова діаграми прецедентів (Use Case Diagram), яка візуалізує взаємодію між зовнішніми сутностями (акторами) та самою платформою, чітко окреслюючи межі системи. В контексті розроблюваної вебплатформи було виділено дві ключові ролі користувачів, що відображають різні бізнес-потреби:

- Фахівець – зареєстрований користувач-творець (3D-художник, аніматор тощо), метою якого є публікація власного портфоліо, просування своїх послуг та управління доступом до контенту.
- Рекрутер – користувач, який представляє інтереси компаній-роботодавців, здійснює пошук талантів, переглядає роботи та проводить їх глибоку технічну оцінку.

Відповідно до розробленої діаграми, актор «Фахівець» реалізує два основні прецеденти. Перший – «Завантажити проєкт», який є базовою функцією для наповнення портфоліо. Ця дія обов'язково включає (через відношення <<include>>) прецедент «Валідація формату файла». Таке архітектурне рішення гарантує дотримання галузевого стандарту: система автоматично перевіряє наявність коректної 3D-моделі (.glb, .gltf) або мультимедійного формату перед ініціалізацією запису до бази даних, що запобігає збереженню битих файлів. Другий прецедент фахівця – «Налаштувати права на скачування», що дозволяє автору гнучко управляти власною інтелектуальною власністю на рівні кожного окремого об'єкта.

Актор «Рекрутер» має базовий прецедент «Переглянути робочий проєкт». Ключовою особливістю розробленої моделі є наявність прецеденту «Скачати вихідні файли», який пов'язаний із базовим переглядом через відношення розширення (<<extend>>). Цей прецедент активується виключно за умови виконання певної точки розширення – наявності відповідного дозволу (авторизаційного токена) від Фахівця.

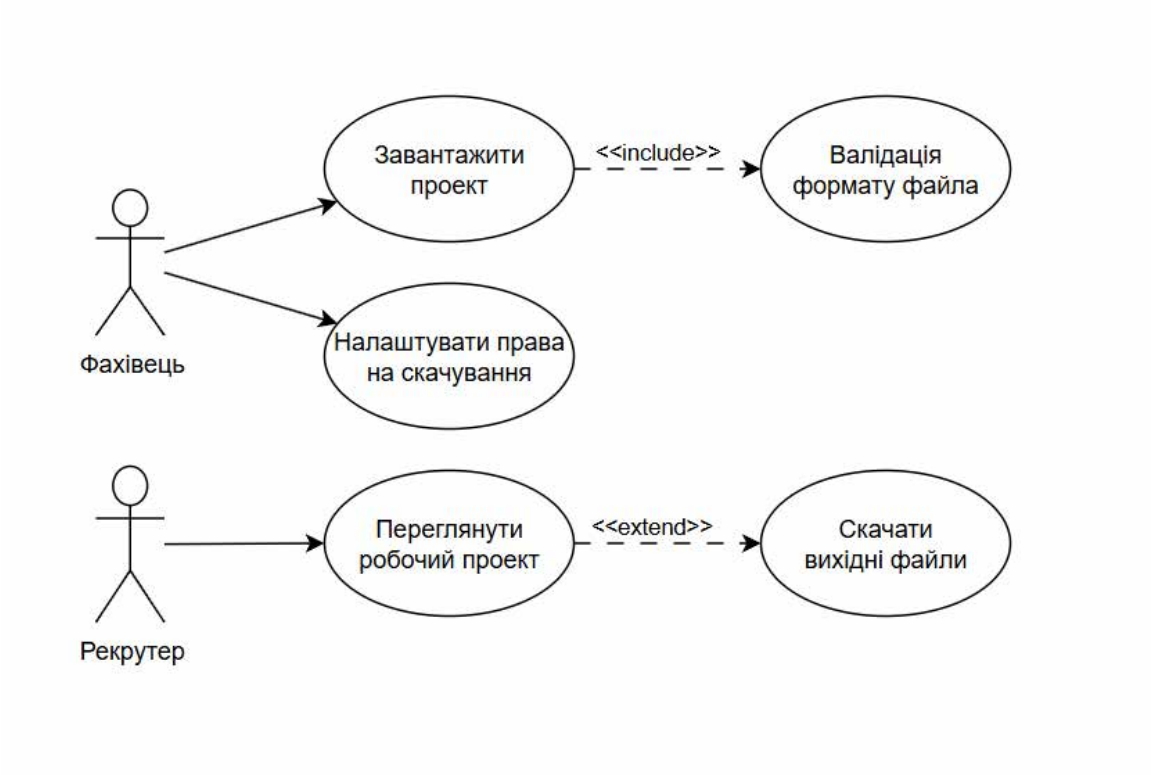

Рис. 1.1 – Діаграма прецедентів взаємодії користувачів із платформою

Для відображення динамічної поведінки об'єктів предметної області у часі побудовано діаграму послідовності (Sequence Diagram). Вона демонструє процес асинхронного обміну даними під час ключового бізнес-сценарію: формування портфоліо та ініціації професійного контакту. Сценарій розпочинається з того, що «Фахівець» створює об'єкт «Портфоліо» у базі даних.

У рамках циклу фахівець багаторазово створює та оновлює проєкти, формуючи базу контенту та завантажуючи бінарні файли у хмарне сховище. Актор «Рекрутер» виконує запити на перегляд портфоліо, отримуючи візуальні дані через клієнтський інтерфейс та аналізуючи навички спеціаліста. Прийнявши рішення, рекрутер генерує подію надсилання офферу через підсистему повідомлень. Фахівець отримує це повідомлення та надає відповідь, замикаючи цикл двосторонньої комунікації.

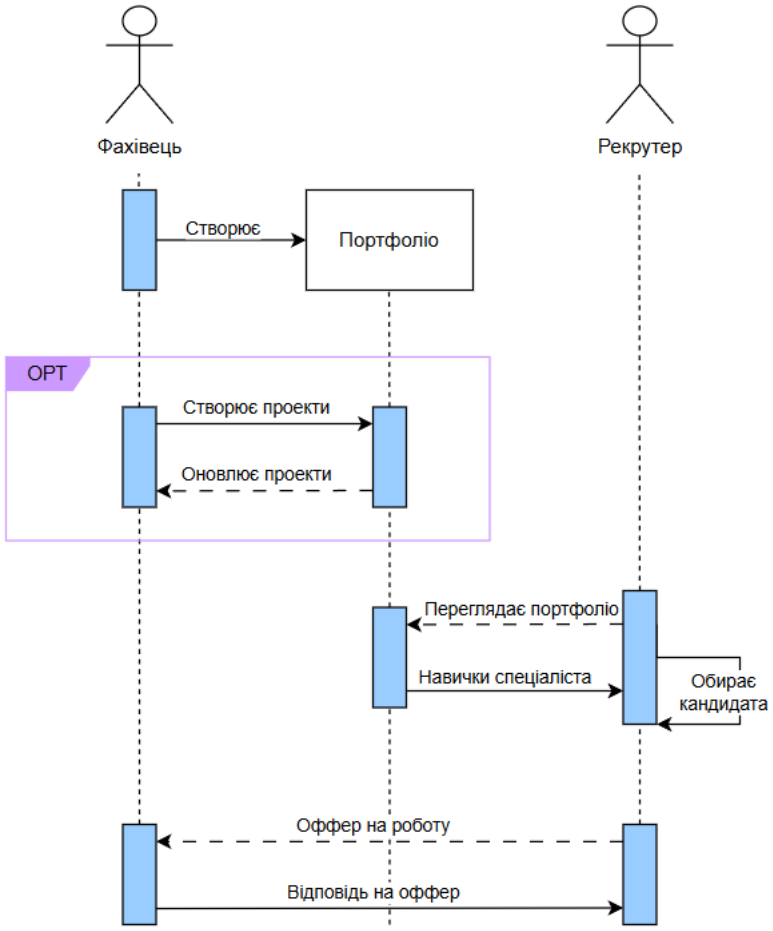

Рис. 1.2 – Діаграма послідовності (Sequence Diagram) бізнес-процесів предметної області

Важливою складовою аналізу є деталізація алгоритмів прийняття рішень всередині системи, особливо в аспектах кібербезпеки. Для моделювання бізнес-логіки управління правами доступу розроблено діаграму активності (Activity Diagram). Алгоритм описує процес доступу до вихідних файлів проєкту. Дія розпочинається з ініціації користувачем (рекрутером) завантаження.

Система отримує метадані проєкту і переходить до блоку прийняття рішення (Decision Node) – перевірки дозволу на скачування у базі даних. Якщо автор заборонив доступ, алгоритм переривається повідомленням про помилку доступу (HTTP 403). Якщо дозвіл є, система генерує безпечну тимчасову URL-адресу з хмарного сховища та ініціалізує фізичне завантаження файлу на пристрій клієнта.

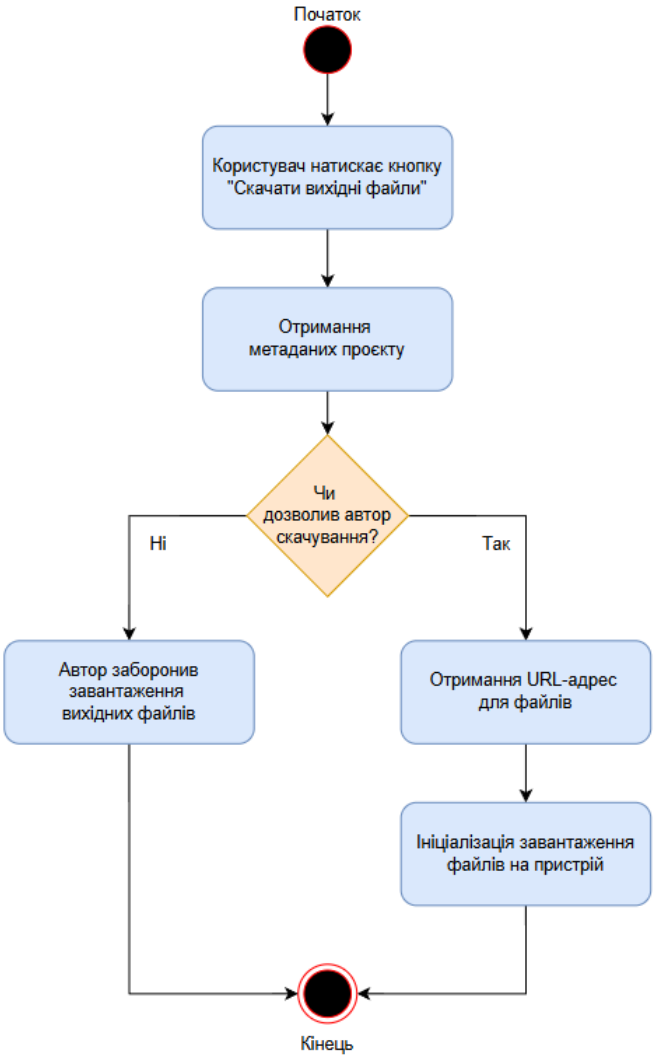

Рис. 1.3 – Діаграма активності (Activity Diagram) алгоритму доступу до файлів

Комплексне використання цих трьох моделей дозволило повністю формалізувати предметну область, визначити межі системи, розподілити ролі та зафіксувати ключові бізнес-правила до початку етапу безпосередньої програмної реалізації.

1.3 Огляд існуючих аналогічних систем та інформаційних джерел

1. LinkedIn. Найбільша у світі професійна мережа для нетворкінгу [2] представлено на рисунку 1.4.

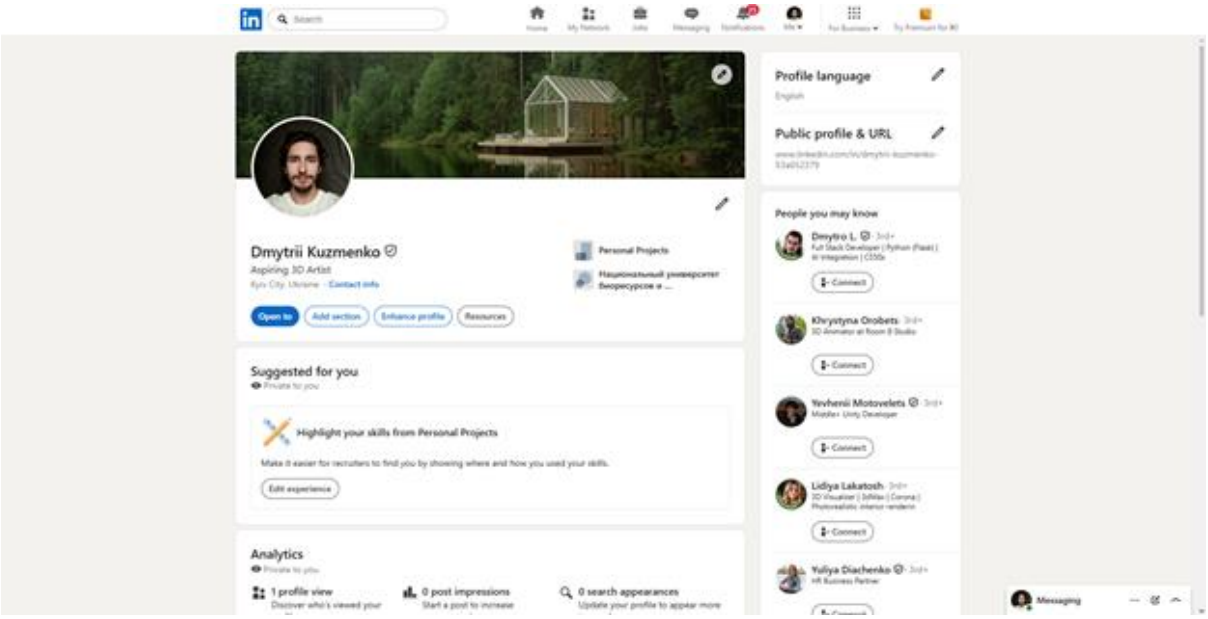

Рис. 1.4 – профіль користувача LinkedIn

- Переваги (з точки зору ПЗ): високонавантажена розподілена архітектура мікросервісів, здатна обробляти мільйони запитів; потужні алгоритми рекомендацій та пошуку на базі машинного навчання; стабільна та оптимізована підсистема обміну повідомленнями в реальному часі; розвинений API для сторонніх інтеграцій.
- Недоліки (з точки зору ПЗ): жорстка реляційна структура бази даних, орієнтована переважно на текстову інформацію та стандартизовані резюме; відсутність вбудованого клієнтського рушія (наприклад, на базі WebGL) для рендерингу та інтерактивної взаємодії з 3D-файлами у браузері; медіасховище застосовує агресивні алгоритми стиснення, що псують якість зображень високої роздільної здатності.

2. **Sketchfab.** Провідна платформа для публікації та перегляду 3D-моделей [3] представлено на рисунку 1.5.

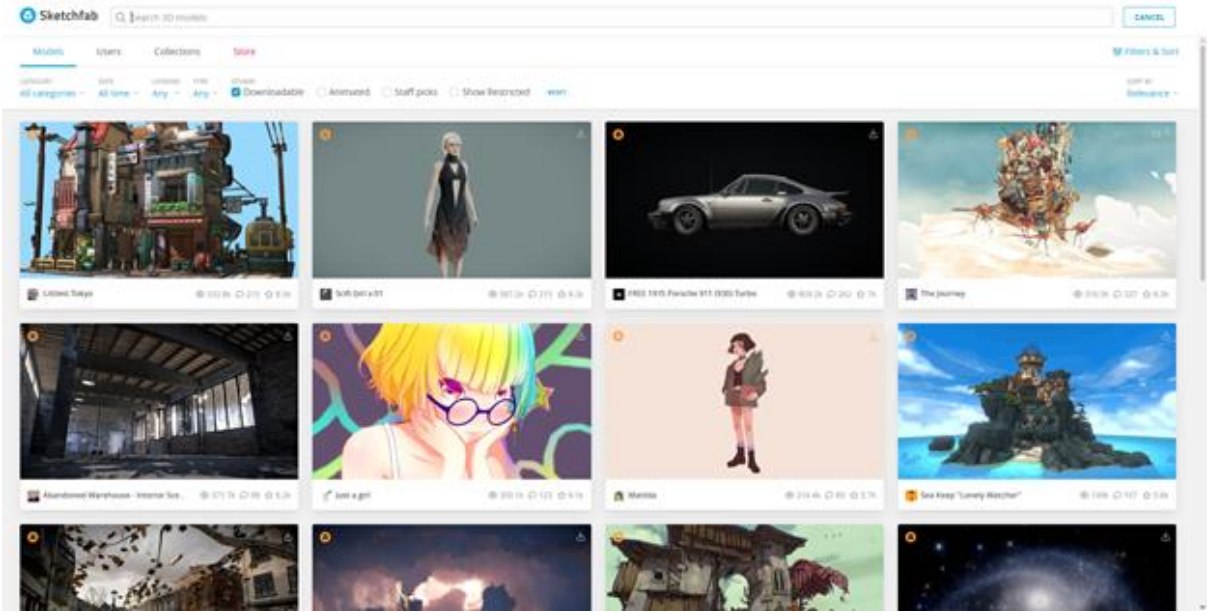

Рис. 1.5 – пошукова стрічка Sketchfab

- Переваги (з точки зору ПЗ): передовий клієнтський 3D-в'ювер, що напряму використовує апаратне прискорення через WebGL API; високоефективні алгоритми потокового завантаження (streaming) та стиснення 3D-даних (підтримка форматів .gltf, .glb); складна система кешування PBR-матеріалів (Physically Based Rendering) та динамічного обчислення освітлення.
- Недоліки (з точки зору ПЗ): архітектура ПЗ не оптимізована для глибокої соціальної взаємодії — відсутні повноцінні модулі формування динамічної стрічки новин (Activity Feed) для публікації текстових дописів та підсистема миттєвого обміну повідомленнями (чату); слабка реляційна складова для побудови складних зв'язків між акторами (наприклад, система не адаптована для потреб HR-фахівців).

3. **ArtStation.** Професійна вітрина для цифрових художників [4] представлено на рисунку 1.6.

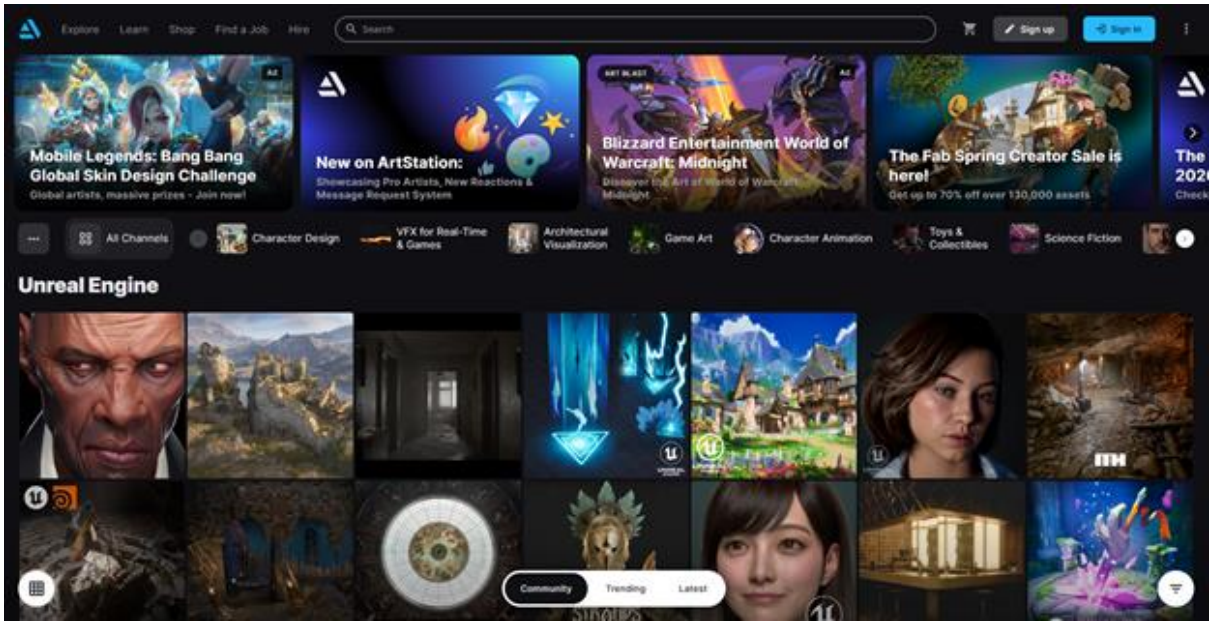

Рис. 1.6 – головна сторінка ArtStation

- Переваги (з точки зору ПЗ): ефективне використання глобальних мереж доставки контенту (CDN) для миттєвого завантаження 2D-зображень без відчутної втрати якості; оптимізований DOM-рендеринг для відображення нескінченних grid-галерей без падіння продуктивності браузера; надійна система тегування та класифікації візуального контенту.
- Недоліки (з точки зору ПЗ): базова інтеграція 3D-моделей часто реалізується через сторонні iframe-рішення (той же Sketchfab), що обмежує можливості внутрішньої кастомізації платформи розробниками; відсутність повноцінної архітектури для ведення діалогів у реальному часі (real-time chat), що змушує рекрутерів використовувати сторонні поштові сервіси для зв'язку з кандидатами.

Отже, інженерний аналіз показує, що існує потреба у розробці спеціалізованого програмного забезпечення, яке б на архітектурному рівні поєднувало комунікаційні можливості (Real-time Database) професійних мереж з технологіями апаратного рендерингу (WebGL/Three.js) для візуалізації 3D-об'єктів.

1.4 Постановка завдання

Метою даної кваліфікаційної роботи є проєктування та розробка програмного забезпечення інтерактивної вебплатформи, призначеної для публікації портфоліо, професійного нетворкінгу та найму фахівців у сфері цифрового мистецтва та 3D-моделювання.

На основі проведеного аналізу предметної області та виявлених недоліків існуючих рішень (розділи 1.1–1.3), до розроблюваної системи висувається комплексний перелік вимог. Відповідно до стандартної інженерної практики розробки програмного забезпечення, вимоги розділено на три основні категорії: функціональні, нефункціональні та вимоги до технічного забезпечення.

1.4.1 Функціональні вимоги до системи. Система повинна безперебійно забезпечувати виконання наступного набору логічних операцій:

1. Підсистема автентифікації та управління користувачами:

- Реєстрація та авторизація користувачів із використанням електронної пошти та захищеного пароля.
- Розділення користувачів на дві логічні ролі («Спеціаліст» та «Рекрутер») із відповідною динамічною зміною користувацького інтерфейсу та набору доступних функцій.
- Налаштування профілю: завантаження аватара, заповнення текстової біографії, додавання тегів професійних навичок (Skills) та послуг (Services) для спеціалістів; зазначення реквізитів компанії для рекрутерів.
- Можливість повного та безповоротного видалення облікового запису користувачем разом із усіма пов'язаними медіаданими з метою дотримання політик конфіденційності.

2. Підсистема управління портфоліо (Проєктами):

- Створення багатокомпонентних проєктів трьох типів: 3D-моделі, 2D-галереї (із підтримкою множинного завантаження зображень) та Відео.
- Класифікація проєктів за визначеними категоріями (Characters, Environment, Animation, Props / Objects, UI/UX Interface) для полегшення пошуку.
- Інтерактивний 3D-в'ювер: апаратний рендеринг завантажених .glb файлів безпосередньо у вікні браузера.
- Наявність діагностичної панелі «Model Inspector» для 3D-моделей з можливістю перемикання режимів перегляду матеріалів (Base Color, Wireframe для оцінки топології, Matcap).
- Можливість дозволити або жорстко заборонити завантаження вихідного бінарного файлу проєкту сторонніми користувачами.

3. Підсистема соціальної взаємодії та нетворкінгу:

- Реалізація глобальної стрічки проєктів із можливістю текстового пошуку за назвою та категорійної фільтрації.
- Персональна стрічка активності (Activity Feed) у профілі користувача для швидкої публікації новин, відкритих вакансій або оновлень робочого процесу (WIP).
- Підтримка галерейного відображення зображень у стрічці (grid-галерея), якщо до допису прикріплено декілька фотографій.
- Можливість ставити відмітки «Подобається» (Likes) та залишати текстові коментарі до проєктів і дописів із миттєвим оновленням лічильників.

4. Підсистема комунікації в реальному часі:

- Вбудований віджет месенджера, доступний поверх основного контенту для авторизованих користувачів.

- Відправлення первинної форми-запиту зі сторінки профілю спеціаліста із обов'язковим зазначенням теми (Subject) майбутнього повідомлення.
- Відображення списку активних діалогів з індикацією кількості непрочитаних повідомлень (червоний бейдж сповіщень).
- Миттєва доставка, збереження у базі даних та відображення повідомлень у вікні активного чату без необхідності перезавантаження сторінки (WebSocket).

1.4.2 Нефункціональні вимоги. Зручність використання (Usability): Інтерфейс користувача (UI) має бути побудований за принципом Single Page Application (SPA) [12], що забезпечує миттєву навігацію між розділами платформи без перезавантаження сторінок та втрати поточного контексту. Дизайн повинен бути сучасним, інтуїтивно зрозумілим, з використанням модальних вікон для перегляду контенту, щоб користувач не втрачав позицію у глобальній стрічці.

- Критерії валідації: Отримання оцінки не нижче 90 балів за показниками «Accessibility» (Доступність) та «Best Practices» під час автоматизованого аудиту інструментом Google Lighthouse [21]. Архітектура навігації має забезпечувати доступ до будь-якого цільового проєкту не більше ніж за 3 кліки з головної сторінки.

Продуктивність (Performance): Апаратний рендеринг 3D-графіки має працювати плавно, не перевантажуючи ресурси клієнтського пристрою. Моделі повинні автоматично масштабуватися та центруватися у просторі в'ювера незалежно від їх початкових світових координат, заданих у редакторі авторами.

- Критерії валідації: Підтримка стабільної частоти кадрів на рівні не менше 45-60 FPS (кадрів на секунду) під час активного обертання та панорамування базових 3D-моделей (до 100 тисяч полігонів) у браузері. Показник First Contentful Paint (FCP) при першому завантаженні головної

сторінки додатка не повинен перевищувати 2 секунди за умови стабільного інтернет-з'єднання.

Надійність та Безпека (Security): Паролі користувачів не повинні зберігатися у відкритому вигляді чи передаватися мережею без шифрування. Доступ до операцій створення, редагування та видалення проєктів, дописів чи коментарів повинен надаватися виключно їх авторам після валідації токена доступу. База даних повинна бути алгоритмічно захищена від несанкціонованого запису або читання приватних повідомлень третіми особами.

Критерії валідації: Успішне проходження перевірок серверними правилами безпеки (імітація запитів від неавторизованих клієнтів або спроба модифікації чужого документа має повертати статус HTTP 403 Permission Denied). Авторизаційні токени та всі мережеві транзакції повинні передаватися виключно через зашифровані канали (протоколи HTTPS та WSS).

1.4.3 Вимоги до технічного забезпечення та архітектури. Платформа повинна мати розподілену клієнт-серверну архітектуру з використанням хмарних технологій:

- Клієнтська частина (Frontend): Має виконуватися у середовищі сучасних веббраузерів (Google Chrome, Opera, Mozilla Firefox, Safari, Edge) та базуватися на стеку технологій HTML5, CSS3, JavaScript (ES6+).
- Вимоги до рендерингу: Для візуалізації 3D-графіки необхідна обов'язкова підтримка технології WebGL з боку браузера [15] та апаратного забезпечення клієнтського пристрою (наявність інтегрованого або дискретного GPU).
- Серверна частина (Backend): Система не вимагає розробки власного монолітного сервера на базі Node.js чи PHP. Доцільно використати керовану

хмарну інфраструктуру типу Backend-as-a-Service (BaaS) для зберігання масивних мультимедійних файлів, ведення гнучкої нереляційної бази даних та безпечної автентифікації.

Таким чином, розроблювана система відноситься до класу спеціалізованих вебплатформ із високим мультимедійним навантаженням. Її реалізація потребує вибору та інтеграції сучасного стеку вебтехнологій, здатного гарантувати високу реактивність користувацького інтерфейсу, надійне зберігання даних та ефективну роботу з тривимірною графікою у середовищі браузера.

Висновки до розділу 1

У першому розділі проведено комплексний системний аналіз предметної області щодо процесу створення та управління цифровим портфоліо фахівців креативних індустрій. Виконано інженерний огляд існуючих платформ-аналогів (LinkedIn, Sketchfab, ArtStation) та виявлено їхні архітектурні недоліки у контексті поєднання інтерактивного 3D-рендерингу з інструментами професійного нетворкінгу. За допомогою уніфікованої мови моделювання (UML) розроблено діаграми прецедентів, послідовності та активності, що дозволило алгоритмізувати бізнес-процеси та чітко розмежувати ролі користувачів («Фахівець» і «Рекрутер»). Результатом розділу є розроблений детальний перелік функціональних, нефункціональних (продуктивність, безпека, юзабіліті) та архітектурних вимог, який став надійним базисом для подальшого проєктування системи.

2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Основою якісного проєктування інформаційного забезпечення будь-якої вебплатформи є побудова логічної структури даних. Вона визначає ключові сутності предметної області, їхні атрибути, типи даних та логічні взаємозв'язки між ними на концептуальному рівні, незалежно від фізичного способу реалізації сховища чи обраної системи управління базами даних (СУБД). Відповідно до архітектурної концепції розроблюваної платформи, що комплексно поєднує можливості професійного нетворкінгу та інтерактивної демонстрації важкого мультимедійного контенту, логічну модель було формалізовано за допомогою ER-діаграми (Entity-Relationship) у стандартизованій нотації Crow's Foot («вороняча лапка»).

Розроблена логічна схема відображає структуру даних, яка, з одного боку, оптимізована для забезпечення складних багатовимірних зв'язків між користувачами різних ролей, а з іншого — чітко структурує ієрархію мультимедійних компонентів портфоліо.

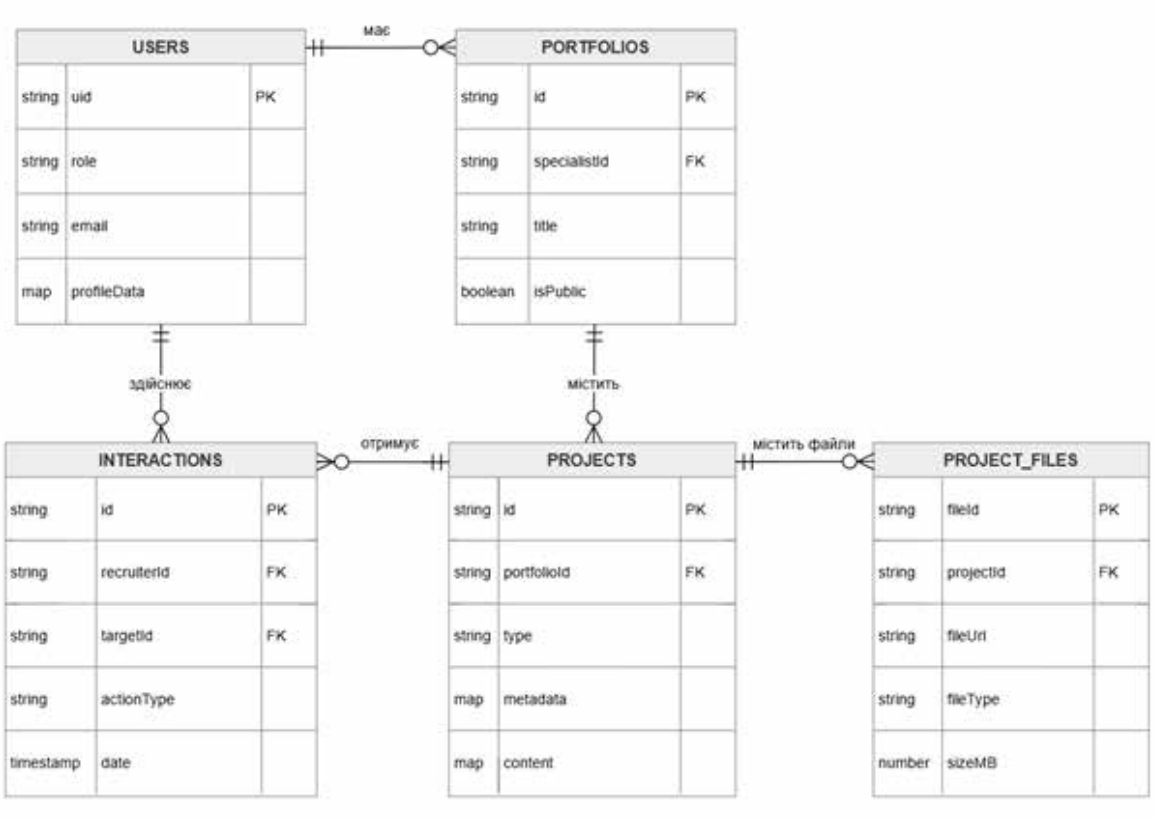

Рис. 2.1 – Логічна модель даних (ER-діаграма)

Аналіз представленої логічної моделі (рис. 2.1) дозволяє виділити п'ять базових сутностей, які формують інформаційний каркас системи:

1. USERS (Користувачі): Центральна сутність системи, яка інкапсулює ідентифікаційні, авторизаційні та профільні дані користувача платформи.

- uid (string, PK) — первинний ключ (Primary Key), що є унікальним криптографічним ідентифікатором користувача, який генерується підсистемою автентифікації.
- role (string) — визначає логічну роль користувача в системі ("specialist" або "recruiter") для маршрутизації та динамічної перебудови інтерфейсу.
- email (string) — електронна адреса користувача для зворотного зв'язку та верифікації сесії.
- profileData (map) — структурований тип даних типу «ключ-значення», який гнучко вміщує розширену інформацію профілю

(біографія, масив тегів навичок, посилання на соціальні мережі, назва компанії тощо).

2. PORTFOLIOS (Портфоліо): Сутність, що виступає логічним контейнером для групування творчих робіт конкретного фахівця.

- id (string, PK) — унікальний ідентифікатор портфоліо.
- specialistId (string, FK) — зовнішній ключ (Foreign Key), який пов'язує портфоліо з конкретним записом (власником) у таблиці USERS.
- title (string) — назва або заголовок цифрової вітрини.
- isPublic (boolean) — прапорець видимості, що діє як глобальний перемикач доступу сторонніх користувачів до вмісту.

3. PROJECTS (Проєкти): Одиниця контенту, яка безпосередньо представляє творчу або професійну роботу (3D-модель, відеопрезентацію, галерею зображень).

- id (string, PK) — унікальний ідентифікатор проєкту.
- portfolioId (string, FK) — зовнішній ключ, що вказує на батьківське портфоліо.
- type (string) — класифікатор типу мультимедійного контенту ("3D", "2D", "Video").
- metadata (map) — специфічні метадані проєкту, які можуть змінюватися залежно від його типу (наприклад, категорія, кількість полігонів, налаштування камери WebGL-в'ювера, колір фону сцени).
- content (map) — динамічна структура, що містить текстовий опис проєкту, а також масиви коментарів та відміток від інших користувачів.

4. PROJECT_FILES (Файли проєкту): Сутність, що описує фізичні бінарні ресурси, які завантажені у хмарне сховище та пов'язані з конкретним проєктом.

- fileId (string, PK) — унікальний ідентифікатор файлу.

- projectId (string, FK) — посилання на проєкт, якому належить файл.
- fileUrl (string) — абсолютне HTTPS-посилання на фізичне розташування згенерованого файлу (blob) у мережі доставки контенту (CDN).
- fileType (string) — розширення або MIME-тип файлу (наприклад, ".glb", ".mp4", "image/png").
- sizeMB (number) — розмір файлу в мегабайтах для контролю квот завантаження та попередження користувачів перед завантаженням.

5. INTERACTIONS (Взаємодії / Логи): Сутність, що забезпечує збір аналітики та реалізує процеси нетворкінгу, надійно фіксуючи активність рекрутерів щодо кандидатів та їхніх робіт.

- id (string, PK) — унікальний ідентифікатор ініційованої події.
- recruiterId (string, FK) — посилання на користувача-рекрутера, який здійснив цільову дію.
- targetId (string, FK) — посилання на цільовий проєкт або профіль фахівця.
- actionType (string) — класифікатор типу взаємодії (наприклад, "view", "like", "chat_init", "send_offer").
- date (timestamp) — точний часовий штамп фіксації взаємодії сервером.

Аналіз кардинальності зв'язків між сутностями вказує на суворе дотримання бізнес-логіки системи. Зв'язок між USERS та PORTFOLIOS (відношення «має») має характер 1:N (один до багатьох), де з боку користувача діє жорстке обмеження цілісності (double bar || — один і тільки один користувач-власник), а з боку портфоліо допускається наявність кількох логічних контейнерів контенту. Сутність PORTFOLIOS пов'язана з PROJECTS (відношення «містить») як 1:N, що дозволяє фахівцю гнучко наповнювати портфоліо необмеженою кількістю робіт. Кожен об'єкт PROJECTS

декомпонується на декілька фізичних файлів (наприклад, сама 3D-модель та її прев'ю-обкладинка) у сутності PROJECT_FILES (зв'язок 1:N, «містить файли»).

Сутність INTERACTIONS реалізує зв'язки типу багатьох-до-багатьох (M:N) між користувачами та проєктами через використання проміжних зовнішніх ключів recruiterId та targetId, формуючи повну неперервну хронологію взаємодії у системі.

2.2 Обґрунтування нереляційної структури даних

Розроблена на попередньому етапі логічна модель даних може бути класично інтерпретована як реляційна структура (система взаємопов'язаних таблиць). Проте для фізичної реалізації розроблюваної платформи було прийнято обґрунтоване інженерне рішення відмовитися від SQL-баз даних на користь нереляційної документо-орієнтованої структури (NoSQL) на базі СУБД Firebase Cloud Firestore [6]. Це архітектурне рішення продиктоване декількома вагомими факторами:

- Денормалізація для досягнення високої продуктивності: Вебплатформа орієнтована на часті операції читання (Read-heavy workload) під час формування глобальної стрічки проєктів. У реляційних базах даних для відображення картки проєкту (з коментарями, лайками, посиланнями на файли та даними автора) необхідно виконувати ресурсомісткі операції об'єднання декількох таблиць (JOIN), що знижує швидкість відповіді сервера. У документо-орієнтованій NoSQL-моделі сутності, які мають жорстку ієрархічну підпорядкованість, цілеспрямовано денормалізуються. Наприклад, дані з сутностей PORTFOLIOS та PROJECT_FILES інтегруються безпосередньо в єдиний документ проєкту у вигляді вкладених масивів об'єктів та асоціативних карт (Map). Це дозволяє клієнтському додатку отримати всю необхідну інформацію для рендерингу компонента за один надшвидкий атомарний запит до бази даних.

- Гнучкість схеми даних (Schemaless архітектура): Проєкти фахівців креативних індустрій мають кардинально різні набори метаданих. Поля, необхідні для налаштування 3D-моделі (кількість полігонів, інтенсивність джерел світла, колір фону WebGL-сцени), не мають жодного логічного сенсу для портфоліо з відеофайлами чи 2D-ілюстраціями. Використання типу даних map (metadata, profileData) вказує на те, що BSON-документи можуть динамічно розширювати та змінювати структуру своїх полів "на льоту", без необхідності створення порожніх колонок (NULL) та проведення складних глобальних міграцій схеми БД, як це вимагається у класичних реляційних системах.

2.3 Вибір системи управління базами даних

Для управління інформаційною базою розроблюваної системи обрано екосистему Firebase, ядром якої виступає хмарна NoSQL СУБД **Cloud Firestore**. Вона працює за моделлю Backend-as-a-Service (BaaS). Головними критеріями такого вибору стали:

- Нативна підтримка Real-time синхронізації: На відміну від традиційних БД, які працюють за моделлю "запит-відповідь" (HTTP REST), Firestore здатна підтримувати постійне з'єднання з клієнтом через протокол WebSocket. Використання слухачів onSnapshot дозволяє реалізувати підсистему миттєвого обміну повідомленнями (чат) та динамічне оновлення лічильників лайків без необхідності розробки додаткового проміжного серверного шару на Node.js.
- Безшовна інтеграція з авторизаційними сервісами: Зв'язок ідентифікаторів (uid) відбувається на системному рівні хмарної інфраструктури. Це спрощує контроль цілісності даних, оскільки база даних напряму "розуміє", який користувач робить запит через токени Firebase Authentication.
- Автоматичне масштабування та індексація: Фізичний розподіл даних по шардах (Sharding), підтримка високої доступності (High Availability) та

створення індексів для складних складених запитів повністю делеговані серверним потужностям Google Cloud, що звільняє розробника від рутинного адміністрування інфраструктури.

2.4 Розробка структури та фізичної схеми інформаційної бази

На фізичному рівні СУБД Cloud Firestore не підтримує поняття "таблиць" та "рядків". Дані зберігаються у вигляді бінарних JSON-об'єктів (BSON) [7], які називаються документами, а документи об'єднуються у колекції. Логічну ER-діаграму було трансформовано у плоску ієрархічну структуру колекцій із застосуванням патернів денормалізації. Фрагмент фізичної реалізації розробленої бази даних у середовищі Firebase Console наведено на рис. 2.2.

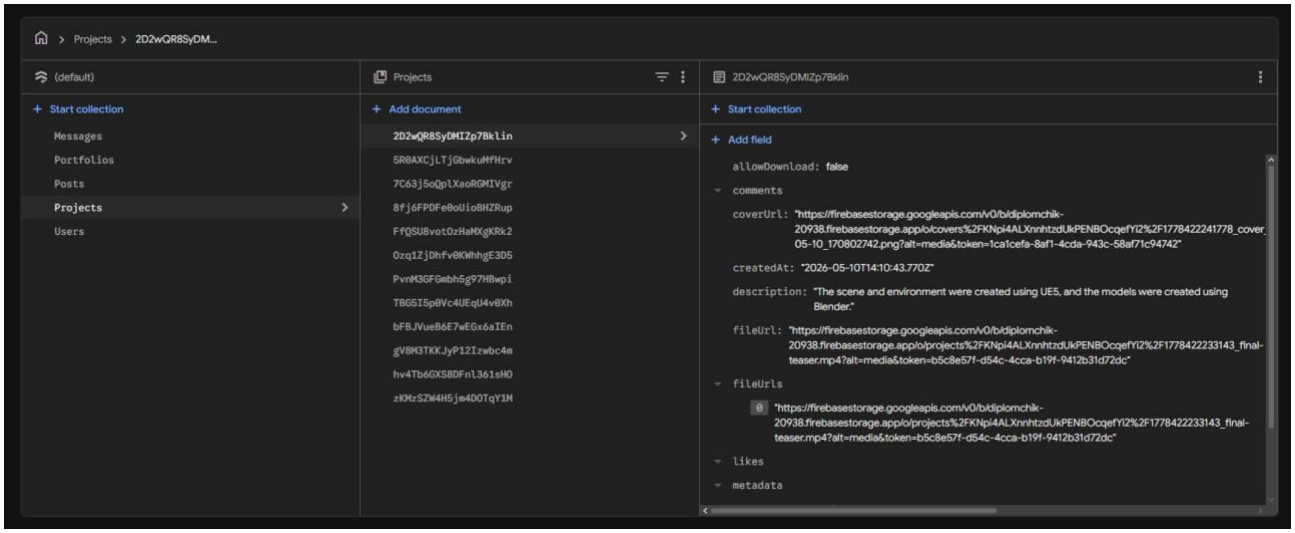

Рис. 2.2 – Структура колекцій та документів у хмарній базі даних Cloud Firestore

2.4.1 Колекція Users. Ідентифікатором документа є uid (відповідає первинному ключу сутності USERS).

- displayName (string) — повне ім'я користувача.
- role (string) — роль ("specialist" / "recruiter").
- email (string) — адреса пошти.

- profileData (map) — вкладений об'єкт профілю: { bio: string, skills: array, profession: string, company: string, avatarUrl: string }.

2.4.2 Колекція Projects. Ця колекція об'єднує в собі логічні сутності PORTFOLIOS, PROJECTS та PROJECT_FILES в один єдиний денормалізований документ. Це критично важливо для прискорення завантаження глобальної стрічки та оптимізації кількості запитів до БД.

- id (string) — унікальний ідентифікатор проєкту (PK), згенерований Firestore.
- authorId (string) — посилання на ідентифікатор документа автора у колекції Users (FK).
- title (string) — назва опублікованої роботи.
- type (string) — тип візуального контенту ("3D", "2D", "Video").
- description (string) — розширений текстовий опис.
- metadata (map) — специфічні параметри відображення об'єкта (наприклад, { category: string, polyCount: number, backgroundColor: string }).
- fileUrls (array of strings) — масив абсолютних HTTPS-посилань на мультимедійні файли, збережені в Cloud Storage. Інтеграція файлів у вигляді масиву замінює окрему таблицю PROJECT_FILES.
- coverUrl (string) — посилання на згенероване прев'ю (обкладинку) проєкту для відображення у загальній сітці стрічки.
- allowDownload (boolean) — параметр доступу сторонніх осіб до вихідників.
- interactions (map) — вкладена структура для швидкого обліку соціальної активності: { likesCount: number, likedBy: array, comments: array }.

2.4.3 Колекція Messages (Підсистема комунікації). Колекція оптимізована для ведення логів та обміну приватними повідомленнями в реальному часі. Кожне повідомлення є окремим документом.

- id (string) — унікальний ідентифікатор повідомлення.
- senderId (string) — ID відправника.
- receiverId (string) — ID отримувача.
- subject (string) — тема звернення (тема вакансії або пропозиції).
- message (string) — текст повідомлення.
- isRead (boolean) — прапорець статусу прочитання.
- createdAt (timestamp) — час генерації повідомлення.

Безпека та цілісність інформаційної бази на фізичному рівні гарантуються правилами безпеки Firestore Security Rules [11]. Записи у колекції Messages захищені правилом, яке дозволяє операції читання та запису виключно користувачам, чий поточний токен автентифікації (request.auth.uid) збігається зі значенням полів senderId або receiverId. Це повністю виключає можливість несанкціонованого доступу сторонніх осіб до конфіденційних листувань та комерційних офферів рекрутерів.

Висновки до розділу 2

У другому розділі здійснено проєктування інформаційного забезпечення розроблюваної вебплатформи. Сформовано логічну ER-модель даних у нотації Crow’s Foot, яка описує ключові сутності предметної області (Users, Portfolios, Projects, Project_Files, Interactions) та кардинальність зв'язків між ними. Інженерно обґрунтовано архітектурне рішення щодо відмови від класичних реляційних СУБД на користь документо-орієнтованої структури (NoSQL) на базі Firebase Cloud Firestore. Це забезпечило гнучкість схеми даних, швидке читання за рахунок денормалізації об'єктів та вбудовану підтримку WebSocket-синхронізації для чатів. Сформовано фізичну ієрархію колекцій та визначено

строгі політики безпеки Firestore Security Rules для криптографічного захисту конфіденційних даних від несанкціонованого доступу.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Абстракції предметної області

Етап об’єктно-орієнтованого проєктування програмної системи традиційно розпочинається з концептуального аналізу та виділення ключових абстракцій предметної області. Для формалізації структури вебплатформи на ранніх етапах проєктування було визначено базові сутності, їхні ключові властивості (атрибути) та обов'язки (поведінку), що інкапсулюються всередині відповідних класів.

Головною абстракцією розроблюваної системи є «Фахівець (Кандидат)». Властивості цієї абстракції включають ідентифікаційні та контактні дані, спеціалізацію (наприклад, 3D-моделювання, концепт-арт) та деталізований перелік професійних навичок (hard skills). До обов'язків фахівця, з точки зору бізнес-логіки системи, належить повне управління життєвим циклом професійного портфоліо: ініціалізація завантаження файлів, налаштування параметрів конфіденційності, демонстрація рівня кваліфікації через наочні приклади робіт та обробка вхідних комунікаційних запитів.

Абстракція: Фахівець (Кандидат)
Важливі властивості: контактні дані; спеціалізація (наприклад, 3D-моделювання); професійні навички (компетенції);
Обов’язки: формувати професійне портфоліо; демонструвати рівень кваліфікації через приклади робіт; керувати власними проектами; дозвіл на скачування вихідників;

Рис. 3.1 – Опис абстракції «Фахівець (Кандидат)»

Другою важливою абстракцією, що представляє сторону бізнесу та споживачів контенту, є «Рекрутер (Роботодавець)». Властивості цієї абстракції зосереджені навколо критеріїв пошуку та реквізитів компанії, яку він представляє. Обов'язки інкапсулюють процеси фільтрації бази фахівців за необхідними навичками, технічний аудит робіт кандидатів (зокрема аналіз сітки 3D-моделей) та ініціювання діалогів для надсилання пропозицій щодо працевлаштування (офферів).

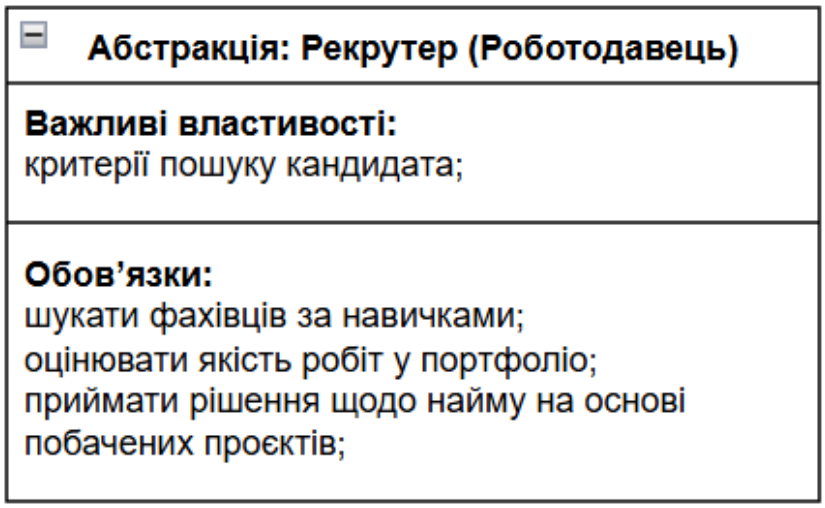

Рис. 3.2 – Опис абстракції «Рекрутер (Роботодавець)»

Базовою абстракцією контенту, що обробляється системою, є «Проєкт Елемент портфоліо)». До його важливих властивостей належать: назва, унікальний ідентифікатор, посилання на бінарний 3D-файл або інший медіафайл у хмарному сховищі, а також булевий прапорець дозволу на завантаження вихідників. Основний обов'язок цієї абстракції — надати структуровані дані для клієнтського інтерактивного рендерингу та наочно продемонструвати застосовані інструментальні навички автора.

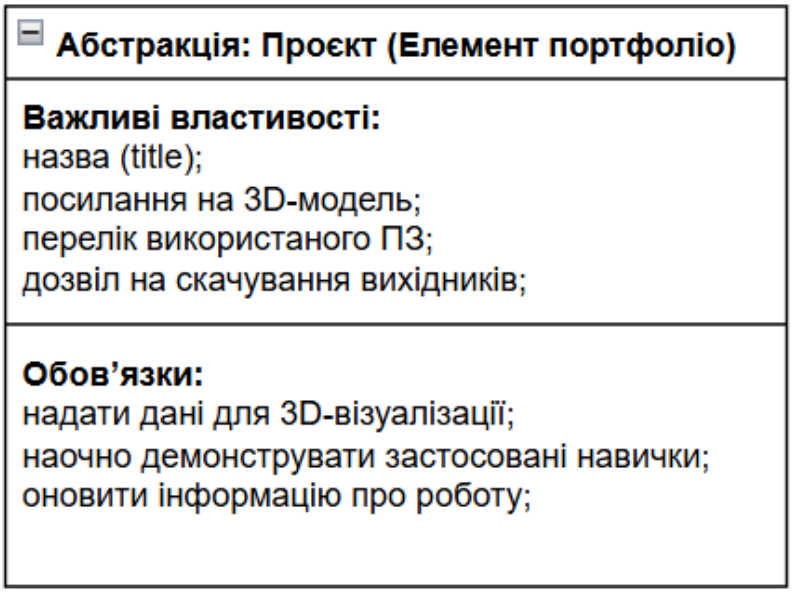

Рис. 3.3 – Опис абстракції «Проєкт (Елемент портфоліо)»

На основі виділених абстракцій побудовано загальну статичну об'єктну модель системи у вигляді діаграми класів (Class Diagram) мовою UML. Вона ілюструє типи даних та операції, а також рівні доступу до них (public, private, protected). Для моделювання ролей створено класи Specialist та Recruiter, які можуть успадковувати базові атрибути від загального суперкласу User. Клас Specialist інкапсулює методи управління контентом (наприклад, createProject(), updateVisibility()). Клас Recruiter містить методи пошуку та взаємодії (searchProjects(), initiateChat()).

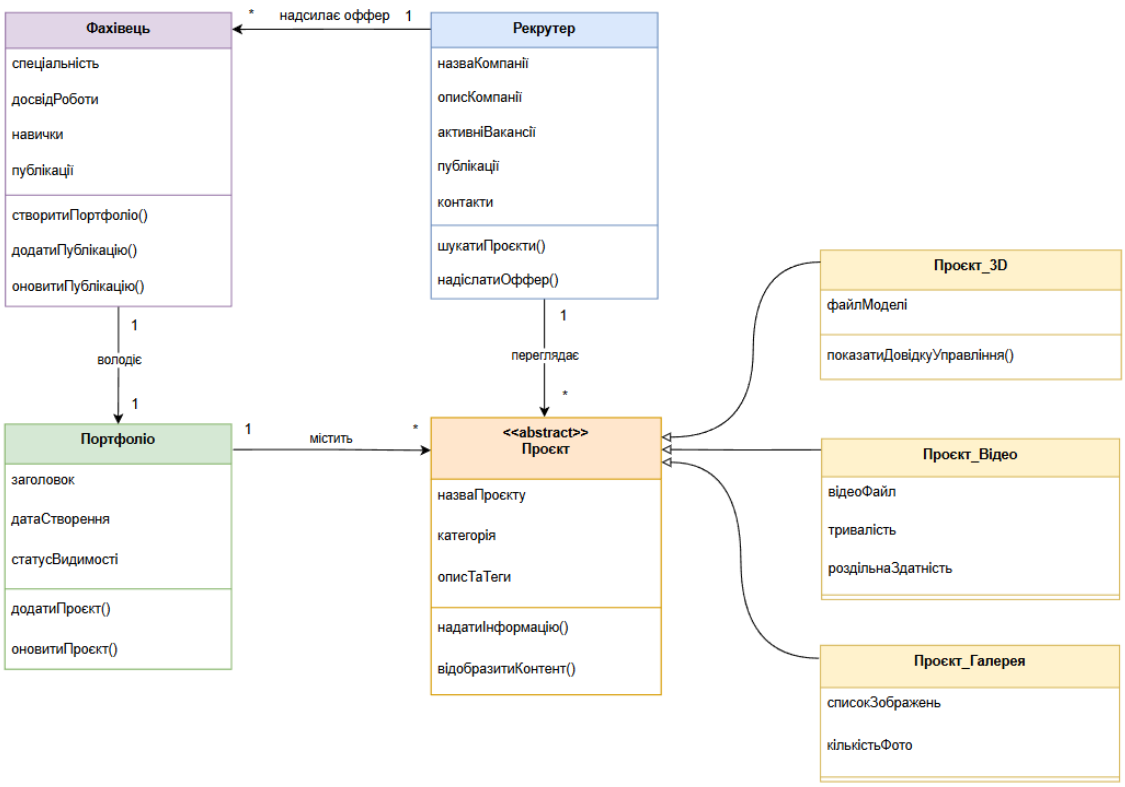

Рис. 3.4 – Загальна діаграма класів предметної області

Організація мультимедійного контенту побудована за принципом об'єктної композиції: екземпляр класу Specialist володіє одним екземпляром класу Portfolio, який, у свою чергу, містить агреговану множину об'єктів класу Project. Щоб забезпечити високу гнучкість системи при роботі з кардинально різними типами медіа, застосовано фундаментальний патерн ООП — поліморфізм. Створено базовий абстрактний клас <<abstract>> Project, від якого успадковуються конкретні спеціалізовані класи-реалізації: Project_3D (містить специфічні параметри WebGL-сцени), Project_Video (містить посилання на відеопотік) та Project_Gallery (містить масив посилань на 2D-зображення).

Рис. 3.5 – Структура портфоліо та наслідування проєктів

Взаємодія між рекрутером та інформаційною базою формалізована через асоціативні зв'язки. Рекрутер аналізує екземпляри класів-нащадків Project та безпосередньо контактує з екземплярами класу Specialist через допоміжний клас Message (який формує підсистему чату).

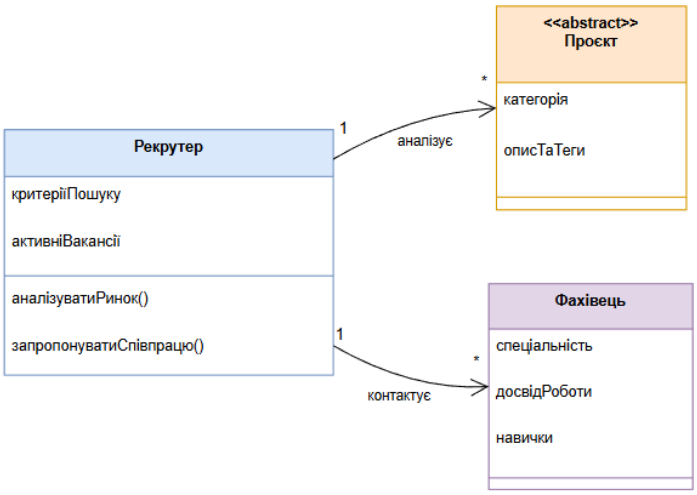

Рис. 3.6 – Взаємодія рекрутера з контентом та фахівцями

3.2 Організаційна структура програмного забезпечення

Для ефективного управління складністю вихідного коду, підвищення його читабельності та суворого дотримання принципів «низького зачеплення» (low coupling) і «високої зв'язності» (high cohesion), архітектуру системи розділено на логічні пакети. Оскільки розроблюваний програмний продукт є складним односторінковим додатком (SPA), який поєднує класичний двовимірний DOM-

інтерфейс та апаратний 3D-рендеринг, модульний підхід є критично необхідним для уникнення конфліктів стану та запобігання витокам оперативної пам'яті.

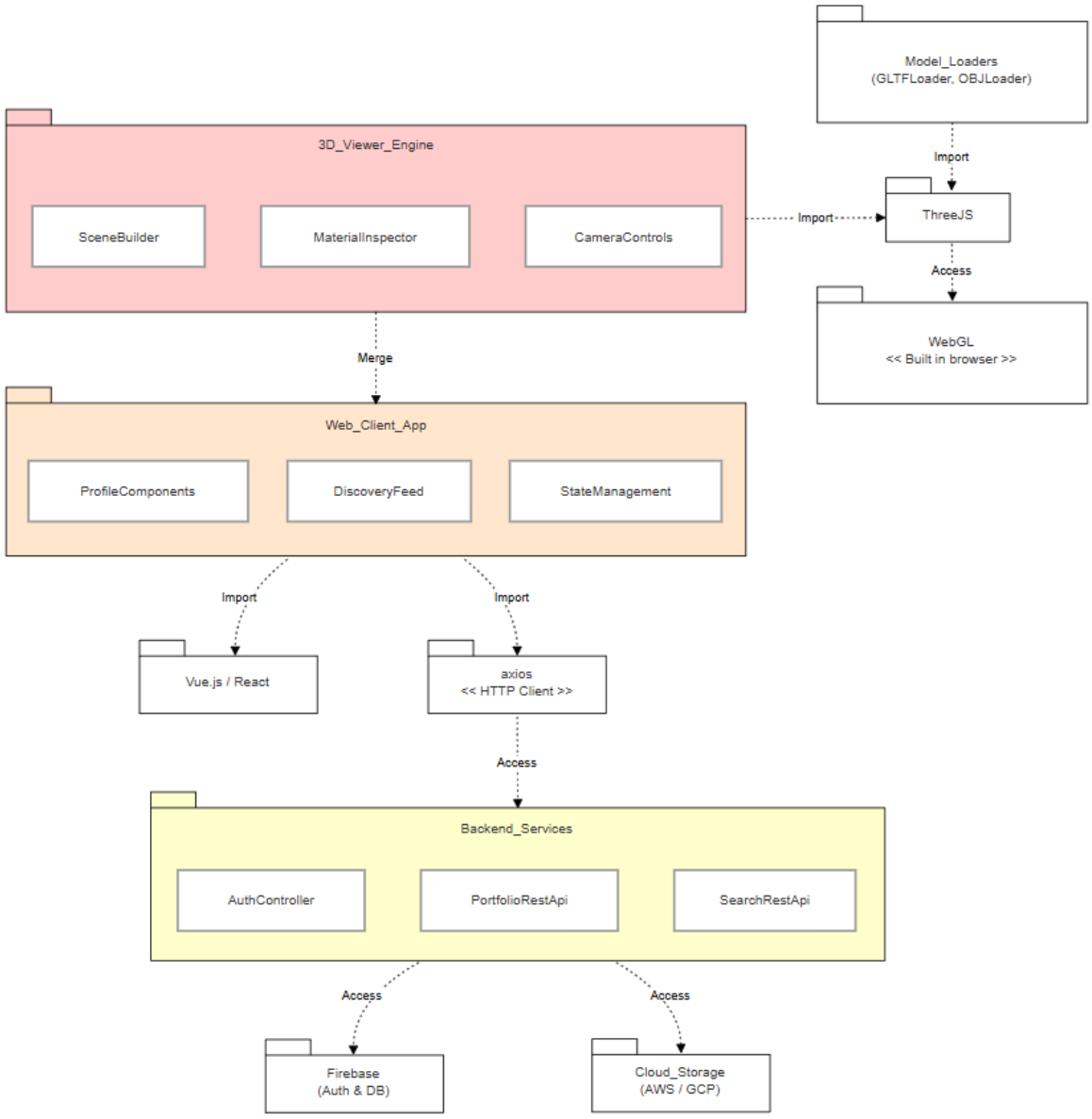

Рис. 3.7 – Діаграма пакетів організаційної структури ПЗ

Глобально архітектуру розділено на три основні функціональні блоки (пакети), кожен з яких відповідає за ізольовану частину бізнес-логіки:

1. Пакет Web_Client_App (Клієнтський рівень) Цей пакет є ядром фронтенд-додатка, побудованого на базі фреймворку Vue.js 3, і відповідає за формування реактивного користувацького інтерфейсу. Він містить наступні внутрішні модулі:

- Views (Представлення): Компоненти-сторінки верхнього рівня, що формують екрани системи (наприклад, глобальна стрічка проєктів, персональний профіль, панель налаштувань).
- Components (UI-компоненти): Перевикористовувані, ізольовані елементи інтерфейсу (кнопки, модальні вікна, картки портфоліо, форми введення), які динамічно компонуються у представлення.
- Router (Маршрутизатор): Модуль навігації, що забезпечує клієнтський перехід між сторінками без перезавантаження сторінки браузера. Включає логіку «захисників маршрутів» (navigation guards) для перевірки прав доступу до приватних розділів.
- State Management (Управління станом): Модуль, який зберігає та синхронізує глобальний стан додатка (поточну сесію авторизованого користувача, дані профілю), забезпечуючи єдине джерело істини (Single Source of Truth) для всіх компонентів.

2. Пакет 3D_Viewer_Engine (Рушій рендерингу) Специфічний і ресурсомісткий функціонал обробки 3D-графіки свідомо винесено у повністю ізольований пакет. Він інкапсулює всю складну математичну логіку взаємодії з графічною бібліотекою Three.js і контекстом WebGL, тим самим суворо відокремлюючи її від дерева DOM-елементів Vue.js. Пакет включає:

- SceneBuilder (Конструктор сцени): Відповідає за ініціалізацію WebGL-рендерера, налаштування віртуальної камери та генерацію фізично коректного навколишнього освітлення (PMREMGenerator).
- ModelLoader (Завантажувач моделей): Модуль асинхронного парсингу та завантаження бінарних файлів форматів .glb/.gltf.
- MaterialInspector (Інспектор матеріалів): Аналітичний модуль технічного аудиту, який містить алгоритми динамічного підрахунку полігонів, розбиття PBR-матеріалів на діагностичні канали (Base Color, Emission) та активації інженерних режимів відображення (Wireframe, Matcap).

3. Пакет Backend_Services (Сервісний рівень інтеграції) Взаємодія з базою даних, файловим сховищем та системою авторизації абстрагована у цьому пакеті, що виступає прошарком (Service Layer) між графічним інтерфейсом та хмарною інфраструктурою Firebase. Такий паттерн проєктування дозволяє модифікувати серверну логіку без ризику порушити роботу UI-компонентів. Пакет поділяється на підмодулі:

- AuthService: Інкапсулює методи реєстрації, автентифікації користувачів та управління сесійними токенами.
- DBService (Служба бази даних): Містить абстракції для виконання CRUD-операцій (створення, читання, оновлення, видалення) над документами у Firestore, а також логіку встановлення WebSocket-з'єднань для отримання оновлень у реальному часі (підсистема чатів та коментарів).
- StorageService (Служба сховища): Керує процесами завантаження мультимедійних файлів (зображень, відео, 3D-моделей) до Cloud Storage, отримує безпечні URL-посилання для їх відображення та обробляє політики спільного доступу (CORS).

Таким чином, запропонована організаційна структура створює безпечний односпрямований потік даних: компоненти Web_Client_App викликають методи Backend_Services для отримання інформації з бази, а при необхідності візуалізації тривимірних об'єктів — передають отримані посилання на файли у незалежний екземпляр 3D_Viewer_Engine. Це гарантує високу продуктивність графіки, загальну масштабованість системи та зручність її подальшої підтримки.

3.3 Компонентна структура системи

Діаграма компонентів (Component Diagram) ілюструє фізичний поділ програмної системи на виконувані модулі та визначає стандартизовані інтерфейси їхньої взаємодії. Оскільки система реалізується за концепцією Single

Page Application (SPA), весь клієнтський вебдодаток упаковується у єдиний скомпільований модуль bundle.js, який завантажується у браузер. Всередині цього бандлу логіка розподілена між такими основними програмними компонентами:

- User_Interface – компонент, відповідальний за реактивну генерацію HTML-розмітки та обробку користувацького вводу.
- Auth_Manager – компонент управління життєвим циклом сесій (логін, реєстрація, логаут), що взаємодіє з хмарою через стандартизований інтерфейс IAuth.
- Data_Manager – модуль, що забезпечує виконання CRUD-операцій (створення, читання, оновлення, видалення) для проєктів і повідомлень через інтерфейс IDatabase.
- ThreeJS_Renderer – ізольований графічний компонент, який монтується в DOM-дерево, отримує бінарні файли через інтерфейс IStorage і здійснює апаратну обробку геометрії через інтерфейс IWebGL.

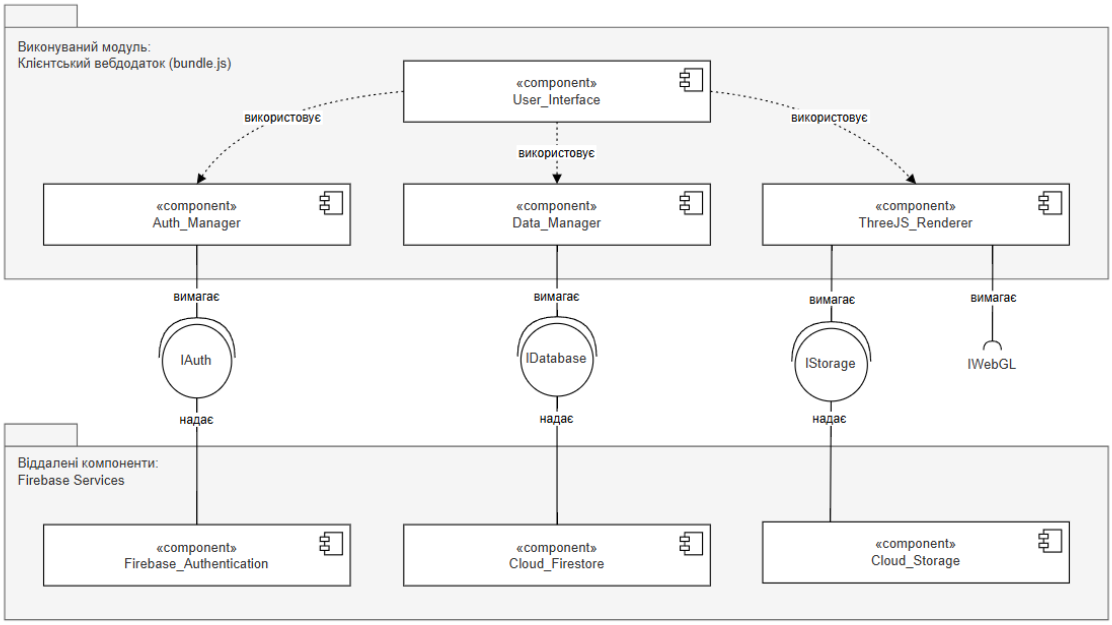

Рис. 3.8 – Діаграма компонентів архітектури системи

3.4 Вибір інструментарію для створення прикладного ПЗ

Для успішної та швидкої реалізації вебплатформи, що відповідає сучасним вимогам продуктивності та безпеки, було проведено аналіз і вибір актуального стеку технологій. Архітектура базується на поєднанні прогресивного фронтенд-фреймворку та хмарної інфраструктури:

1. Vue.js 3 (Composition API). Прогресивний JavaScript-фреймворк [13], обраний як основний інструмент для побудови клієнтського інтерфейсу. Використання нового підходу Composition API дозволяє гнучко інкапсулювати та перевикористовувати логіку компонентів. Головною перевагою Vue.js є його система реактивності та використання Virtual DOM, що є критично важливим для забезпечення миттєвого оновлення динамічних стрічок новин, лічильників лайків та вікон чату без перезавантаження сторінки.

2. Vite. Сучасний інструмент збірки фронтенд-проєктів (build tool) [14]. Обраний на заміну традиційному Webpack завдяки використанню нативних ES-модулів (ESM) у браузері. Vite забезпечує миттєвий запуск сервера розробки та надшвидке гаряче оновлення модулів (Hot Module Replacement - HMR), що суттєво пришвидшує процес розробки та тестування складних інтерфейсів.

3. Three.js. Високорівнева кросбраузерна JavaScript-бібліотека [16]. Використовується для реалізації власного 3D-в'ювера. Вона слугує обгорткою над низькорівневим та складним WebGL API, надаючи готові абстракції для управління графами сцени (Scene Graph), камерами, матеріалами (PBR) та освітленням, що дозволяє рендерити 3D-моделі безпосередньо на HTML5 Canvas із залученням апаратного прискорення GPU.

4. Firebase (Auth, Firestore, Storage). Хмарна платформа від Google, що реалізує концепцію Backend-as-a-Service (BaaS). Її використання дозволило повністю відмовитися від розробки та адміністрування власного монолітного сервера. Firebase Authentication забезпечує надійну автентифікацію [9] користувачів, Cloud Firestore слугує гнучкою NoSQL базою даних із підтримкою

WebSocket-з'єднань для миттєвої синхронізації повідомлень, а Cloud Storage виступає надійним хмарним сховищем для зберігання оптимізованих медіафайлів великого обсягу.

Висновки до розділу 3

У третьому розділі виконано об’єктно-орієнтоване проєктування та розроблено архітектуру прикладного програмного забезпечення. За допомогою UML-моделювання (діаграми класів, пакетів та компонентів) виділено ключові абстракції системи та реалізовано поліморфізм для гнучкої обробки різних типів медіаконтенту (3D, 2D, Відео). Обрано та обґрунтовано сучасний стек технологій: фреймворк Vue.js 3 для побудови реактивного SPA-інтерфейсу, інструмент Vite для оптимізованої модульної збірки, високорівневу графічну бібліотеку Three.js для апаратного WebGL-рендерингу та хмарну інфраструктуру Firebase (BaaS). Описано логіку взаємодії ізольованого графічного рушія з компонентами користувацького інтерфейсу та сервісами автентифікації.

4. ВПРОВАДЖЕННЯ ТА ПРАКТИЧНЕ ВИКОРИСТАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Конфігурування операційного оточення та налаштування залежностей

Перед початком розгортання та тестування розробленої вебплатформи для створення та управління портфоліо фахівців креативних індустрій необхідно підготувати середовище виконання. Це середовище має забезпечити стабільну роботу клієнтської частини, безперебійний апаратний рендеринг тривимірної графіки у браузері, інтеграцію з хмарною базою даних та коректну реалізацію механізмів обміну повідомленнями в реальному часі. Основними компонентами середовища виконання є інструментарій Node.js, менеджер пакетів npm, середовище збірки Vite для запуску Single Page Application (SPA), а також хмарні сервіси інфраструктури Google Firebase.

Для локальної розробки та подальшої підготовки клієнтської частини до впровадження використовується програмна платформа Node.js версії не нижче 18.x (LTS) [18]. Використання актуальної версії Node.js забезпечує повну сумісність із сучасними бібліотеками та утилітами, які застосовуються у проєкті. Клієнтська частина базується на прогресивному фреймворку Vue.js 3 та системі автоматизації збірки Vite, які орієнтовані на сучасні стандарти ECMAScript і потребують підтримки останніх можливостей JavaScript runtime. Node.js у даній архітектурі використовується для запуску локального сервера розробки, управління зовнішніми залежностями через npm-пакети та компіляції фінального інсталяційного бандлу. Завдяки архітектурі інструменту Vite, який використовує нативні ES-модулі браузера, суттєво прискорюється процес гарячого оновлення (Hot Module Replacement), що дозволяє розробнику миттєво відстежувати зміни

в інтерфейсі під час тестування функціоналу 3D-в'ювера без повного перезавантаження сторінки.

У якості менеджера пакетів застосовується інструмент npm [19], за допомогою якого встановлюються та контролюються всі залежності програмного забезпечення, зафіксовані у конфігураційному файлі package.json. До базових бібліотек системи, що формують середовище виконання, належать: Core-пакет Vue 3, маршрутизатор Vue Router для організації SPA-навігації, графічна бібліотека Three.js для роботи з WebGL-контекстом, контролери OrbitControls для інтерактивного керування камерою, а також комплексний хмарний пакет Firebase SDK.

На відміну від традиційних клієнт-серверних вебдодатків, у розробленій програмній системі не використовується окремий виділений серверний бекенд або локальна реляційна СУБД. Архітектура системи повністю побудована за сучасною безсерверною концепцією Backend-as-a-Service (BaaS) на базі екосистеми Firebase.

Для візуалізації фізичної топології розробленої системи та розподілу її компонентів побудовано діаграму розгортання (Deployment Diagram), яку наведено на рис. 4.1. Як видно з діаграми, фізична архітектура складається з вузла клієнта (пристрій користувача з веббраузером) та групи хмарних серверних вузлів: Firebase Hosting для роздачі статичних файлів, Cloud Firestore для BSON-документів та Cloud Storage для збереження важких медіафайлів (.glb, відео).

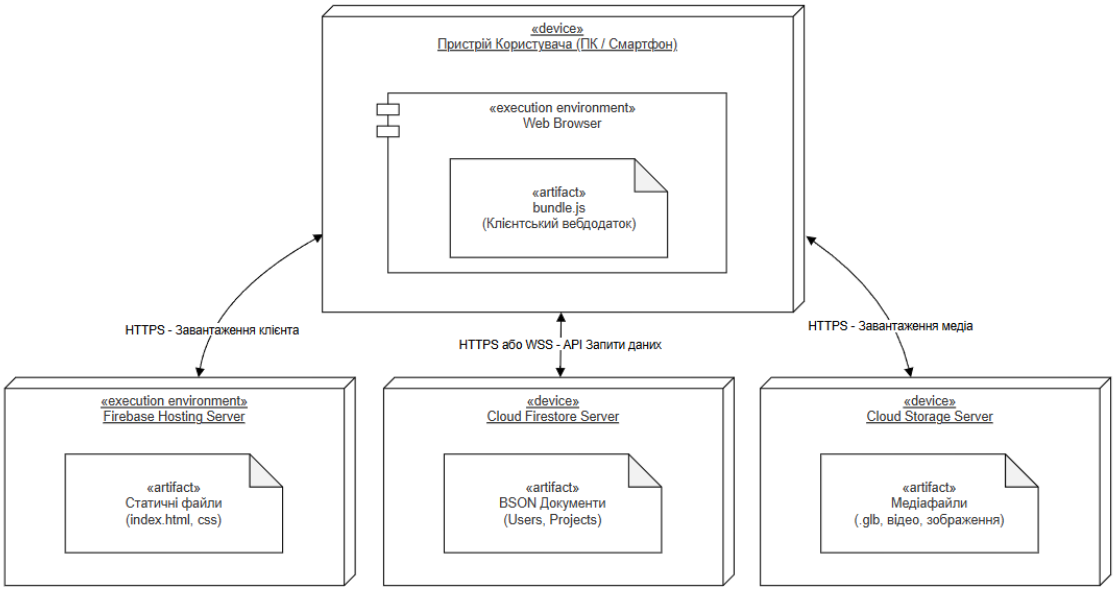

Рис. 4.1 – Діаграма розгортання (Deployment Diagram) фізичної архітектури системи

Хмарна нереляційна СУБД Cloud Firestore виступає основним сховищем даних, забезпечуючи низьку затримку та автоматичну синхронізацію інформації між користувачами через протокол WebSocket [8]. Для підтримки механізмів ідентифікації та автентифікації користувачів інтегровано сервіс Firebase Authentication. Цей інструмент дозволяє безпечно реалізувати процеси реєстрації, авторизації, підтримки сесій користувачів (на базі захищених JWT-токенів) та розмежування прав доступу на рівні ролей «Спеціаліст» та «Рекрутер» без необхідності написання та адміністрування власного авторизаційного сервера.

Для успішного функціонування клієнтської частини та ініціалізації з'єднань із хмарною інфраструктурою в корені проєкту створюється конфігураційний файл змінних середовища .env. У цьому файлі у вигляді зашифрованих ключів задаються параметри підключення до Firebase, зокрема: унікальний API key, project ID, auth domain, storage bucket, messaging sender ID та app ID. Наявність цих змінних є обов'язковою умовою для того, щоб модулі

автентифікації, бази даних Firestore та хмарного сховища Cloud Storage коректно ініціалізувалися на стороні веббраузера клієнта під час старту вебплатформи.

Завдяки використанню безсерверної архітектури та хмарних сервісів процес підготовки середовища виконання значно спрощується порівняно із класичними багаторівневими системами, де необхідно окремо налаштовувати операційні системи серверів, вебсервери типу Nginx або Apache, бази даних та складну мережеву інфраструктуру. Такий підхід дозволяє швидко розгорнути систему, забезпечити її лінійну масштабованість та спростити подальшу технічну підтримку.

4.2 Розміщення та деплой програмного продукту у хмарному середовищі

Процес розгортання розробленого програмного забезпечення (деплой) включає підготовку вихідного коду, конфігурацію параметрів безпеки, локальний запуск для проведення внутрішнього аудиту та фінальну публікацію оптимізованих файлів на віддаленому хмарному хостингу для надання відкритого доступу користувачам.

Першим етапом розгортання є завантаження або клонування репозиторію з вихідним кодом проєкту на локальну робочу станцію. Після цього через інтерфейс командного рядка здійснюється перехід до кореневої директорії додатка, де запускається команда npm install. Менеджер пакетів автоматично аналізує дерево залежностей файлу package.json, завантажує необхідні версії бібліотек та формує локальну директорію node_modules. Наступним кроком є розміщення файлу .env із актуальними параметрами хмарного сховища Google Firebase, як це було описано у попередньому підрозділі.

Після успішного налаштування конфігураційних файлів виконується запуск системи у режимі розробки за допомогою команди npm run dev. Оптимізований сервер розробки Vite ініціалізує локальний хост та надає

мережеву адресу для тестування системи у браузері. Під час цього етапу клієнтський додаток автоматично виконує ініціалізацію Firebase SDK, встановлює захищене з'єднання з Cloud Firestore та готує Canvas-елементи для рендерингу графіки за допомогою Three.js.

Для перевірки коректності розгортання виконується базовий набір сценаріїв:

- реєстрація нового облікового запису та перевірка успішного створення документа користувача у хмарі;
- завантаження тестової 3D-моделі у форматі .glb для перевірки працездатності лоадерів та ініціалізації WebGL-контексту;
- відправлення тестових повідомлень між різними профілями для перевірки real-time слухачів бази даних.

Фінальним етапом впровадження системи є її деплой у production-середовище. Для цього спочатку запускається команда компіляції npm run build. Інструмент збірки Vite використовує Rollup для виконання глибокої оптимізації вихідного коду: транспілює синтаксис Vue 3 у кросбраузерний JavaScript, проводить мініфікацію стилів та скриптів, застосовує алгоритми tree-shaking для видалення невикористаного коду бібліотек, а також здійснює бандлінг (розбиття коду на оптимізовані частини — chunks — для прискорення завантаження сторінок). Результатом збірки є статична директорія dist.

Розгортання сформованої директорії на віддаленому сервері здійснюється через інтерфейс Firebase CLI. За допомогою послідовних команд firebase login (авторизація розробника), firebase init hosting (вибір хмарного проєкту та призначення папки dist як публічної) та firebase deploy файли завантажуються на сервери Firebase Hosting [20]. Хмарний хостинг автоматично виділяє захищений SSL/TLS-сертифікат та надає постійну URL-адресу з доступом по протоколу HTTPS. Використання розподіленої мережі доставки контенту (CDN) гарантує високу швидкість завантаження портфоліо з будь-якої географічної точки,

оскільки файли кешуються на найближчих до користувача граничних серверах (edge nodes).

4.3 Адміністрування та розмежування прав доступу

Адміністрування та керування даними у розробленій вебплатформі реалізовано за гнучким дворівневим підходом, який поєднує вбудований безпосередньо у клієнтський інтерфейс функціонал керування правами доступу (Role-Based Access Control) та глобальне адміністрування інфраструктури бази даних через хмарну консоль розробника Firebase Console.

Внутрішньосистемне адміністрування базується на рольовій моделі доступу, яка жорстко розділяє можливості акторів на етапі авторизації. Перевірка приналежності користувача до певної категорії здійснюється асинхронно: після успішного логіну система зчитує поле role з документа користувача в колекції Users бази даних Firestore. Залежно від значення ("specialist" або "recruiter"), додаток динамічно змінює конфігурацію Vue Router та склад доступних компонентів інтерфейсу:

1. Адміністрування з боку Спеціаліста: Користувач-творець володіє повними правами адміністрування власного контенту (CRUD-операції). Він може створювати, редагувати опис або повністю видаляти свої проєкти та дописи. Важливою адмін-функцією на рівні користувача є динамічне перемикання прапорця allowDownload. Зміна цього параметра миттєво оновлює правила доступу до бінарного файлу моделі, що зберігається у Cloud Storage. Крім того, спеціаліст може моделювати візуальний стан своєї цифрової вітрини, налаштовуючи індивідуальний колір фону для 3D-сцени через палітру інспектора.

2. Адміністрування з боку Рекрутера: Для представників компаній активується розширений пошуковий та оціночний інструментарій. Рекрутер управляє параметрами фільтрації стрічки, здійснює технічний аудит

завантажених ассетів (перегляд топологічної сітки, аналіз matcap-матеріалів) та ініціює створення нових каналів зв'язку через надсилання первинних форм-запитів фахівцям.

Глобальне адміністрування інфраструктури системи здійснюється розробником через вебконсоль Firebase. Через цей інтерфейс адміністратор має можливість моніторити метрики активності бази даних, переглядати логи авторизацій, вручну блокувати або видаляти користувачів у разі порушення ними правил платформи, а також контролювати обсяг дискового простору та мережевого трафіку у хмарному сховищі Cloud Storage [10].

Безпека даних та захист від несанкціонованого адміністрування зловмисниками реалізовані на рівні серверних правил декларативного програмування Firestore Security Rules. Завдяки цим правилам клієнтський додаток фізично позбавлений можливості модифікувати чужі документи, навіть шляхом втручання у вихідний код фронтенду. Операції запису та видалення дозволені сервером лише тоді, коли криптографічний ідентифікатор авторизованої сесії користувача request.auth.uid (переданий у JWT-токені) повністю збігається з полем authorId цільового документа. Таким чином, адміністрування системи є децентралізованим, безпечним та не потребує утримання штату системних адміністраторів.

4.4 Працездатність та інтерфейс вебплатформи

Першим екраном, який бачить користувач під час відкриття вебплатформи у браузері, є головна сторінка, поєднана з глобальною стрічкою відкритих проєктів. Інтерфейс виконано у стриманій темній кольоровій гамі, що є фундаментальним правилом UI/UX дизайну для портфоліо-сервісів цифрових художників, оскільки такий фон зменшує навантаження на очі та не спотворює сприйняття кольорів і контрасту на зображеннях та 3D-моделях.

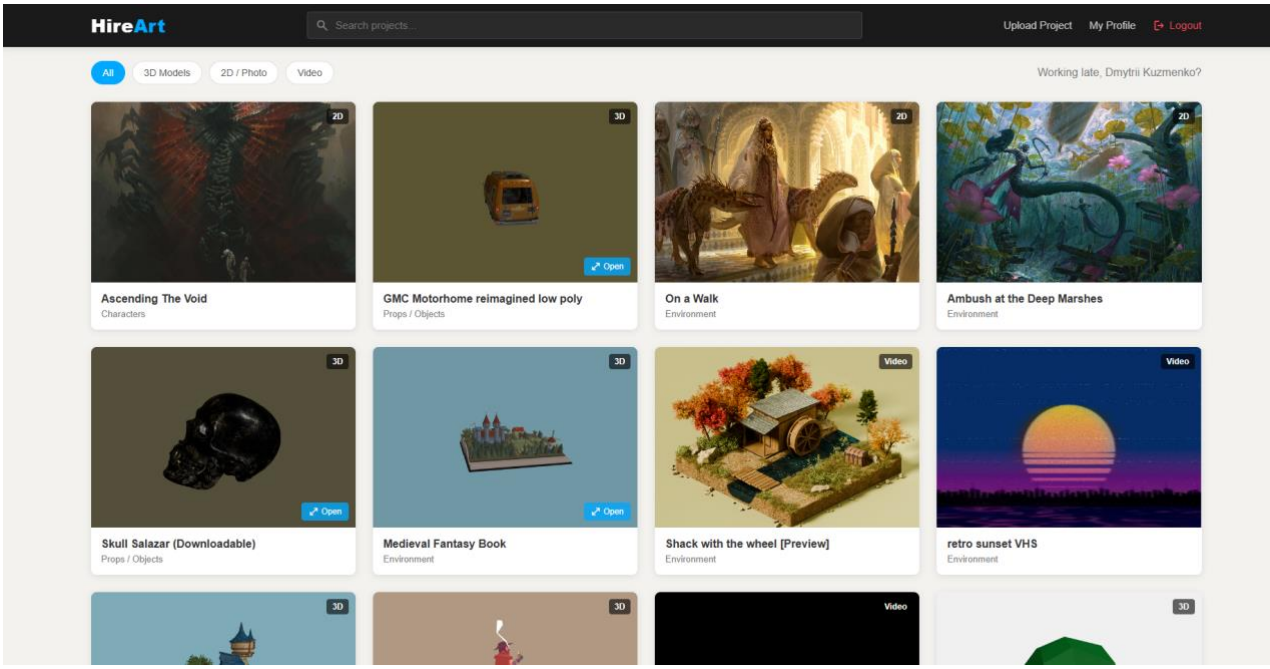

Рис. 4.2 – Головна сторінка вебплатформи (стрічка проєктів)

У верхній частині екрана розташовано функціональний хедер системи, який містить логотип, інтегрований пошуковий рядок та елементи керування сесією. Нижче розташована панель категорійної фільтрації, яка дозволяє миттєво відсіювати контент за обраними типами. Основну площу екрана займає адаптивна сітка (CSS Grid), що відображає картки проєктів різних авторів. Кожна картка містить згенероване прев'ю-зображення роботи, її назву, ім'я автора та лічильники соціальної взаємодії.

Для отримання доступу до функцій створення власного портфоліо або зв'язку з кандидатами користувачеві необхідно пройти процес автентифікації. При натисканні на кнопку входу відкривається спеціалізоване модальне вікно або окрема сторінка авторизації, що підтримує історію маршрутизації (History API).

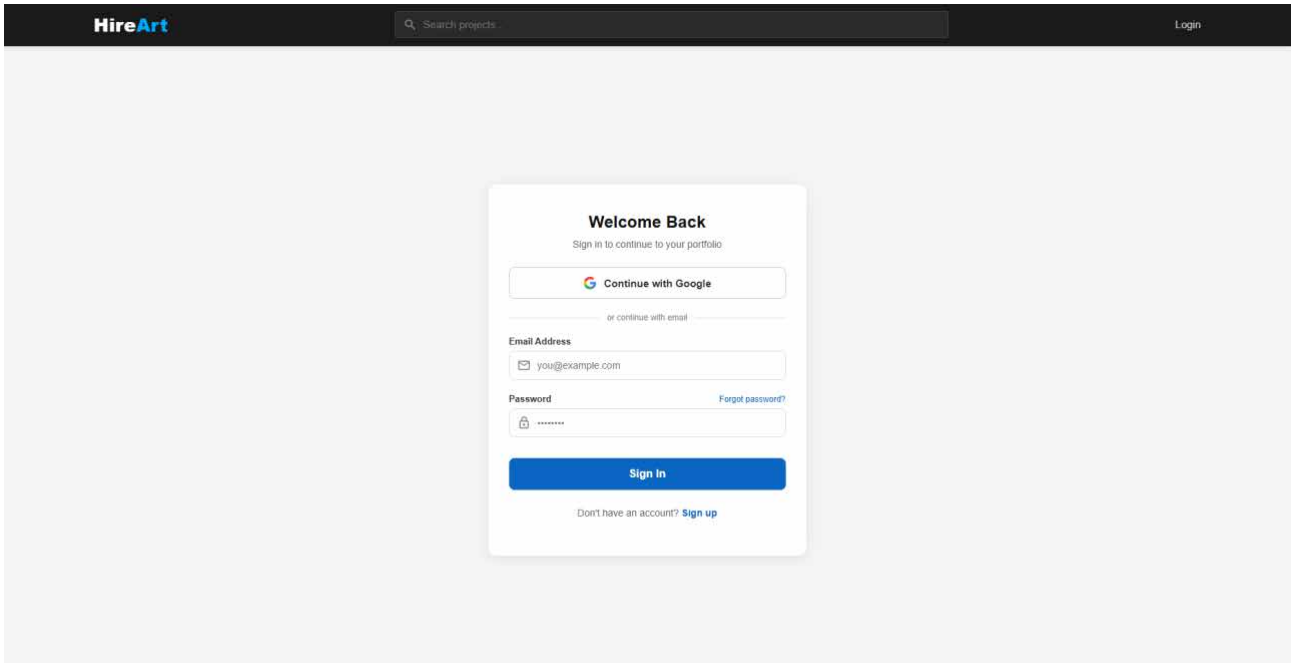

Рис. 4.3 – Сторінка автентифікації користувачів

Форма авторизації містить стандартні поля для введення адреси електронної пошти та пароля, а також посилання для переходу до режиму реєстрації нового облікового запису. Система виконує первинну клієнтську валідацію полів (перевірка формату email, довжини пароля), після чого надсилає асинхронний запит до Firebase Authentication. У разі успішного збігу даних інтерфейс миттєво перебудовується під авторизовану сесію без перезавантаження сторінки.

4.4.1 Огляд інструментарію перегляду контенту: 3D-інспектор, 2D-галерея та відеоплеєр. Після проходження автентифікації користувач отримує повний доступ до робочого вікна системи. Ключовим етапом взаємодії з контентом є модальне вікно детального перегляду проєкту, яке активується при кліку на будь-яку картку у загальній стрічці. Залежно від типу збереженого контенту, система динамічно монтує відповідний модуль відображення (3D, 2D або Відео), кожен з яких має специфічний функціонал для технічної оцінки робіт.

4.4.1.1 Інтерактивний 3D-в'ювер та панель «Model Inspector». Якщо обраний проєкт має тип «3D», система ініціалізує контекст WebGL та запускає Three.js рушій. Головною інновацією модуля є спеціалізована бічна панель «Model Inspector», призначена для глибокого технічного аудиту 3D-моделей рекрутерами та лід-артистами.

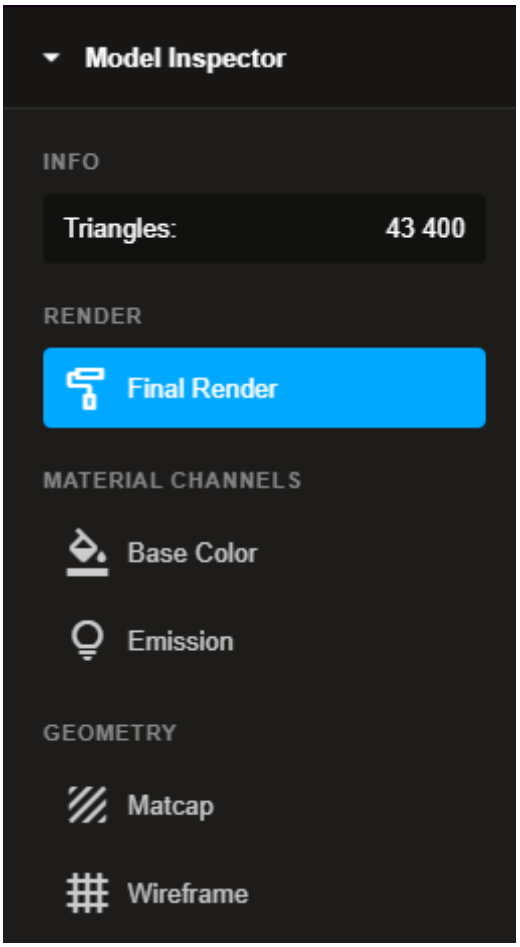

Рис. 4.4 – Панель Model Inspector для технічного аудиту 3D-моделей

Що це таке: Головним елементом цього блоку є метрика Triangles (Лічильник трикутників/полігонів). Цей показник відображає точну сумарну кількість елементарних геометричних фігур (трикутників), з яких побудована полігональна сітка (mesh) 3D-моделі.

Для чого потрібно: В індустрії комп'ютерної графіки, особливо у сфері розробки відеоігор (GameDev), симуляцій та створення доповненої реальності (AR/VR), абсолютною вимогою до 3D-моделей є оптимізація. Сучасні ігрові рушії мають жорсткі апаратні обмеження на використання ресурсів графічного

процесора (GPU). Наявність лічильника полігонів безпосередньо в інтерфейсі платформи дозволяє роботодавцю миттєво оцінити рівень технічної компетенції фахівця. Він наочно доводить, чи володіє кандидат навичками «ретопології» (створення правильної та легкої сітки), повністю позбавляючи рекрутера необхідності завантажувати файл і відкривати його у сторонніх десктопних редакторах (Blender чи Maya).

Функціонал та технічна реалізація: З інженерної точки зору, підрахунок трикутників не є статичним параметром у базі даних (Firestore). Це динамічна метрика, яка обчислюється клієнтським пристроєм у режимі реального часу під час ініціалізації бінарного файлу. Алгоритм виконує рекурсивний обхід усіх дочірніх вузлів моделі (traverse()). Для кожної знайденої сітки (isMesh) аналізується структура буфера геометрії. Якщо геометрія індексована, кількість обчислюється як відношення geometry.index.count / 3. Сумарне значення округлюється і передається у реактивну змінну Vue 3, що ініціює миттєве оновлення значення на панелі інспектора зі зручним візуальним розділенням розрядів.

1) Канал Final Render (Рендеринг)

Рис. 4.5 – Відображення 3D-моделі у режимі Final Render

Що це таке: Режим відображення фінального результату тривимірної сцени, який активує повний цикл фізично коректного рендерингу (PBR — Physically Based Rendering) [17] у середовищі браузера.

Для чого потрібно: Цей режим слугує для демонстрації автентичного художнього та інженерного замислу автора в його повному обсязі. Він дозволяє рекрутеру або артдиректору комплексно оцінити кінцеву якість проєкту: взаємодію матеріалів із джерелами світла, реалістичність відблисків, налаштування шорсткості, металевості, а також коректність відображення карт запікання тіней у WebGL-просторі.

Панель інспектора дозволяє детально декомпозувати складні PBR-матеріали на окремі складові канали для проведення глибокого інженерного аналізу текстур.

2) Канал Base Color (Базовий колір)

Рис. 4.6 – Відображення каналу базового кольору Base Color

Що це таке: Діагностичний режим ізоляції дифузного колірного каналу матеріалу (карти кольору), який повністю відключає вплив зовнішнього та направленого освітлення, шорсткості, металевості та карт рельєфу (нормалей).

Для чого потрібно: В індустрії тривимірної графіки поширені випадки, коли недоліки або низька якість розрізнення текстур маскуються інтенсивними

віртуальними джерелами світла чи ефектами постобробки. Режим «Base Color» дозволяє рекрутерам об'єктивно оцінити безпосередню роботу художників з текстуруванням: чистоту та правильність накладання UV-розгортки, деталізацію дифузної карти, відсутність розтягувань пікселів та візуальних швів на стиках геометрії.

Функціонал та технічна реалізація: При натисканні кнопки вибору каналу запускається функція обходу мешів, яка тимчасово підміняє поточний матеріал на спрощений рушійний клас THREE.MeshBasicMaterial. У його властивість map передається виключно дифузна карта кольору оригінального матеріалу (origMat.map). Оскільки цей клас матеріалу повністю ігнорує будь-яке динамічне освітлення сцени, відеокарта рендерить плоскі не затінені пікселі, забезпечуючи стовідсоткову ізоляцію колірного каналу.

3) Канал Emission (Світіння)

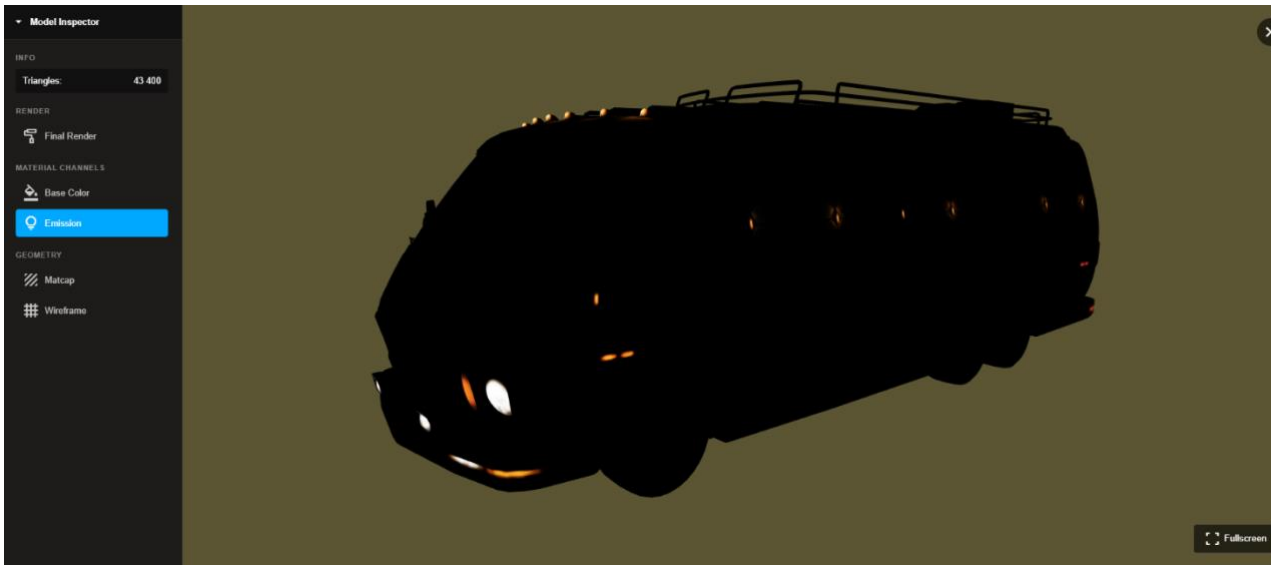

Рис. 4.7 – Відображення каналу світіння Emission

Що це таке: Спеціалізований висококонтрастний режим візуалізації, який ізолює карту самосвітіння матеріалу (Emissive map), забарвлюючи всю решту неактивної геометрії моделі у суцільний чорний колір.

Для чого потрібно: Цей канал необхідний для верифікації та перевірки коректності експорту масок світіння для інтерактивних елементів (наприклад,

фар транспортних засобів, неонових маркерів, інтерфейсів моніторів або магічних ефектів). Рекрутер може миттєво переконатися, що маска світіння не має паразитних розмить і накладена виключно на цільові полігони.

Функціонал та технічна реалізація: За аналогією з колірним каналом, логіка компонента замінює фізичний матеріал на THREE.MeshBasicMaterial. Проте у властивість map передається виключно оригінальна текстурна карта світіння (origMat.emissiveMap), а базовий колір об'єкта примусово встановлюється в абсолютно чорний колір за допомогою конструктора new THREE.Color(0x000000). Це дозволяє повністю приховати неосвітлену форму об'єкта, залишаючи видимими лише світні пікселі у WebGL-контексті.

4) Канал Matcap (Material Capture)

Рис. 4.8 – Відображення геометрії у діагностичному режимі Matcap

Що це таке: Це діагностичний режим накладання на об'єкт спеціальної статичної сферичної текстури, яка програмно імітує ідеальне студійне освітлення та фізичний матеріал (найчастіше матовий пластик або метал) незалежно від положення камери.

Для чого потрібно: Оскільки в цьому режимі повністю ігноруються оригінальні кольори, текстури та вплив віртуального світла, він ідеально підходить для перевірки якості самої поверхні (shading). Дозволяє технічним

спеціалістам (Tech Artists) легко виявляти артефакти згладжування (smooth groups), небажані вм'ятини, «потягнутості» геометрії або порушення нормалей, які могли б бути приховані під строкатою текстурою базового кольору.

Функціонал та технічна реалізація: З інженерної точки зору, алгоритм тимчасово замінює матеріали обраної моделі на спеціалізований клас THREE.MeshMatcapMaterial. У властивість matcap цього матеріалу передається заздалегідь завантажена клієнтом високоякісна сферична текстура (customMatcap), що зберігається на сервері. Для збереження фізично коректного відображення матеріалу, базовий колір примусово встановлюється у білий.

5) Канал Wireframe (Сітка)

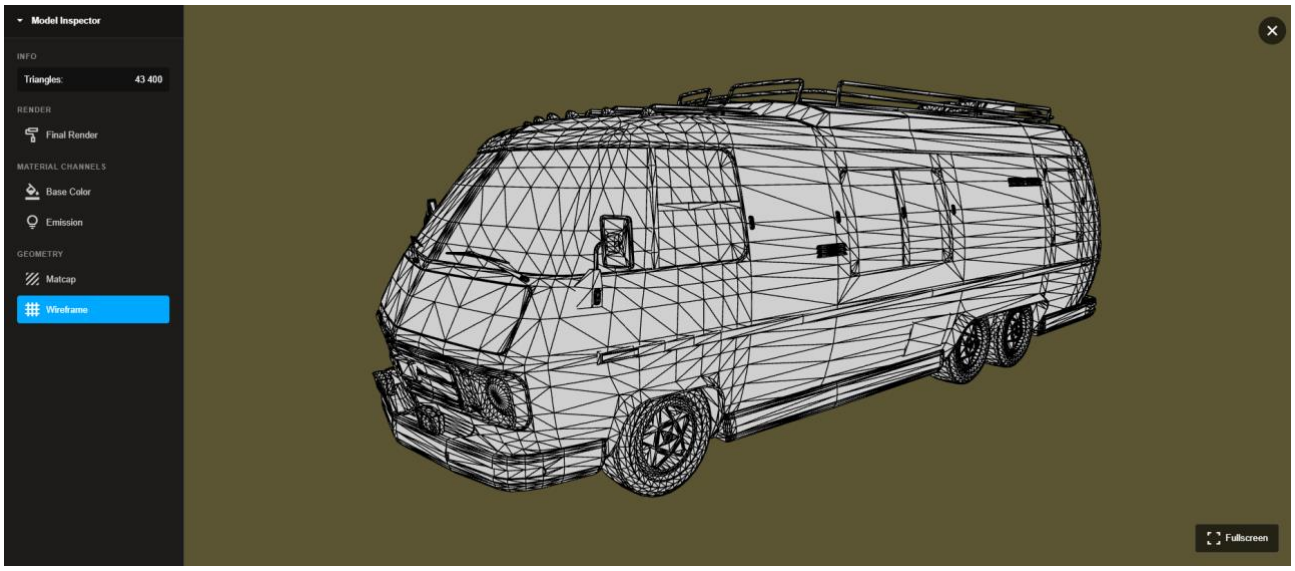

Рис. 4.9 – Відображення топологічної сітки у режимі Wireframe

Що це таке: Це режим відображення полігональної сітки об'єкта у вигляді суцільного лінійного каркаса (Wireframe), який накладається безпосередньо поверх суцільної геометрії моделі.

Для чого потрібно: Є найважливішим інструментом рекрутера для оцінки якості топології (науки про правильну побудову 3D-сітки). Дозволяє перевірити правильність побудови циклів ребер (edge loops), рівномірність щільності сітки, а також виявити наявність проблемних багатокутників — n-гонів (полігонів з

п'ятьма і більше кутами), які можуть викликати критичні помилки під час анімації персонажа.

Функціонал та технічна реалізація: Система не застосовує стандартну зміну властивості wireframe: true у базовому матеріалі (що часто призводить до візуальних артефактів — z-fighting), а використовує просунутий підхід. Створюється окрема геометрична сутність THREE.WireframeGeometry, яка накладається поверх моделі за допомогою об'єкта THREE.LineSegments з чорним кольором ліній. Базовому матеріалу моделі при цьому задається зміщення полігонів polygonOffset та polygonOffsetFactor: 1, що дозволяє рендерити чорний каркас ідеально рівно поверх сірого матеріалу бази без візуального миготіння пікселів.

4.4.1.2 Інтерактивна 2D-галерея (Фото-галерея). Окрім роботи зі складною тривимірною графікою, розроблена вебплатформа повноцінно підтримує традиційні формати цифрового портфоліо, забезпечуючи зручний перегляд двовимірних матеріалів.

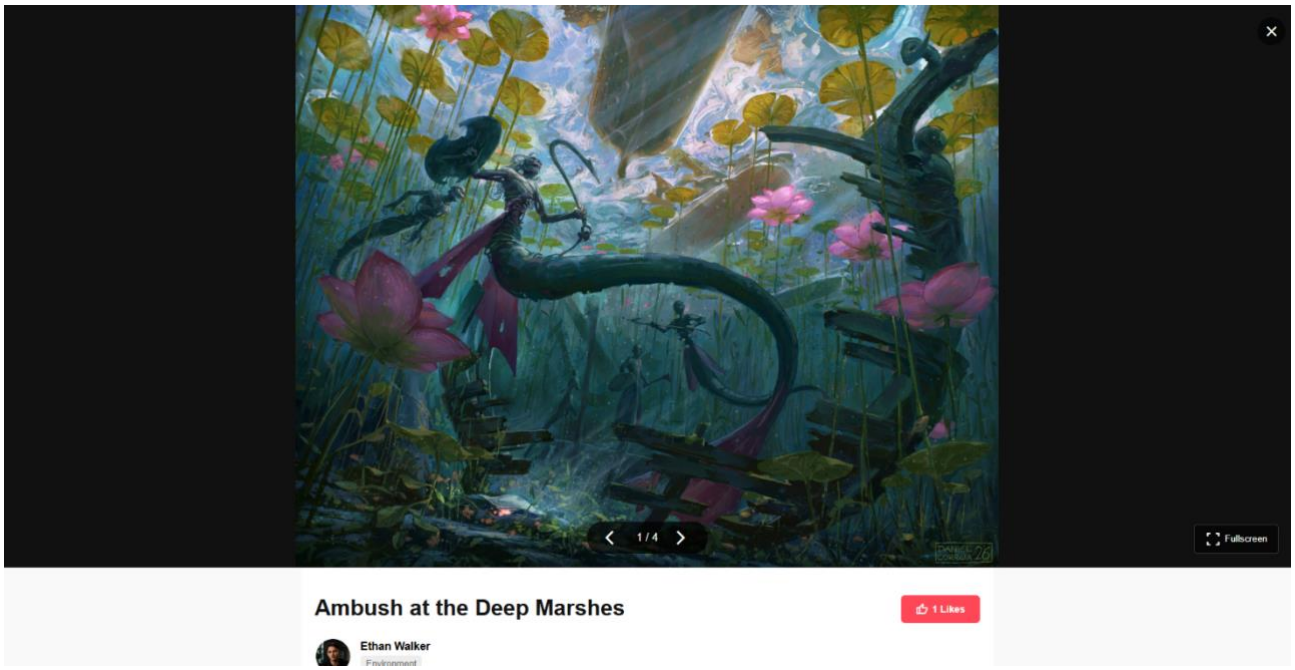

Рис. 4.10 – Модуль інтерактивної 2D-галереї

Що це таке: Спеціалізований клієнтський UI-компонент перегляду статичних зображень високої роздільної здатності із вбудованою підтримкою множинного завантаження файлів (слайдер/карусель).

Для чого потрібно: Цей модуль є ключовим для UI/UX дизайнерів, 2D-ілюстраторів та концепт-артистів. Він дозволяє авторам логічно об'єднувати серію пов'язаних зображень (наприклад, екрани мобільного додатку, різні ракурси одного персонажа, або етапи створення арту від скетчу до фінального рендеру) в межах єдиного проєкту. Такий підхід запобігає засміченню загальної стрічки платформи однотипними постами і структурує подачу матеріалу для рекрутера.

Функціонал та технічна реалізація: З інженерної точки зору, система перевіряє структуру документа проєкту в базі даних. Якщо масив посилань на медіафайли (fileUrls) містить більше одного елемента, Vue-компонент замість звичайного тегу <img> ініціалізує режим інтерактивної галереї. Створюється реактивна змінна стану galleryIndex, яка відстежує поточне зображення. У нижній частині екрана генерується напівпрозорий блок навігації з кнопками перемикання («вперед/назад»). При натисканні на ці кнопки викликаються інкапсульовані функції (prevPhoto, nextPhoto), які циклічно змінюють індекс, що призводить до миттєвої реактивної підміни атрибута src у зображенні.

4.4.1.3 Модуль відеоплеєра із захистом авторського права. Для спеціалістів із моушн-дизайну, візуальних ефектів (VFX) та аніматорів реалізовано окрему підсистему відтворення динамічного контенту.

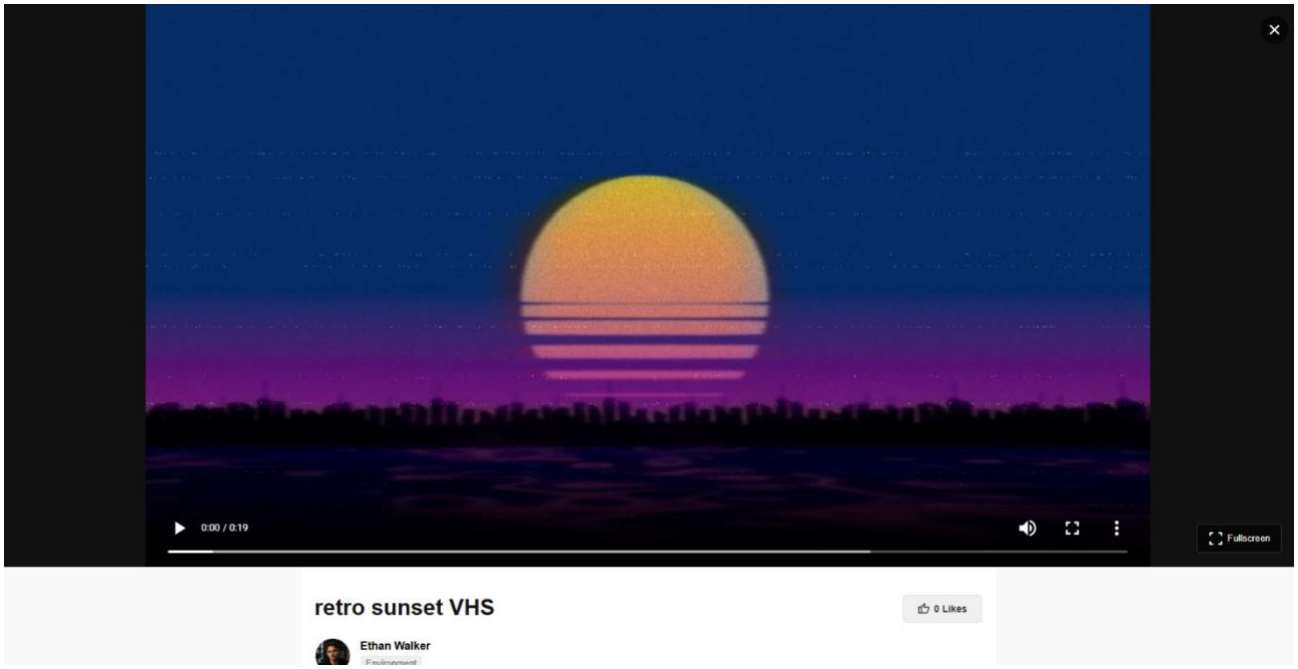

Рис. 4.11 – Вбудований відеоплеєр

Що це таке: Вбудований безпосередньо у середовище вебплатформи нативний HTML5-відеоплеєр для відтворення анімацій, симуляцій та відеовізиток (Showreels).

Для чого потрібно: Інтеграція власного плеєра забезпечує плавне відтворення форматів .mp4 та .webm безпосередньо у браузері. Це позбавляє фахівця необхідності завантажувати свої роботи на сторонні відеохостинги (такі як YouTube або Vimeo) і вставляти посилання (iframe), що дозволяє утримувати увагу рекрутера виключно всередині професійної платформи, не відволікаючи його на сторонні рекомендації чи рекламу.

Функціонал та технічна реалізація: Відео відтворюється за допомогою стандартного HTML-тегу <video> з активованим атрибутом controls, що надає користувачеві доступ до керування відтворенням (пауза, гучність, таймлайн). Додатково плеєр підтримує завантаження динамічних обкладинок-прев'ю (через атрибут poster), які відображаються до початку відтворення. Важливим інженерним рішенням є програмна реалізація механізмів захисту інтелектуальної власності. Якщо автор при створенні проєкту зняв прапорець «Дозволити завантаження» (змінна allowDownload дорівнює false), компонент динамічно

застосовує до відеоплеєра атрибут controlsList="nodownload", який повністю приховує системну кнопку скачування у плеєрі. Додатково, на рівні обробки подій DOM-дерева застосовується Vue-директива @contextmenu.prevent. Це жорстко блокує виклик системного контекстного меню браузера при натисканні правою кнопкою миші на відео, ефективно унеможливлюючи застосування функції «Зберегти відео як...» і захищаючи комерційний контент автора від несанкціонованого копіювання.

4.4.2 Функціональні елементи взаємодії та реалізація

комунікаційного віджета. Важливою складовою робочого вікна системи є інтерфейс персонального профілю користувача. Профіль фахівця відображає його спеціалізацію, розширену біографію, блоки з тегами компетенцій (Skills) та послуг (Services). Тут також розміщено стрічку активності (Activity Feed) для публікації проміжних етапів розробки (WIP) та, в самому низу, інтерактивну галерею з проєктами.

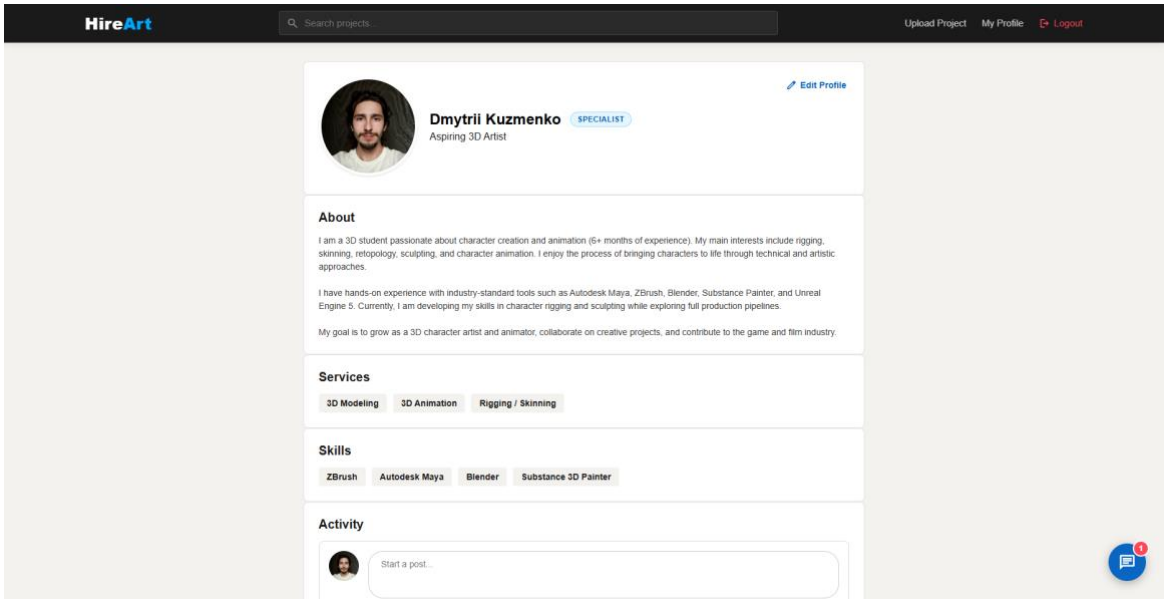

Рис. 4.12 – Профіль фахівця

Для забезпечення ефективного нетворкінгу в систему інтегровано комплексну підсистему приватних повідомлень. Коли рекрутер відвідує профіль

фахівця і приймає рішення про співпрацю, він може скористатися контактною формою для відправки пропозиції з обов'язковим зазначенням теми (Subject).

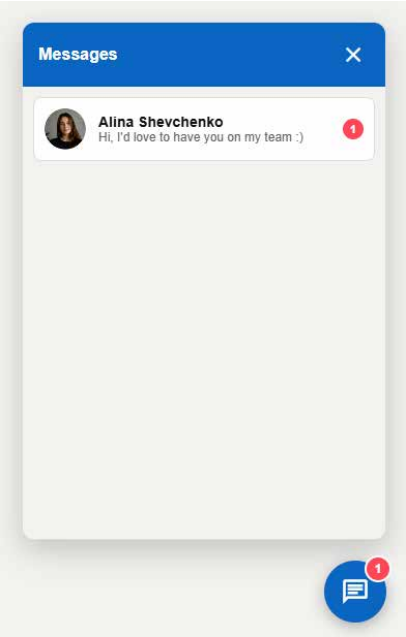

Рис. 4.13 – Вбудований чат-віджет

Чат-віджет відображає список активних діалогів із червоними індикаторами непрочитаних повідомлень. Вікно відкритого чату підтримує хронологічне сортування та миттєво оновлює інтерфейс при надходженні нових відповідей від співрозмовника. Це досягається завдяки real-time слухачам бази даних (onSnapshot), які підтримують постійне WebSocket-з'єднання з сервером Firestore, забезпечуючи зручне, сучасне та швидке середовище для укладання професійних угод прямо всередині платформи.

Висновки до розділу 4

У четвертому розділі описано процеси розгортання, конфігурування та експлуатації розробленої вебплатформи. Визначено вимоги до операційного оточення (Node.js) та описано алгоритм деплою оптимізованої статичної збірки на віддалений хмарний сервер Firebase Hosting із використанням CDN. Успішно формалізовано децентралізовану систему адміністрування з рольовим доступом

для авторів та рекрутерів. Візуальна демонстрація підтвердила безперебійне функціонування ключових модулів: глобальної стрічки проєктів, підсистеми комунікації та інноваційного 3D-в'ювера. Зокрема, підтверджено ефективність панелі «Model Inspector» для технічного аудиту тривимірної графіки (підрахунок полігонів, ізоляція каналів PBR, режими Matcap/Wireframe) та механізмів захисту авторського права у вбудованому відеоплеєрі.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було успішно вирішено актуальне інженерне завдання — спроєктовано та розроблено програмне забезпечення інтерактивної вебплатформи для створення та управління портфоліо фахівців креативних індустрій. За підсумками проведеної роботи можна зробити наступні висновки:

1. У першому розділі проведено детальний системний аналіз процесу створення та управління портфоліо. Дослідження існуючих аналогів (LinkedIn, Sketchfab, ArtStation) з інженерної точки зору виявило потребу у створенні єдиної платформи, що поєднує високопродуктивний 3D-рендеринг та інструменти нетворкінгу. За допомогою методології UML (діаграми прецедентів, послідовності та активності) формалізовано бізнес-процеси та розподілено ролі між акторами «Фахівець» та «Рекрутер». На основі отриманих моделей сформовано чіткий перелік функціональних, нефункціональних (продуктивність, безпека, юзабіліті) та технічних вимог.

2. У другому розділі здійснено проєктування інформаційного забезпечення розроблюваної платформи. Побудовано логічну ER-модель даних, що описує ключові сутності та зв'язки між ними. Інженерно обґрунтовано архітектурне рішення щодо відмови від реляційних баз даних на користь документо-орієнтованої (NoSQL) СУБД Firebase Cloud Firestore. Це забезпечило гнучкість (schemaless) структури мультимедійних проєктів, надшвидке читання за рахунок денормалізації колекцій та нативну підтримку WebSocket-синхронізації для роботи чатів. Окремо визначено політики безпеки Firestore Security Rules для криптографічного захисту конфіденційних даних.

3. У третьому розділі проведено об’єктно-орієнтоване проєктування та безпосередню реалізацію програмного забезпечення. За допомогою UML-моделювання визначено внутрішню архітектуру SPA-додатка, виділено

абстракції даних та реалізовано поліморфізм для обробки різних типів контенту (3D, 2D, Відео). Обґрунтовано вибір сучасного технологічного стеку: фреймворку Vue.js 3, системи збірки Vite, графічної бібліотеки Three.js та хмарної інфраструктури Firebase (BaaS). Описано логіку ключових програмних рішень, зокрема ініціалізацію WebGL-контексту для безперебійного апаратного рендерингу моделей у браузері.

4. У четвертому розділі здійснено підготовку розробленої вебплатформи до практичної експлуатації та перевірено її працездатність. Успішно налаштовано середовище виконання (Node.js), виконано оптимізовану збірку вихідного коду та проведено розгортання продукту на віддаленому сервері Firebase Hosting із підтримкою глобальної CDN. Візуальна демонстрація підтвердила коректну роботу ключових модулів: глобальної стрічки, підсистеми комунікації у реальному часі та мультимедійних в'юверів. Зокрема, успішно протестовано роботу унікального діагностичного модуля «Model Inspector», який забезпечує глибокий технічний аудит 3D-моделей (підрахунок полігонів, ізоляція каналів PBR-матеріалів, режими Matcap та Wireframe), а також механізми захисту авторського права у відеоплеєрі.

Загалом, розроблена інформаційна система повністю відповідає початковим архітектурним вимогам, є масштабованою, стійкою до навантажень та готовою до практичного впровадження й повноцінної експлуатації цільовою аудиторією.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Галузевий звіт про стан цифрового рекрутингу та використання електронних портфоліо в IT. URL: https://www.daxx.com/blog/development-trends/tech-recruitment-trends (дата звернення: 22.05.2026).

2. Офіційний сайт професійної мережі LinkedIn. URL: https://www.linkedin.com/ (дата звернення: 22.05.2026).

3. Sketchfab – 3D models platform. URL: https://sketchfab.com/ (дата звернення: 22.05.2026).

4. ArtStation – Showcase platform for games, film, media & entertainment. URL:https://www.artstation.com/ (дата звернення: 22.05.2026).

5. OMG Unified Modeling Language (OMG UML). Version 2.5.1. URL:https://www.omg.org/spec/UML/ (дата звернення: 22.05.2026).

6. Firebase Cloud Firestore Documentation. URL:https://firebase.google.com/docs/firestore (дата звернення: 22.05.2026).

7. BSON (Binary JSON) Specification. URL: https://bsonspec.org/ (дата звернення: 22.05.2026).

8. The WebSocket API (WebSockets) – MDN Web Docs. URL:https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 22.05.2026).

9. Firebase Authentication Documentation. URL: https://firebase.google.com/docs/auth (дата звернення: 22.05.2026).

10. Firebase Cloud Storage Documentation. URL:https://firebase.google.com/docs/storage (дата звернення: 22.05.2026).

11. Firebase Security Rules Documentation. URL: https://firebase.google.com/docs/rules (дата звернення: 22.05.2026).

12. SPA (Single-page application) Concepts – MDN Web Docs. URL:https://developer.mozilla.org/en-US/docs/Glossary/SPA (дата звернення: 22.05.2026).

13. Vue.js – The Progressive JavaScript Framework. URL: https://vuejs.org/ (дата звернення: 22.05.2026).

14. Vite – Next Generation Frontend Tooling. URL: https://vitejs.dev/ (дата звернення: 22.05.2026).

15. WebGL API – MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API (дата звернення: 22.05.2026).

16. Three.js – JavaScript 3D Library. URL: https://threejs.org/ (дата звернення: 22.05.2026).

17. Basic Theory of Physically-Based Rendering (PBR). URL:https://marmoset.com/guides/basic-theory-of-physically-based-rendering/ (дата звернення: 22.05.2026).

18. Node.js Official Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 22.05.2026).

19. npm Docs. URL: https://docs.npmjs.com/ (дата звернення: 22.05.2026).

20. Firebase Hosting Documentation. URL: https://firebase.google.com/docs/hosting (дата звернення: 22.05.2026).

21. Google Lighthouse Performance Audit. URL:https://developer.chrome.com/docs/lighthouse/overview/ (дата звернення: 22.05.2026).

ДОДАТОК А

Лістинг програмного коду компонента 3D-в'ювера (Three.js + Vue.js)

```
<script setup>
import { ref, onMounted, onBeforeUnmount, watch } from 'vue';
import * as THREE from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js';
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js';

const props = defineProps({
  modelUrl: { type: String, required: true },
  backgroundColor: { type: String, default: '#2a2a2a' }
});

const canvasContainer = ref(null);
let scene, camera, renderer, controls, animationId;

const initThreeJS = () => {
  scene = new THREE.Scene();
  scene.background = new THREE.Color(props.backgroundColor);
  camera = new THREE.PerspectiveCamera(45, canvasContainer.value.clientWidth /
canvasContainer.value.clientHeight, 0.1, 1000);
  camera.position.set(0, 2, 5);
  renderer = new THREE.WebGLRenderer({ antialias: true });
  renderer.setSize(canvasContainer.value.clientWidth,
canvasContainer.value.clientHeight);
```

```
renderer.setPixelRatio(window.devicePixelRatio);
renderer.outputEncoding = THREE.sRGBEncoding;
canvasContainer.value.appendChild(renderer.domElement);

const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
scene.add(ambientLight);
const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);
directionalLight.position.set(5, 10, 7);
scene.add(directionalLight);
controls = new OrbitControls(camera, renderer.domElement);
controls.enableDamping = true;
const loader = new GLTFLoader();
loader.load(props.modelUrl, (gltf) => {
  const model = gltf.scene;
  // Автоматичне центрування моделі
  const box = new THREE.Box3().setFromObject(model);
  const center = box.getCenter(new THREE.Vector3());
  model.position.sub(center);
  scene.add(model);
}, undefined, (error) => {
  console.error('Помилка завантаження моделі:', error);
});

const animate = () => {
  animationId = requestAnimationFrame(animate);
  controls.update();
```

```
    renderer.render(scene, camera);
  };
  animate();
};
onMounted(() => {
  if (props.modelUrl) initThreeJS();
});
onBeforeUnmount(() => {
  cancelAnimationFrame(animationId);
  if (renderer) renderer.dispose();
});
</script>
<template>
  <div ref="canvasContainer" class="threejs-canvas-container"></div>
</template>
<style scoped>
.threejs-canvas-container {
  width: 100%;
  height: 100%;
  min-height: 400px;
}
</style>
```

ДОДАТОК Б

Лістинг конфігурації правил безпеки бази даних (Firestore Security Rules)

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Перевірка авторизації
    function isAuthenticated() {
      return request.auth != null;
    }

    // Перевірка чи користувач є власником документа
    function isOwner(userId) {
      return isAuthenticated() && request.auth.uid == userId;
    }

    // Колекція Users: читати можуть усі, редагувати - лише власник профілю
    match /Users/{userId} {
      allow read: if true;
      allow create, update, delete: if isOwner(userId);
    }

    // Колекція Projects: читати можуть усі, редагувати - лише автор
```

```
    match /Projects/{projectId} {
      allow read: if true;
      allow create: if isAuthenticated() && request.resource.data.authorId ==
request.auth.uid;
      allow update, delete: if isOwner(resource.data.authorId);
    }

    // Колекція Messages: читати та писати можуть лише відправник або
отримувач
    match /Messages/{messageId} {
      allow read, write: if isAuthenticated() &&
        (request.auth.uid == resource.data.senderId || request.auth.uid ==
resource.data.receiverId ||
         request.auth.uid == request.resource.data.senderId);
    }
  }
}
```

ДОДАТОК В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Кузьменко Дмитрій Олегович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення для створення та управління портфоліо фахівця» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
   - [ ] інструменти генеративного ШІ не використовувалися.
   - [✓] інструменти ШІ ChatGPT, Gemini були використані для:
     - [✓] перекладу іншомовних джерел;
     - [✓] стилістичного редагування та перевірки граматики;
     - [✓] допомоги у написанні/відлагодженні програмного коду;
     - [] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___»_____ 2026 р. _____ **Дмитрій КУЗЬМЕНКО**