ДОДАТОК В

Специфікація програмного продукту

**СПЕЦИФІКАЦІЯ ПЗ**

До програмної платформи для продажу та перегляду 3D-моделей з використанням WebGL та інтерактивного 3D-перегляду

Виконав:

Карпов Денис Олександрович

**1. Вступ**

**1.1 Огляд продукту**

Розроблювана система — це веб-платформа для завантаження, публікації, перегляду та покупки 3D-моделей у форматі .glb. Основна функція продукту — забезпечити простий і зручний спосіб взаємодії з тривимірним контентом без встановлення додаткового програмного забезпечення. Інтерфейс платформи дозволяє користувачам переглядати інтерактивні моделі безпосередньо в браузері, а також завантажувати власні файли, продавати їх і купувати роботи інших авторів.

Система орієнтована на авторів 3D-контенту (дизайнерів, художників, розробників) і кінцевих користувачів, які зацікавлені в придбанні якісних моделей для власних проєктів. Продукт реалізовано як сучасний вебзастосунок із використанням технологій Next.js, Three.js, Supabase, Stripe та інших рішень, що забезпечують масштабованість, стабільну роботу й привабливий користувацький досвід.

**1.2 Область застосування системи**

Розроблювана система є веб-платформою, що дозволяє користувачам публікувати власні 3D-моделі у форматі .glb, переглядати їх за допомогою інтерактивного візуалізатора, а також здійснювати покупки та завантаження моделей. Платформа працює повністю в браузері, без необхідності встановлення додаткового ПЗ.

Цільовою аудиторією платформи є 3D-художники, дизайнери, розробники ігор та інші користувачі, які працюють з тривимірним контентом. Основна увага приділяється зручності використання, доступності інтерфейсу та швидкодії при роботі з важкими 3D-файлами.

**1.3 Межі та обмеження системи**

Система створена виключно як веб-застосунок і не передбачає мобільного додатка або офлайн-режиму. Для коректної роботи потрібен сучасний браузер із підтримкою WebGL і JavaScript. Усі основні дії — авторизація, завантаження моделей, здійснення покупок — виконуються через браузер.

Платформа підтримує лише формат .glb як основний стандарт для тривимірних моделей. Завантаження файлів великого розміру обмежене технічними можливостями Supabase Storage. Покупки здійснюються винятково через Stripe. Анонімні користувачі мають доступ лише до перегляду моделей та пошуку, тоді як авторизовані можуть завантажувати власні роботи та купувати інші.

**1.4 Визначення, терміни та скорочення**

У цьому документі використовуються наступні поняття:

* 3D-модель — цифровий тривимірний об’єкт, представлений у форматі .glb, який можна переглядати в браузері.
* Гість — неавторизований користувач платформи, який має доступ до перегляду, пошуку та ознайомлення з моделями.
* Автор — зареєстрований користувач, який може завантажувати, публікувати та продавати власні моделі.
* Next.js — фреймворк для побудови фронтенду на базі React.
* Three.js — JavaScript-бібліотека для відтворення 3D-графіки у браузері.
* Supabase — хмарна платформа, що надає базу даних PostgreSQL, файлове сховище та API.
* Stripe — стороння платіжна система для обробки транзакцій.
* Auth.js — бібліотека для реалізації автентифікації та авторизації користувачів.

**2. Загальний опис**

**2.1 Перспективи продукту**

Розроблювана система — це веб-платформа для завантаження, публікації, перегляду та покупки 3D-моделей у форматі .glb. Основна функція продукту — забезпечити простий і зручний спосіб взаємодії з тривимірним контентом без встановлення додаткового програмного забезпечення. Інтерфейс платформи дозволяє користувачам переглядати інтерактивні моделі безпосередньо в браузері, а також завантажувати власні файли, продавати їх і купувати роботи інших авторів.

Система орієнтована на авторів 3D-контенту (дизайнерів, художників, розробників) і кінцевих користувачів, які зацікавлені в придбанні якісних моделей для власних проєктів. Продукт реалізовано як сучасний вебзастосунок із використанням технологій Next.js, Three.js, Supabase, Stripe та інших рішень, що забезпечують масштабованість, стабільну роботу й привабливий користувацький досвід.

**2.2 Функціональні можливості**

Система надає користувачам можливість публікувати, переглядати, купувати та завантажувати 3D-моделі у форматі .glb. Основні функції включають авторизацію, управління обліковим записом, завантаження 3D-файлів до бази даних із супровідною інформацією, а також здійснення онлайн-оплати та отримання доступу до куплених моделей. Перегляд моделей реалізовано через інтегрований 3D-програвач на основі Three.js, що дозволяє взаємодіяти з моделлю прямо в браузері.

Користувачі можуть здійснювати пошук моделей за ключовими словами, фільтрувати їх за тегами та ціновими параметрами. Інтерфейс дозволяє швидко переходити до перегляду обраної моделі та, за потреби, додати її до кошика для подальшої покупки. Усі дії, пов’язані з покупкою чи завантаженням моделей, доступні лише після авторизації.

**2.3 Характеристики користувачів**

У системі розрізняються два основні типи користувачів: гість і автор.

Гість — це неавторизований користувач, який має доступ лише до публічного функціоналу: перегляду каталогу моделей, пошуку, фільтрації та інтерактивного перегляду 3D-сцен. Авторизований користувач (автор) має додаткові права — можливість створювати обліковий запис, входити в систему, завантажувати власні моделі, керувати ними, здійснювати покупки та переглядати список придбаних моделей.

Платформа не передбачає розділення користувачів на адміністратора або модератора, що спрощує логіку доступу й зменшує складність архітектури на початковому етапі.

**2.4 Середовище функціонування**

Система реалізована як вебзастосунок і призначена для роботи у сучасних браузерах, які підтримують JavaScript, WebGL та HTML5. Коректна робота гарантована на десктопних версіях Google Chrome, Firefox, Safari та Microsoft Edge останніх версій. Мобільна оптимізація інтерфейсу є частково реалізованою, однак мобільний застосунок як окремий продукт не передбачається.

З боку сервера застосунок використовує Supabase як бекенд-платформу, що включає PostgreSQL-базу даних, хмарне сховище для моделей і API-доступ. Клієнтська частина реалізована за допомогою Next.js з використанням Tailwind CSS для стилізації та Three.js для інтерактивного перегляду моделей.

**2.5 Загальні обмеження**

Платформа працює виключно онлайн, без підтримки офлайн-доступу. Підтримується тільки один формат 3D-моделей — .glb, що обумовлено сумісністю з Three.js. Для обробки платежів використовується Stripe, і додаткові способи оплати не передбачені.

Максимальний розмір моделі, яку можна завантажити, обмежується технічними характеристиками Supabase Storage та обмеженнями браузера щодо обробки файлів. Система не передбачає редагування вже завантажених моделей або комунікації між користувачами. Вся взаємодія між клієнтом і сервером відбувається через REST API, без використання WebSocket або інших реальногочасових рішень.

**2.6 Припущення й залежності**

Розробка системи базується на припущенні, що користувачі мають стабільне інтернет-з’єднання, сучасний браузер і доступ до електронної платіжної системи. Уся взаємодія з моделями (завантаження, перегляд, покупка) здійснюється виключно через веб-інтерфейс.

Система покладається на зовнішні сервіси: Stripe для обробки платежів, Supabase для зберігання даних і Auth.js для авторизації. У разі недоступності будь-якого з цих сервісів певна функціональність платформи буде тимчасово обмежена.

**3 Конкретні вимоги**

**3.1 Інтерфейси**

3.1.1 Інтерфейс користувача

Інтерфейс користувача реалізовано як адаптивний вебінтерфейс, створений на базі фреймворку Next.js із використанням App Router. Уся взаємодія з системою здійснюється без перезавантаження сторінки, що забезпечує плавну навігацію та миттєвий відгук. Інтерфейс структуровано за принципами мінімалізму та доступності: головна сторінка містить інтерактивний каталог моделей, кожна модель має власну сторінку з візуалізацією, а зареєстровані користувачі мають доступ до особистого кабінету.

Усі ключові функції — реєстрація, вхід, пошук, завантаження моделі, перегляд 3D-сцени, оплата та скачування — виконуються через клієнтський інтерфейс. Завдяки Tailwind CSS платформа підтримує базову адаптацію під різні розміри екрану, хоча мобільна версія не є повноцінною ціллю на поточному етапі. Основна увага приділена десктопному середовищу з максимально простою структурою взаємодії.

3.1.2 Програмний інтерфейс

Програмний інтерфейс реалізований як набір REST API-роутів у межах серверної частини Next.js. Кожен запит користувача — наприклад, на реєстрацію, отримання моделі, пошук, завантаження або оплату — надсилається у вигляді HTTP-запиту до відповідного endpoint. Усі API працюють у serverless-середовищі, що дозволяє ефективно масштабувати систему та знижує затримку відповіді.

Результати запитів повертаються у форматі JSON і можуть містити як дані (модель, список, об'єкт покупки), так і службову інформацію (статус, повідомлення, помилки). Для авторизованих запитів до кожного запиту додається токен доступу, який перевіряється на сервері. Ця логіка дозволяє чітко розмежовувати доступ до публічних і приватних частин платформи.

3.1.3 Зовнішні інтерфейси

У системі реалізовано кілька зовнішніх інтеграцій через API сторонніх сервісів. Основна з них — платіжна система Stripe, яка використовується для створення Checkout-сесій і підтвердження транзакцій. Через API Stripe створюється платіжна сесія із вказанням суми, ID користувача та опису товару. Після оплати Stripe надсилає підтвердження, яке обробляється на стороні платформи.

Ще один важливий зовнішній інтерфейс — Supabase API. Його компоненти відповідають за автентифікацію користувачів (через Supabase Auth), збереження файлів моделей та прев’ю (через Supabase Storage), а також доступ до бази PostgreSQL через Drizzle ORM. Інтеграція з Auth.js дозволяє зручно керувати сесіями користувачів на клієнтській стороні без прямої роботи з токенами.

Ці зовнішні сервіси є критично важливими для роботи всієї платформи, тому взаємодія з ними організована через надійні, перевірені механізми, що відповідають сучасним стандартам безпеки (HTTPS, токенізація, CORS, Webhook-захист тощо).

3.1.4 Інтерфейс доступу до бази даних

Для взаємодії із серверною базою даних PostgreSQL використовується ORM-бібліотека Drizzle, яка дозволяє здійснювати доступ до таблиць і зв'язків між ними у типізованому середовищі TypeScript. Завдяки Drizzle розробка API стала безпечнішою — перевірки типів і схем забезпечують цілісність усіх операцій читання та запису.

База містить таблиці моделей, тегів, користувачів, покупок та прив’язок між ними. Всі CRUD-операції з моделями, покупками чи тегами здійснюються через Drizzle-запити до PostgreSQL. Вибір саме такої технології дозволив скоротити ризик помилок, покращити продуктивність і легко масштабувати структуру даних у майбутньому.

**3.2 Реєстрація та авторизація**

Вхідні дані:  
Користувач вводить email у форматі name@example.com і пароль, що відповідає мінімальній довжині (6 символів). Дані передаються через клієнтську форму до Auth.js, який виконує авторизацію через Supabase Auth.

Обробка:  
Auth.js створює або перевіряє обліковий запис у Supabase Auth. У разі успіху користувач отримує токен доступу (JWT), який зберігається в сесії. Надалі токен додається до всіх захищених запитів, що дозволяє серверу визначити ідентичність користувача.

Обробка помилок:  
Невірна пошта або пароль, повторна реєстрація з існуючим email, або втрата сесії призводять до відображення відповідного повідомлення користувачу без перезавантаження сторінки.

Результат:  
Користувач успішно входить у систему, отримує доступ до функцій завантаження моделей, покупок і кабінету.

Обмеження:  
Соціальні входи не реалізовані. Підтвердження електронної пошти не вимагається. Автоматичний вихід зі сесії реалізовано лише після очищення кешу або ручного виходу.

**3.3 Завантаження моделі**

Вхідні дані:  
Користувач заповнює форму: назва (до 100 символів), опис (до 1000), ціна (додатнє число з точністю до 2 знаків), список тегів (обирає наявні або створює нові), .glb-файл (до 100 МБ) і мініатюру (зображення .jpg/.png).

Обробка:  
Форма надсилає FormData на API /api/models/upload. API зберігає модель і мініатюру в Supabase Storage, а метадані записує в PostgreSQL через Drizzle. Slug створюється автоматично з назви та додається як унікальний шлях до моделі.

Обробка помилок:  
Невірний формат файлів, надмірна вага, відсутні обов’язкові поля або спроба створити дублікат slug — усе це викликає повідомлення про помилку. Валідація також дублюється на бекенді.

Результат:  
Модель додається до каталогу та відображається з усіма метаданими на окремій сторінці.

Обмеження:  
Формат тільки .glb. Один slug — одна модель. Файл моделі не можна змінити після публікації.

**3.4 Перегляд моделі**

Вхідні дані:  
На сторінці перегляду модель завантажується за slug, переданим у URL (/models/[slug]).

Обробка:  
API /api/models/[slug] повертає інформацію з бази про модель, включаючи посилання на файл. Файл передається до Three.js, який завантажує його в інтерактивну сцену. Користувач бачить об’єкт у просторі та може взаємодіяти з ним.

Обробка помилок:  
У разі відсутності моделі система повертає 404. Якщо виникає помилка рендерингу 3D-файлу, відображається запасне зображення або повідомлення про помилку.

Результат:  
Модель відображається в браузері з можливістю обертання, масштабування та взаємодії у вікні переглядача.

Обмеження:  
Сторінка працює лише в браузерах із підтримкою WebGL. Рендеринг може бути повільним на слабких пристроях або в мобільному середовищі.

**3.5 Пошук і фільтрація**

Вхідні дані:  
Користувач вводить текст у поле пошуку, вибирає теги та встановлює діапазон ціни.

Обробка:  
Запит надсилається через GET до /api/models?search=...&tags=...&priceMin=...&priceMax=.... Бекенд формує SQL-запит із фільтрами. Результати повертаються у вигляді масиву моделей.

Обробка помилок:  
Якщо запит некоректний або жодна модель не відповідає критеріям, користувачу виводиться повідомлення. Якщо параметри мають помилковий формат, API повертає статус 400.

Результат:  
Виводиться оновлений список моделей, що відповідають заданим параметрам.

Обмеження:  
Поєднання тегів працює за логікою AND. Ціна обмежується тільки позитивними значеннями. Пошук не враховує регістр символів.

**3.6 Купівля моделі**

Вхідні дані:  
При натисканні кнопки “Купити” передається ID моделі, ім’я користувача з токена, ціна, опис і Stripe-ключ.

Обробка:  
API викликає Stripe Checkout, створює сесію платежу та переспрямовує користувача на сторінку оплати. Після оплати Stripe переадресовує назад, а бекенд записує покупку в базу (таблиці orders і order\_items).

Обробка помилок:  
Проблеми з оплатою або мережеві збої призводять до скасування операції. Якщо користувач уже купував модель — система не дозволяє повторну покупку.

Результат:  
Успішна покупка — користувач бачить кнопку “Завантажити” та запис у своєму кабінеті.

Обмеження:  
Оплата доступна лише для зареєстрованих користувачів. Stripe — єдина платіжна система. Покупку однієї моделі не можна повторити.

**3.7 Завантаження придбаної моделі**

Вхідні дані:  
ID моделі передається через кнопку “Завантажити”. Користувач повинен бути авторизований.

Обробка:  
Система перевіряє у таблиці orders, чи дійсно дана модель була куплена користувачем. Якщо так — генерується signed URL до Supabase Storage. Посилання діє обмежений час.

Обробка помилок:  
Якщо запису немає — повертається 403. Якщо файл відсутній або помилка в URL — 404.

Результат:  
Модель завантажується на пристрій користувача через пряме посилання.

Обмеження:  
Завантаження доступне лише тому користувачеві, який здійснив покупку. Посилання не можна розшарити.

**3.8 Обробка помилок і валідація**

Вхідні дані:  
Всі вхідні дані, які надходять від користувача: поля форм, API-запити, файли.

Обробка:  
На клієнті перевіряються формати, довжина, типи файлів. На сервері ці перевірки дублюються. У разі невідповідності система повертає відповідний код (400, 403, 404, 500) разом із описом помилки.

Обробка помилок:  
Повідомлення виводяться у зрозумілому вигляді: “Файл занадто великий”, “Це поле є обов’язковим”, “Ця модель уже існує”, тощо.

Результат:  
Користувач розуміє, чому дія не виконалась. Усі дії з неприпустимими параметрами блокуються.

Обмеження:  
Всі помилки обробляються без перезавантаження сторінки. Винятки на сервері логуються й можуть бути виведені в адміністративній панелі (в майбутніх версіях).

**3.9 Основні класи та об'єкти**

Система має чітку структуру з розподілом відповідальності між об’єктами. Основними об’єктами є користувач, модель, тег, замовлення та файл. Кожен із них реалізовано як окрему сутність у базі даних із відповідними зв’язками.

Користувач має унікальний ідентифікатор і базову інформацію — електронну пошту, роль, список завантажених та придбаних моделей. Модель — це центральна одиниця системи, яка містить назву, опис, ціну, slug, прев’ю та посилання на 3D-файл. Вона пов’язана з автором (користувачем) і списком тегів. Теги реалізовано окремою таблицею, а зв’язок між моделями і тегами — через проміжну таблицю.

Замовлення — це об’єкт, що поєднує користувача з купленими моделями. Він фіксує дату, суму й набір замовлених моделей. Усі об’єкти пов’язані в систему типів (TypeScript), що дозволяє зберігати узгодженість при запитах до API, роботі з ORM і передачі даних у компоненти React.

**3.10 Нефункціональні вимоги**

3.10.1 Вимоги до продуктивності

Система повинна забезпечувати швидке завантаження всіх сторінок за стандартних умов користування. При середній швидкості інтернет-з’єднання час відповіді не повинен перевищувати 2 секунд. Інтерактивний 3D-преглядач має відображати моделі до 50 МБ без помітних затримок. Застосунок використовує serverless-архітектуру, кешування запитів і оптимізоване рендерення компонентів, що дозволяє досягати високої продуктивності без надмірного навантаження на сервер.

3.10.2 Вимоги до надійності

Система має працювати стабільно та передбачувано у звичайному режимі роботи. Усі ключові операції — такі як завантаження моделей, автентифікація, оплата — повинні супроводжуватись захистом від збоїв. У разі втрати з’єднання з базою даних або зовнішніми сервісами, користувач повинен отримати коректне повідомлення про недоступність функції. Важливі дії не мають завершуватись без підтвердження або з частковими результатами.

3.10.3 Вимоги до безпеки

Кожна дія, що змінює дані або стосується доступу до захищеної інформації, повинна виконуватись тільки після авторизації. Всі авторизовані запити супроводжуються перевіркою токена, який передається через захищене HTTPS-з'єднання. Сторонні системи, такі як Stripe, повністю беруть на себе обробку та збереження платіжних даних — платформа їх не зберігає. Доступ до приватних моделей захищений: моделі можна завантажити лише з підписаними URL, доступними лише їхнім власникам.

3.10.4 Вимоги до масштабованості

Архітектура проєкту передбачає можливість горизонтального масштабування без зміни основної логіки. Завдяки використанню Supabase як бекенд-платформи з хмарним сховищем та базою даних, Stripe для платіжних операцій та serverless API-роутів на Next.js, додавання нових функцій або збільшення навантаження не потребує повного перепроєктування. Система також передбачає легке введення нових ролей користувачів, інтеграцію адміністративного інтерфейсу або статистики без зміни основної моделі даних.