Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук,
кандидат технічних наук, доцент
_____ Ірина МАШКІНА
« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика
Тема роботи: СТВОРЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ЗАМОВЛЕННЯ ТА ВИКОНАННЯ 3D-ДРУКУ

Виконав
студент групи ІНб-1-21-4.0д
Ільїнов Кирило Миколайович
_____
(підпис)
Науковий керівник
кандидат технічних наук, доцент
ЯСКЕВИЧ В.О.
_____
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка

Факультет інформаційних технологій та математики

Кафедра комп'ютерних наук

«Затверджую»

Завідувач кафедри комп'ютерних наук,

кандидат технічних наук, доцент

_____ Ірина МАШКІНА

«____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА

«Створення вебзастосунку для замовлення та виконання 3D-друку»

Виконавець – студент групи ІНб-1-21-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Ільїнов Кирило Миколайович

1. Вихідні дані: сучасний стан і проблеми ринку 3D-друку, наукові підходи до розробки веб-платформ, актуальна потреба в зручному сервісі для замовлення послуг 3D-друку онлайн.
2. Провести аналіз сучасних аналогічних платформ.Визначити вимоги до сервісу для замовлення 3D-друку.Спроєктувати архітектуру і основні функціональні блоки вебзастосунку.Реалізувати прототип MVP-платформи з двома ролями (замовник і виконавець), підтримкою завантаження STL-файлів, фото-звітів, системи рейтингу та чату.Провести тестування роботи системи.Підготувати пояснювальну записку, графічні матеріали та додатки відповідно до вимог кафедри.
3. Пояснювальна записка: Обсяг – до 70 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.
4. Графічні матеріали: презентація.
5. Додатки: не передбачено.
6. Строк подання роботи на кафедру: «____» червня 2025 р.

Науковий керівник
кандидат технічних наук, доцент

_____ ЯСКЕВИЧ В.О.

_____ дата

Виконавець:
Ільїнов К. М.

_____ дата

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи бакалавра	до 25 листопада 2024 р.	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	до 10 грудня 2024 р.	Виконано
3	Складання змісту та календарного плану роботи	до 16 грудня 2024 р.	Виконано
4	Аналіз предметної області, огляд аналогів, формування цілей і задач	до 20 грудня 2024 р.	Виконано
5	Написання реферату та вступу	до 25 грудня 2024 р.	Виконано
6	Написання першого розділу	до 23 березня 2025 р.	Виконано
7	Написання другого розділу	до 20 квітня 2025 р.	Виконано
8	Написання третього розділу	до 15 травня 2025 р.	Виконано
9	Написання висновків	до 22 травня 2025 р.	Виконано
10	Виправлення зауважень	до 6 червня 2025 р.	Виконано
11	Створення презентації	до 22 травня 2025 р.	Виконано
12	Захист матеріалів дипломної роботи на засіданні кафедри	22 травня 2025 р.	
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	згідно графіку захистів	

«Затверджую»

Науковий керівник **Індивідуальний план склав**

Яскевич В.О. **Ільїнов К.М**

_____ _____

РЕФЕРАТ бакалаврської роботи

Кваліфікаційна робота: 60 стор., 26 мал., 1 таблицю, 30 джерел.

Мета роботи: створити просту й зручну веб-платформу для замовлення та виконання 3D-друку, яка з'єднає замовників і власників 3D-принтерів, забезпечить чесну конкуренцію, простий пошук і безпечну взаємодію.

Об'єкт дослідження: процес онлайн-замовлення послуг 3D-друку.

Предмет дослідження: програмні рішення для взаємодії замовника і виконавця через веб-платформу.

Актуальність: 3D-друк стрімко розвивається в Україні, але єдиного простого сервісу для замовлень не було. Платформа вирішує цю проблему, дає зручність користувачам та підтримує розвиток галузі.

Завдання роботи:

1. Описати сучасний стан і проблеми ринку 3D-друку.
2. Проаналізувати аналоги.
3. Визначити вимоги до сервісу.
4. Спроєктувати архітектуру і функціонал.
5. Реалізувати та протестувати робочий прототип.

Короткий зміст роботи:

У роботі показано шлях від ідеї до робочого MVP: як обирався стек (Python/Flask, Bootstrap), як будувалась база даних і логіка платформи. Описано ключові модулі — особисті кабінети, каталог оферів, форма замовлення, чат, фото-звіт, система рейтингів. Показано сценарії користування та результати тестування. Платформа проста для користувача, готова до впровадження і масштабування.

Ключові слова: 3D-друк, веб-платформа, онлайн-замовлення, Flask, Python, чат, рейтинг, MVP

ЗМІСТ

ВСТУП

У сучасному світі 3D-друк стає все важливішим. Його використовують у військовому виробництві, машинобудуванні, створенні одягу, побутової техніки та для особистих потреб. Багато людей потребують унікальні деталі під свої потреби чи потреби бізнесу, а власники 3D-принтерів шукають замовлення для своїх пристроїв.

Я створив веб-платформу для онлайн-замовлень 3D-друку, яка може стати центром зв’язку між тими, хто має обладнання, і тими, хто потребує друкованих виробів. Тут можна швидко знайти пропозицію від виконавця, порівняти ціни, переглянути приклади робіт, замовити друк та обмінюватися повідомленнями під кожною пропозицією.

Платформа проста і зрозуміла навіть новачку. Її головна ідея — створити чесне конкурентне середовище, де кожен виконавець легко отримує клієнтів без зайвих витрат на рекламу, а замовник — отримує найкращу пропозицію за оптимальною ціною.

Мета роботи:Розробка та впровадження веб-платформи, що з’єднує замовників і виконавців 3D-друку та створює для них прозорий, конкурентний і масштабований онлайн-сервіс. Платформа має забезпечити швидкий пошук, об’єктивний рейтинг, можливість перегляду фото-робіт, обмін повідомленнями та легко розширюватися на суміжні технології (лазерне різання, ЧПК, лиття тощо), формуючи сучасну цифрову інфраструктуру для українського дрібносерійного виробництва.

Завдання дослідження:

1. Проаналізувати сучасні способи взаємодії між замовниками і виконавцями, виявити їх недоліки;

2. Сформулювати функціональні та нефункціональні вимоги до платформи;
3. Спроєктувати архітектуру веб-застосунку та базу даних;
4. Реалізувати MVP з реєстрацією двох ролей, каталогом пропозицій, обміном повідомленнями та можливістю оцінювання/відгуків;
5. Протестувати прототип і оцінити його можливості для масштабування.

Об’єкт дослідження: Процес онлайн-замовлення послуг 3D-друку.

Предмет дослідження: Програмно-алгоритмічні рішення, що забезпечують взаємодію замовника і виконавця через веб-платформу.

Методи: Аналіз предметної області, структурне і об’єктно-орієнтоване проєктування, прототипування, тестування.

Практична цінність: Створення робочого MVP-маркетплейса, який можна одразу розгорнути у виробничих умовах та легко адаптувати для суміжних технологій.

Структура роботи: Робота складається з вступу, трьох розділів (аналіз предметної області, проектування рішення, реалізація і тестування), висновків, списку використаних джерел та додатків.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ринку 3D-друку

3D-друк, або адитивне виробництво, уже став звичайною технологією у різних сферах: від машинобудування та військової галузі до медицини, освіти й індивідуального прототипування. За оцінками, у 2024 році обсяг світового ринку 3D-друку перевищує 17 мільярдів доларів США, і цей показник продовжує зростати щороку на понад 15% [1]. Головними напрямами використання 3D-друку є виготовлення унікальних деталей на замовлення, швидке прототипування нових виробів, створення індивідуальних медичних імплантатів, а також виробництво складових для дронів та інших технічних пристроїв. Сьогодні значна частина компаній та приватних осіб, які займаються розробкою або ремонтом складної техніки, розглядають 3D-друк як основний інструмент для вирішення нетипових виробничих задач.

Особливу динаміку розвиток цієї галузі отримав в Україні з початку повномасштабної війни. Поява спільноти волонтерів "Друк Армія", до якої долучилися сотні власників 3D-принтерів, дозволила виготовити й передати на фронт мільйони надрукованих деталей та пристроїв, зокрема комплектуючі для безпілотників, захисні корпуси, пристрої для евакуації поранених тощо [2; 3]. Цей феномен не лише посилив обороноздатність країни, а й дав новий поштовх для формування спільноти професіоналів і майстрів, які оволоділи сучасними технологіями та здатні розвивати цю галузь у мирний час.

Водночас структура українського ринку 3D-друку поки що залишається фрагментованою. Переважна більшість замовлень обговорюється в закритих групах у соцмережах чи месенджерах, де замовнику складно швидко знайти виконавця, порівняти ціни чи оцінити якість виконаних робіт. Деякі друкарі створюють власні сайти або сторінки для пошуку клієнтів, однак це вимагає

значних зусиль для просування та додаткових витрат на рекламу, що підвищує кінцеву вартість послуги для замовника. Багато замовлень досі здійснюються "по знайомству" або через рекомендації, а питання стандартизації ціноутворення й об'єктивної оцінки якості роботи залишається відкритим.

Для порівняння, у світі вже існують великі платформи для обміну 3D-моделями — наприклад, Printables.com, яка об'єднує мільйони користувачів і щомісяця поповнюється тисячами нових моделей [4]. Однак ці ресурси, як правило, орієнтовані на поширення STL-файлів і не забезпечують замовлення друку з вибором виконавця та подальшим контролем якості. Така ситуація створює "вікно можливостей" для появи саме сервісу, який дозволяє об'єднати потенціал українських друкарів і зручність для замовників, скорочуючи витрати часу, забезпечуючи прозорість цін і якісне обслуговування.

Отже, актуальність створення національної платформи для онлайн-замовлення послуг 3D-друку обумовлена не лише зростанням технологічного рівня та збільшенням попиту на індивідуальні вироби, а й конкретною потребою в прозорому, швидкому й конкурентному ринку. Реалізація такого проєкту здатна підвищити ефективність української спільноти виробників, стимулювати розвиток суміжних галузей і закласти основу для цифрової трансформації виробництва на новому рівні [1–4].

1.2 Аналіз аналогічних платформ

Ринок онлайн-сервісів для замовлення 3D-друку поки що залишається малорозвиненим як у світі, так і в Україні. Більшість існуючих сайтів орієнтовані на обмін 3D-моделями або на презентацію послуг одного конкретного виконавця. Наприклад, популярна міжнародна платформа **Printables.com** пропонує величезну бібліотеку STL-файлів, але не дає змоги обирати друкаря та оформлювати замовлення з порівнянням цін, рейтингу і портфоліо [5]. Аналогічно, сервіси на

кшталт Thingiverse чи Cults3D більше схожі на тематичні спільноти або вітрини моделей, а не на повноцінні маркетплейси з можливістю вибору виконавця і контролю замовлення [6].

У межах України ситуація ще складніша. Більшість друкарів або майстерень мають окремі сайти-візитівки чи сторінки у соціальних мережах, де пропонують свої послуги, проте фактичне замовлення й обговорення деталей часто відбувається через месенджери, телефонні дзвінки або електронну пошту. Це створює додаткові складнощі для клієнта: порівняти ціни між різними виконавцями можна лише вручну, а остаточну якість роботи чи реальні відгуки перевірити доволі важко. Деякі сайти впроваджують калькулятори вартості друку, але точність цих розрахунків низька, а сам процес часто вимагає реєстрації, що відштовхує нових користувачів [7].

Ще одним важливим аспектом є відсутність єдиної системи незалежного рейтингу або об’єктивної оцінки якості. Навіть якщо виконавець має власний сайт, потенційному замовнику доводиться самостійно шукати відгуки на сторонніх ресурсах чи запитувати поради в знайомих. Водночас для самих друкарів залишається проблемою просування своїх послуг: без ефекту масштабу й централізованої платформи витрати на рекламу суттєво впливають на кінцеву ціну послуги.

Особливе місце серед українських платформ посідає ініціатива **“Друк Армія”** — волонтерський проєкт, який об’єднує багатьох власників 3D-принтерів і дозволяє координувати виготовлення корисних виробів для потреб Збройних Сил [3]. Проте ця спільнота працює на безоплатній основі й орієнтована лише на військові потреби, а не на комерційні чи побутові замовлення. Таким чином, навіть за умов активного розвитку волонтерського руху питання створення єдиної платформи для бізнесу й приватних клієнтів залишається відкритим.

До основних недоліків нинішніх рішень слід віднести складність порівняння вартості, відсутність об’єктивної системи рейтингу, обмеженість пошуку, відсутність зручної фільтрації, а також складний або незручний інтерфейс для користувача. Все це свідчить про те, що існує значна ніша для появи сучасної конкурентної платформи, яка вирішить ці проблеми і стане корисним інструментом для всіх учасників ринку.

1.3. Вимоги до платформи

Розвиток технологій 3D-друку в Україні, особливо в умовах війни, призвів до значного зростання попиту на виготовлення індивідуальних деталей та прототипів. Волонтерські ініціативи, такі як «ДрукАрмія», продемонстрували ефективність децентралізованого виробництва, проте відсутність єдиної комерційної платформи для замовлення 3D-друку залишається проблемою [8].

Згідно з дослідженнями, ринок 3D-друку в Україні має значний потенціал для зростання, особливо в галузях оборони, медицини та машинобудування [9]. Однак, існуючі сервіси не забезпечують достатньої прозорості цін, якості виконання та зручності взаємодії між замовниками та виконавцями.

На основі аналізу ринку та потреб користувачів, можна виокремити наступні вимоги до платформи:

1. **Двостороння модель користувачів:** Платформа повинна підтримувати два типи користувачів — замовників та виконавців (друкарів). Це дозволить чітко розмежувати функціональні можливості та забезпечити відповідний рівень доступу до різних розділів сервісу [10].
2. **Прозора система фільтрації:** Необхідно впровадити зручну систему фільтрації та сортування пропозицій за різними критеріями, такими як ціна, якість, терміни виконання тощо. Це сприятиме формуванню чесного конкурентного середовища та стимулюватиме виконавців до підвищення рівня обслуговування.

3. **Інтегрований чат:** Для забезпечення ефективної комунікації між сторонами необхідно впровадити інтегрований чат, прив'язаний до конкретної пропозиції. Це дозволить замовникам та друкарям обговорювати деталі замовлення, ставити запитання та надавати відповіді в межах одного інтерфейсу, що підвищить зручність та прозорість взаємодії.

4. **Система оцінювання та відгуків:** Замовники повинні мати змогу залишати відгуки про виконані замовлення, що сприятиме формуванню репутації виконавців та допоможе іншим користувачам у виборі надійних партнерів.

5. **Автоматичні сповіщення:** Платформа повинна забезпечувати автоматичні сповіщення для друкарів про нові замовлення або запити під їхніми пропозиціями. Це дозволить оперативно реагувати на потреби замовників та підвищить ефективність взаємодії.

6. **Інтуїтивно зрозумілий інтерфейс:** Платформа повинна мати інтуїтивно зрозумілий та адаптивний інтерфейс, що забезпечить позитивний користувацький досвід. Також важливо передбачити можливість масштабування сервісу для підтримки суміжних технологій, таких як лазерне різання, ЧПК та інші.

7. **Безпека та стабільність:** Забезпечення безпеки персональних даних користувачів та стабільна робота системи при високому навантаженні є невід'ємними складовими успішного функціонування платформи.

Врахування зазначених вимог сприятиме створенню ефективної та конкурентоспроможної платформи для ринку 3D-друку в Україні, що відповідатиме потребам як замовників, так і виконавців послуг.

1.4 Функціональні вимоги до платформи для замовлення послуг 3D-друку

У сучасному світі, де технології 3D-друку стають дедалі популярнішими, виникає потреба у створенні платформи, яка б забезпечувала ефективну взаємодію між замовниками та виконавцями послуг 3D-друку. Функціональні вимоги до такої платформи повинні враховувати не лише технічні аспекти, але й забезпечувати зручність користування, прозорість процесів та безпеку даних.

1.4.1 Стартова сторінка

Перший контакт користувача з платформою відбувається через стартову сторінку. Вона повинна бути інформативною та привабливою, містити короткий опис сервісу, галерею прикладів робіт, статистику зареєстрованих користувачів та посилання на реєстрацію та авторизацію для обох типів користувачів. Такий підхід сприяє залученню нових користувачів та підвищує довіру до платформи. Згідно з рекомендаціями щодо дизайну UI/UX для онлайн-маркетплейсів, важливо забезпечити легку навігацію та чітке представлення інформації на головній сторінці [17].

1.4.2 Аутентифікація та розмежування ролей

Система повинна забезпечувати чітке розмежування ролей користувачів: замовник та друкар. Після авторизації кожен користувач отримує доступ до персоналізованого кабінету з відповідним функціоналом. Такий підхід дозволяє уникнути плутанини та забезпечує безпеку даних. Згідно з принципами дизайну інтерфейсів, важливо забезпечити відповідність між системою та реальним світом, використовуючи знайомі терміни та концепції [2].

1.4.3 Персоналізація профілю

Користувачі повинні мати можливість редагувати свій профіль, зокрема змінювати аватар, ім'я користувача та інші персональні дані. Це сприяє створенню довірчих відносин між учасниками платформи та підвищує рівень взаємодії.

Персоналізація є ключовим аспектом у забезпеченні позитивного користувацького досвіду [1].

1.4.4 Створення та управління пропозиціями

Друкарі повинні мати можливість створювати пропозиції, вказуючи матеріал, ціну за грам, товщину шару, габаритні обмеження та завантажуючи до десяти фото-зразків. Створені пропозиції мають миттєво відображатися в каталозі та бути доступними для редагування або видалення. Такий функціонал забезпечує гнучкість та актуальність пропозицій.

1.4.5 Каталог пропозицій та фільтрація

Замовники повинні мати доступ до каталогу пропозицій з можливістю фільтрації за матеріалом, ціною, рейтингом виконавця та іншими параметрами. Також доцільно впровадити комбінований фільтр «ціна-якість», який дозволяє знаходити оптимальні варіанти за співвідношенням вартості та якості. Згідно з найкращими практиками дизайну UI/UX для маркетплейсів, ефективна фільтрація є ключовим елементом для покращення користувацького досвіду [1].

1.4.6 Детальна інформація про пропозицію

Кожна пропозиція повинна містити детальну інформацію, включаючи галерею зображень, опис параметрів друку, дані про друкаря та його рейтинг. Це дозволяє замовникам приймати обґрунтовані рішення та підвищує прозорість процесу.

1.4.7 Система замовлень та чат

Замовники повинні мати можливість надсилати запити на замовлення, завантажуючи STL-файли та вказуючи контактну інформацію. Після цього відкривається чат між замовником та друкарем, який прив'язаний до конкретної

пропозиції. Такий підхід забезпечує ефективну комунікацію та зберігає історію взаємодії.

1.4.8 Обробка замовлень та зворотний зв'язок

Друкарі повинні мати можливість переглядати надіслані моделі, оцінювати їх та приймати або відхиляти замовлення з обґрунтуванням. Після завершення друку необхідно надати фото-звіт, а замовник може прийняти роботу або запросити доопрацювання. Такий процес забезпечує прозорість та підвищує якість послуг.

1.4.9 Система оцінювання

Після завершення замовлення замовник має можливість оцінити роботу друкаря за п'ятибальною шкалою. Середній бал відображається в профілі друкаря та впливає на його позицію в пошуку. Це стимулює виконавців до підвищення якості послуг.

1.4.10 Безпека та збереження сесій

Система повинна забезпечувати збереження сесій користувачів до моменту їхнього виходу, а також захищати сторінки від несанкціонованого доступу. Це підвищує рівень безпеки та довіри до платформи.

1.4.11 Обов'язкова фото-звітність та індикація статусу відправлення

Після завершення друку друкар повинен надати фото-звіт готової деталі, який автоматично додається в чат. Це дозволяє замовнику перевірити якість виконання перед відправленням. Після цього друкар натискає кнопку «Відправлено на пошту», що змінює статус замовлення та інформує замовника про відправлення. Такий підхід забезпечує прозорість процесу та покращує комунікацію між сторонами.

1.4.12 Відмова від замовлення з обґрунтуванням

У випадку, якщо друкар не може виконати замовлення, він повинен мати можливість відхилити його, обов'язково вказавши причину відмови. Це дозволяє замовнику розуміти ситуацію та приймати подальші рішення.

1.4.13 Інтуїтивно зрозумілий інтерфейс

Платформа повинна мати інтуїтивно зрозумілий та адаптивний інтерфейс, що забезпечить позитивний користувацький досвід. Також важливо передбачити можливість масштабування сервісу для підтримки суміжних технологій, таких як лазерне різання, ЧПК та інші. Згідно з рекомендаціями щодо дизайну UI/UX, інтуїтивно зрозумілий інтерфейс є ключовим фактором успішності платформи [1].

1.4.14 Безпека та стабільність

Забезпечення безпеки персональних даних користувачів та стабільна робота системи при високому навантаженні є невід'ємними складовими успішного функціонування платформи. Згідно з принципами дизайну інтерфейсів, важливо забезпечити видимість статусу системи та інформувати користувачів про поточні процеси [2].

Висновки до розділу 1

Сучасний розвиток 3D-друку, особливо в Україні, вимагає появи зручної онлайн-платформи для взаємодії замовників і виконавців. Така платформа має бути зрозумілою для кожного користувача, із простим інтерфейсом, де всі основні дії — пошук, вибір, замовлення — виконуються легко й швидко. Дуже важливо чітко розділити ролі: замовник і друкар повинні мати окремі кабінети з потрібними функціями. Щоб зробити роботу сервісу прозорою і чесною, потрібні чат для спілкування по кожному замовленню, система оцінювання та відгуків, а також фото-звітність після виконання роботи. Друкар повинен мати змогу відмовитися

від замовлення з поясненням причини, а замовник — отримати відповідь і зробити вибір. Вся інформація про статус замовлення має бути доступною для обох сторін, щоб уникати непорозумінь і затримок. Безпека даних і стабільна робота навіть при великому навантаженні — це основа довіри до платформи. Дотримання цих принципів дозволить створити сучасний сервіс, який зручно використовувати і замовникам, і виконавцям, а також зробить процес замовлення 3D-друку прозорим, швидким і надійним.

2. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ЗАСТОСУНКУ

2.1 Загальна технічна постановка задачі

Технічною задачею є розробити MVP[21] веб-платформи для замовлення 3D-друку з двома основними ролями — замовник та виконавець (друкар), де кожна роль отримує свій набір функцій і інтерфейсів. Для цього застосовується Python з фреймворком Flask[18], який організовує серверну логіку та розбиває функціонал на ізольовані blueprints: окремо для аутентифікації, управління оферами, замовленнями, чатами, профілем та іншими модулями. Зберігання й пошук даних реалізовано через SQLAlchemy — ORM[20], що дозволяє працювати із сутностями «користувач», «офер», «замовлення», «чат» та іншими, будуючи зв’язки між ними прямо в коді та підтримуючи транзакційність і цілісність даних. Доступ до приватних маршрутів і операцій захищений через Flask-Login[19], що відповідає за зберігання сесій та перевірку ролей.

Всі основні взаємодії між користувачами — реєстрація, логін, створення оферів, фільтрація та перегляд пропозицій, подача замовлення з завантаженням STL-файлу, діалог у чаті, додавання фото-звітів і оцінка друкаря — розбиті на окремі маршрути та функції. Фронтенд будується з використанням Bootstrap 5 для швидкої верстки й адаптивності інтерфейсу, а всі сторінки рендеряться через Jinja2-шаблони[23], які отримують потрібні дані із Flask. Передача файлів (STL, фото) організована через стандартну логіку Flask із збереженням у папці static, а для перевірки доступу до файлів та безпеки використовуються стандартні перевірки у маршрутах. Динамічні дії (наприклад, миттєва зміна статусу, оновлення чатів) реалізуються на фронті простим JavaScript через fetch-запити, що дозволяє не ускладнювати структуру важкими SPA-фреймворками[22].

Вибрана структура проєкту (модульний Flask, SQLAlchemy ORM, Bootstrap + Jinja2) забезпечує легкість масштабування і підтримки, дозволяє ізолювати

логіку кожного модуля, а також дає можливість швидко змінювати або розширювати функціонал у майбутньому. Усі взаємодії фронтенда й бекенда чітко розділені: фронтенд відповідає за відображення і простоту користування, бекенд — за обробку запитів, бізнес-логіку і безпеку, а база даних — за збереження, пошук і зв’язки між усіма даними в системі. Такий підхід гарантує стабільну, швидку і зрозумілу роботу застосунку навіть при зростанні кількості користувачів або додаванні нових ролей і сервісів.

2.2. Вибір технологічного стеку

Вибір технологічного стеку для створення платформи 3D-друку був зумовлений необхідністю забезпечити швидкий старт розробки, простоту підтримки й можливість масштабування у майбутньому. Основу серверної частини становить мікрофреймворк Flask, який відомий своєю лаконічністю, гнучкістю й низьким порогом входу. Саме Flask дозволяє організувати чисту, модульну структуру застосунку, що є критично важливим для довгострокового розвитку проєкту. У його офіційній документації чітко підкреслено, що цей фреймворк ідеально підходить для створення MVP і швидкого прототипування веб-платформ [13].

Для роботи з базою даних обрано SQLAlchemy — потужний ORM (Object Relational Mapper[24]), який використовується через розширення Flask-SQLAlchemy. Це розширення значно спрощує інтеграцію бази даних із застосунком і дає змогу працювати з даними у вигляді об'єктів, а не сирих SQL-запитів. Такий підхід не лише прискорює розробку, а й мінімізує ризик помилок та полегшує міграцію між різними СУБД у разі масштабування. Офіційний сайт SQLAlchemy наводить численні приклади гнучкої інтеграції з Flask [14].

Щоб забезпечити сучасний вигляд та адаптивність інтерфейсу користувача, застосовується Bootstrap 5 — один із найпопулярніших фреймворків для фронтенд-розробки. Його використання дозволяє швидко створювати адаптивні макети сторінок, гарантує однаковий вигляд на різних пристроях і підтримує кросбраузерність без зайвих зусиль. Bootstrap має широку документацію, численні приклади та активну спільноту, що значно полегшує впровадження навіть складних UI-компонентів [15].

Генерація HTML-сторінок відбувається за допомогою Jinja2 — сучасного й потужного шаблонізатора для Python. Він дозволяє створювати динамічні сторінки, підставляючи змінні, обробляючи цикли й умови безпосередньо у шаблоні. Такий підхід робить розробку більш гнучкою та дозволяє швидко змінювати структуру сторінок залежно від логіки програми. Jinja2 інтегрується з Flask «із коробки», і ця зв'язка рекомендована в офіційній документації [16].

Вибраний стек технологій не тільки забезпечує швидкий старт і мінімізацію «ручної роботи» для базового MVP, а й створює міцний фундамент для подальшого масштабування, впровадження нових модулів і підтримки великої кількості користувачів. Кожен компонент стеку активно підтримується спільнотою, має багату екосистему і дозволяє легко знаходити рішення для більшості стандартних задач, що виникають у процесі розробки сучасних веб-платформ.

2.3. Визначення опорних функцій

В основі розробленого веб-застосунку лежить низка ключових функцій, які забезпечують повний і логічно зв'язаний цикл взаємодії між користувачами — замовниками та виконавцями 3D-друку. Насамперед система передбачає обов’язкову реєстрацію та аутентифікацію, що дозволяє чітко розмежувати ролі користувачів: замовник бачить каталог оферів і історію власних замовлень, тоді як

виконавець отримує доступ до особистої панелі керування своїми пропозиціями й чергою робіт. Така модель гарантує безпеку, персоналізацію інтерфейсу та захист доступу до даних, що є стандартом для сучасних веб-сервісів [1].

Виконавці мають змогу створювати, редагувати та видаляти офери — власні пропозиції послуг 3D-друку, вказуючи матеріал, ціну, габарити й додаючи фотозразки робіт. Цей функціонал реалізовано через спеціальні форми і забезпечує швидке оновлення каталогу, а також контроль за актуальністю пропозицій. Для замовників доступний каталог оферів із можливістю фільтрації за матеріалом, ціною чи рейтингом виконавця, що дозволяє швидко знаходити найкращі варіанти відповідно до своїх потреб. Застосування сучасних фреймворків і бібліотек забезпечує ефективну роботу фільтрів навіть за великої кількості пропозицій [2].

Коли замовник знаходить відповідний офер, він може створити замовлення, вказавши параметри виробу та завантаживши STL-файл, що є стандартом для 3D-друку. Це дозволяє одразу передати виконавцю всю необхідну інформацію для оцінки й подальшої роботи. Для обговорення деталей та уточнення нюансів між сторонами відкривається чат, який є невід'ємною частиною платформи. Саме чат дає змогу швидко вирішувати питання щодо характеристик виробу, строків чи логістики, а також фіксувати всі домовленості у структурованому вигляді [3].

Після завершення друку виконавець завантажує фото готової деталі — це служить доказом виконаної роботи та допомагає уникнути непорозумінь. Замовник, у свою чергу, може підтвердити якість чи попросити доопрацювання, а весь процес фіксується в історії замовлення. Останнім етапом є система оцінювання, де замовник виставляє бал за виконану роботу. Середній рейтинг автоматично відображається в профілі виконавця, впливаючи на його довіру на платформі та ранжування у каталозі. Це стимулює якісне виконання замовлень і підтримує здорову конкуренцію між учасниками ринку.

Такий набір функцій формує цілісний цикл сервісу: від пошуку й створення пропозицій до фінального оцінювання якості, що відповідає сучасним підходам до побудови ефективних онлайн-платформ [16]. Завдяки цьому кожен користувач має чіткі інструменти для досягнення власної мети, а система — єдину логіку взаємодії без зовнішніх месенджерів чи додаткових платформ.

2.4. Проєктування архітектури застосунку

Проєктування архітектури застосунку базується на модульному принципі, що забезпечується використанням Flask та його механізму Blueprints, що дозволяє створювати гнучкі та незалежні компоненти, кожен з яких відповідає за конкретну область функціональності. Обраний підхід полегшує підтримку, розробку та подальше масштабування застосунку завдяки чіткому розподілу відповідальності між окремими частинами системи.

Архітектура складається з кількох взаємопов'язаних модулів, серед яких важливе місце займає модуль автентифікації та авторизації. Цей модуль забезпечує реєстрацію нових користувачів з розмежуванням ролей — замовників та виконавців (друкарів), автентифікацію вже зареєстрованих користувачів, контроль доступу до ресурсів застосунку та управління профілями. Реалізація здійснюється за допомогою бібліотеки Flask-Login, яка спрощує процеси авторизації, збереження сесій та захисту від несанкціонованого доступу. Всі дані про користувачів зберігаються в базі даних SQLite, інтегрованій за допомогою Flask-SQLAlchemy.

Наступним важливим компонентом є модуль керування оферами (пропозиціями), де виконавці можуть створювати, редагувати та видаляти свої пропозиції, визначаючи матеріали, ціну за грам, товщину шару та допустимі розміри моделей. Офери супроводжуються можливістю завантаження до десяти фотографій-зразків для демонстрації якості робіт. Це реалізовано за допомогою

завантаження та безпечного зберігання файлів із застосуванням Werkzeug та вбудованих механізмів Flask для роботи з файлами та шляхами.

Модуль управління замовленнями є ключовим для взаємодії між замовником та друкарем. Замовники можуть створювати замовлення, завантажуючи STL-файли, після чого система розраховує орієнтовну вагу та вартість друку за параметрами оферу. Підтвердження, відхилення чи оновлення статусу замовлень здійснюються друкарем через власний кабінет, де також є можливість додавати причину відмови чи фото-звітність про виконання замовлення. Це забезпечує прозорість процесу виконання робіт та значно підвищує зручність і довіру між сторонами.

Окремим важливим компонентом є модуль чату, реалізований з прив'язкою до кожного конкретного замовлення. Це дозволяє сторонам оперативно вирішувати питання щодо конкретних замовлень, уточнювати деталі роботи чи вносити корективи безпосередньо у процесі виконання завдань. Повідомлення чату зберігаються у відповідній таблиці бази даних, що забезпечує постійний доступ до історії спілкування навіть після завершення замовлення.

Основні загальні маршрути, такі як головна сторінка, каталоги пропозицій, детальні сторінки пропозицій та сторінки перегляду профілів користувачів, реалізуються в загальному модулі "main", який також відповідає за обробку початкових запитів користувачів та маршрутизацію на інші модулі відповідно до потреб.

З точки зору фронтенду використовується HTML5 у поєднанні з Jinja2 для генерації динамічних сторінок, а інтерфейс застосунку адаптивно побудований на основі CSS-фреймворку Bootstrap 5. Завдяки такій реалізації сторінки коректно відображаються на різних пристроях, забезпечуючи зручність використання на мобільних телефонах, планшетах та десктопах.

Таким чином, модульна структура та використання сучасних інструментів і технологій дозволяють досягти високої гнучкості, продуктивності та можливості розширення системи, що є надзвичайно важливим для MVP-проєктів, спрямованих на швидку розробку та тестування основних гіпотез. Вибраний підхід забезпечує чіткий розподіл обов'язків між фронтендом, бекендом і базою даних, гарантує простоту підтримки коду, його читабельність та зрозумілість для майбутніх розробників, що відповідатиме сучасним вимогам та стандартам розробки веб-застосунків на Python з використанням Flask.

2.5. Проєктування бази даних

Малюнок 2.5 - Діаграма класів

У застосунку використано реляційну базу даних на основі Flask-SQLAlchemy. Моделі чітко поділені за ролями: Customer (замовник) та

Contractor (друкар). Обидва класи реалізують автентифікацію через UserMixin і зберігають email, пароль, зображення профілю та рейтинг. Кожен замовник має список замовлень, кожен друкар — список оферів і замовлень.

Модель Offer описує 3D-пропозицію, що включає матеріал, висоту шару, ціну за грам і допустимі розміри. Кожен офер прив'язаний до конкретного друкаря.

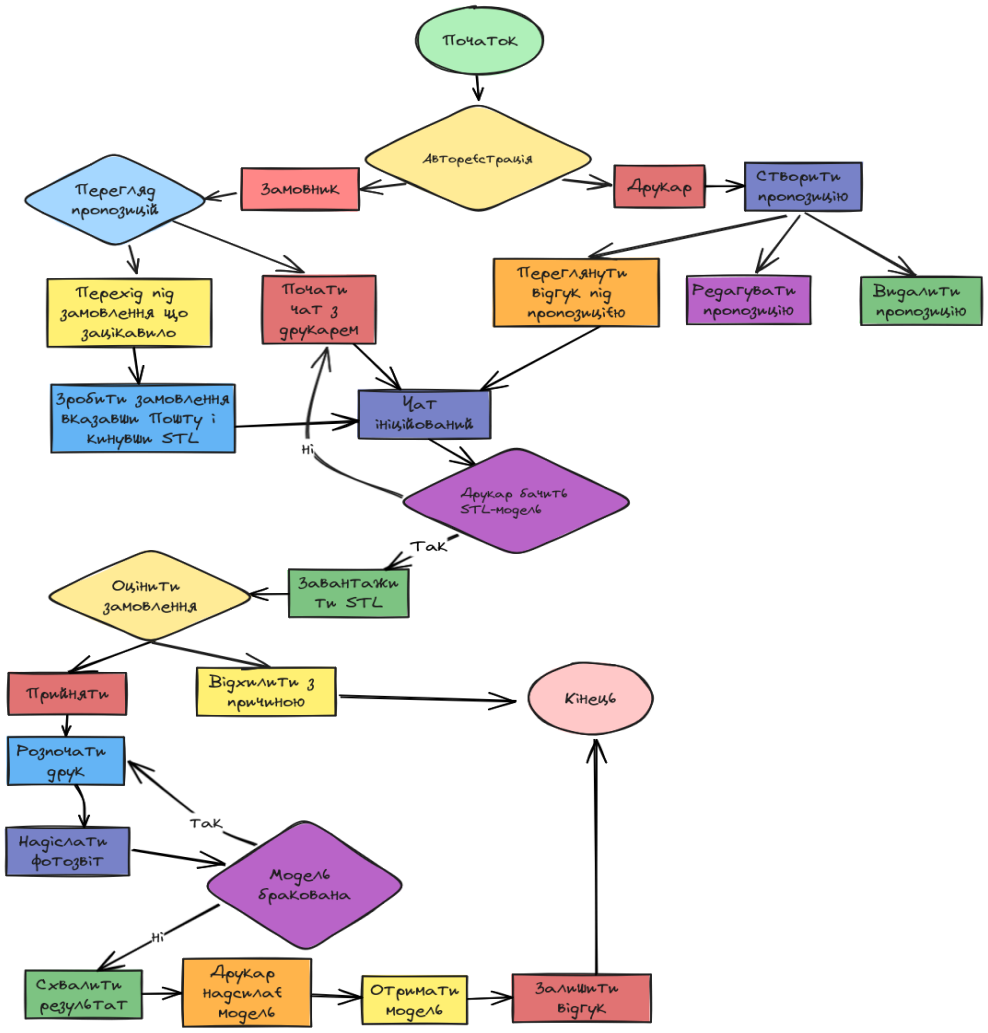
Order містить дані замовлення: STL-файл, ціну, вагу, статус виконання, дані доставки, і належить одразу трьом сутностям — оферу, замовнику і друкарю. Також замовлення має зв’язок із повідомленнями (ChatMessage) та оцінками (Rating).

Модель ChatMessage дозволяє зберігати історію спілкування між користувачами в межах конкретного замовлення. Передбачено збереження тексту, часу, ролі відправника та прикріплених зображень.

Модель Rating зберігає оцінки від замовників після доставки. Вона пов'язана із замовленням, замовником і друкарем, а також забезпечує автоматичне оновлення середнього рейтингу виконавця.

Усі моделі зв’язані між собою через зовнішні ключі. Це забезпечує узгодженість, спрощує складні запити та дозволяє легко виконувати фільтрацію, сортування й обрахунки в межах реляційної структури. Структура розширювана, з підтримкою багатьох замовлень, пропозицій та повідомлень одночасно.

2.6. Детальний аналіз опорних функцій



Малюнок 2.6 Діаграма взаємодії користувачів

2.6.1. Реєстрація та аутентифікація користувачів

У системі реєстрації та аутентифікації користувачів все побудовано на тому, щоб будь-хто міг швидко стати частиною платформи та потім безпечно заходити у свій кабінет. На фронтенді це реалізовано у вигляді простих і зрозумілих форм (register.html та login.html), які працюють на Bootstrap 5 для гарної адаптивності на будь-яких пристроях. Для кожної ролі — замовник чи виконавець — під час реєстрації відразу визначається тип профілю. При надсиланні форми бекенд приймає дані, перевіряє, чи унікальний email, чи співпадають паролі, чи все правильно введено. Якщо дані коректні, пароль хешується через функції з werkzeug.security (це стандарт для Flask [13]) і дані нового користувача

записуються у таблицю Customer або Contractor (окремі таблиці для кожної ролі). Якщо завантажене фото — воно зберігається у окремій папці, і шлях до нього також зберігається у профілі. Вхід у систему працює за класичним принципом — користувач вводить email та пароль, система знаходить користувача по email, порівнює захешований пароль з тим, що введено, і якщо все ок — авторизує його через Flask-Login, встановлює роль у сесії та перенаправляє на свій дашборд. Інформація про користувача після реєстрації і під час кожного входу зберігається у базі даних (site.db) у відповідних таблицях. Завдяки цьому легко відокремити доступ до різних частин платформи для різних ролей та гарантувати безпеку авторизації. Усе зроблено стандартними інструментами Flask, без зайвих бібліотек, максимально просто й ефективно.

2.6.2. Створення та управління оферами

Офер — це пропозиція друкаря, яку можуть переглядати і замовляти клієнти. Створення офера починається з того, що виконавець заходить у свій кабінет та натискає “Створити пропозицію”. На фронтенді відкривається форма (create_offer.html), де потрібно обрати матеріал (з випадаючого списку — PLA, ABS, PETG та інші), ввести ціну за грам, мінімальний і максимальний розмір виробу, а також за бажанням завантажити фото прикладу своїх робіт (до 10 фото). Далі дані відправляються через POST на сервер, де бекенд (routes.py) приймає їх, перевіряє на валідність і додає у таблицю Offer. Фотографії зберігаються у папці app/static/examples з іменем, прив’язаним до ідентифікатора офера. Усі поля офера одразу зв’язуються з конкретним користувачем-виконавцем через зовнішній ключ. Офер після створення стає доступним для перегляду клієнтам у каталозі. Також друкар може редагувати або видалити свою пропозицію через схожі форми (все перевіряється, щоб змінювати або видаляти офер міг лише його власник). Усі дані про офери, включаючи фото, матеріали, ціни, габарити, акуратно зберігаються у таблиці Offer, що забезпечує швидку роботу фільтрації та пошуку по базі. Так

побудовано логіку, щоб виконавець мав повний контроль над своїми пропозиціями, а клієнт завжди бачив актуальну інформацію.

2.6.3. Каталог оферів з фільтрацією

Каталог оферів — це головна сторінка для клієнта, де він бачить усі актуальні пропозиції від друкарів. Вся логіка побудована так, щоб користувач міг легко знайти саме те, що йому потрібно: можна фільтрувати офери по матеріалу, ціні, рейтингу, або комбінувати ці фільтри разом. На фронтенді (offers_list.html) є поля для вибору матеріалу, вводу діапазону цін, сортування по рейтингу чи ціні, що відправляють параметри у вигляді GET-запитів. Бекенд у routes.py приймає ці параметри, формує складний SQLAlchemy-запит — наприклад, можна відібрати всі офери з матеріалом PLA, з ціною до 50 грн/грам і рейтингом виконавця не менше 4. Запит виконується до таблиці Offer із підключенням таблиці Contractor для рейтингу. Після виконання запиту відфільтровані дані потрапляють у шаблон і показуються користувачу у вигляді зручних карток із фотографіями, описом, ціною та рейтингом. Це забезпечує простий, але потужний механізм пошуку потрібної пропозиції серед багатьох друкарів, не навантажуючи сервер і не ускладнюючи інтерфейс для користувача.

2.6.4. Створення замовлень

Створення замовлення — це ключова дія для клієнта: він обирає офер, натискає “Замовити” і заповнює просту форму, в якій додає STL-файл, вводить деталі доставки і, якщо потрібно, додає додаткову інформацію. На фронтенді ця форма (наприклад, у create_order.html) перевіряє, щоб був завантажений файл саме з розширенням .stl. Коли форма відправляється, бекенд (routes.py) приймає файл, перевіряє його, записує дані у таблицю Order, а файл фізично зберігається у папці app/static/stl_files. У таблицю записуються всі основні дані: ім’я файлу, орієнтовна вага, ціна, статус замовлення, прив’язка до офера, замовника і виконавця. Якщо

раніше було створене “чернеткове” замовлення або відхилене — воно просто оновлюється, не створюючи дублікати. Це спрощує подальшу комунікацію між клієнтом і виконавцем та дозволяє уникнути зайвого навантаження на базу. Кожне замовлення має унікальний статус (“Draft”, “Очікує підтвердження”, “У виконанні” тощо), що допомагає відстежувати прогрес виконання роботи.

2.6.5. Чат між замовником та виконавцем

Після створення замовлення між клієнтом та виконавцем відкривається особистий чат, де вони можуть обговорити деталі замовлення. На фронтенді (у шаблоні offer_detail.html та через JavaScript) відображається історія переписки і поле для введення нових повідомлень. Усі повідомлення надсилаються через AJAX-запити (без перезавантаження сторінки) на спеціальний endpoint бекенду. Бекенд (routes.py) приймає нове повідомлення, записує його у таблицю ChatMessage із зазначенням замовлення, автора і ролі (замовник чи виконавець), а потім повертає оновлену історію чату у вигляді JSON. Кожен чат прив’язаний до конкретного замовлення, тож ніхто сторонній не зможе прочитати або написати повідомлення у чужу переписку. Якщо виконавець надсилає фото готової деталі, повідомлення позначається спеціальним тегом і автоматично відображає фото у чаті. Це дає змогу вирішувати всі питання безпосередньо у замовленні, забезпечує прозорість виконання роботи та економить час обом сторонам.

2.6.6. Фото-звітність та підтвердження виконання

Коли виконавець завершує друк, він має можливість завантажити фотографію готової деталі — це підтвердження того, що замовлення виконано. На фронтенді це виглядає як просте поле для вибору фото у деталях замовлення. Бекенд приймає файл, перевіряє, чи все гаразд, і зберігає фото у папці app/static/examples. Посилання на файл зберігається у полі progress_image у таблиці Order. Статус замовлення автоматично змінюється на “Друк завершено”.

Після цього замовник бачить фото у своєму кабінеті, може переконатися у якості виконання, і якщо все добре — підтвердити виконання замовлення. Таким чином, уся інформація зберігається централізовано, і кожна сторона має докази виконання роботи. Це мінімізує конфлікти і робить процес максимально прозорим для обох сторін.

2.6.7. Система оцінювання

Після отримання готового виробу замовник може оцінити роботу виконавця через форму, де вибирає оцінку від 1 до 5. Фронтенд дає простий інтерфейс для вибору рейтингу, який надсилається на сервер. Бекенд приймає цю оцінку, створює запис у таблиці Rating, перевіряючи, що замовлення вже відправлене і що оцінку ставить саме той клієнт, для якого було виконано замовлення. Після цього у профілі виконавця автоматично перераховується середній рейтинг на основі всіх оцінок. Це простий і справедливий механізм, який мотивує виконавців працювати якісніше, а клієнтів — залишати чесні відгуки, що в кінцевому підсумку підвищує довіру до платформи.

Висновок до розділу «Постановка задачі та проєктування застосунку»

У цьому розділі чітко описано, яку платформу потрібно створити та як саме це зробити: для замовників і друкарів, з мінімумом складнощів і максимальною простотою. Пояснено, чому обрано саме такі технології — Flask, Bootstrap, SQLAlchemy — і як вони допомагають швидко розробити й легко підтримувати сайт. Кожна важлива функція розібрана окремо — показано, як фронтенд, бекенд і база даних працюють разом. Вся система побудована так, щоб її можна було легко доповнювати й масштабувати без зайвих переробок. Завдяки цьому підходу можна швидко отримати робочу платформу, яку зручно розвивати далі, додаючи нові можливості для користувачів і не ламаючи основну логіку.

3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ПЛАТФОРМИ

3.1. Опис середовища розробки

Розробку веб-платформи для замовлення та виконання 3D-друку було здійснено на ноутбуці HP Laptop 15s-eq1xxx із встановленою операційною системою Linux Mint 22.1 Cinnamon (x86_64). Такий вибір забезпечив стабільність роботи, підтримку сучасних програмних інструментів і високу продуктивність для тестування локального серверного оточення. Основною мовою програмування став Python версії 3.12.3, що дозволяє використовувати актуальні функції та бібліотеки для розробки сучасних веб-застосунків.

Для реалізації серверної частини було використано мікрофреймворк Flask, який надає можливість гнучкої організації структури проєкту та дозволяє швидко впроваджувати необхідний функціонал MVP. Разом із Flask застосовувалися такі ключові розширення: Flask-SQLAlchemy (для інтеграції з реляційною базою даних через ORM), Flask-Login (для керування автентифікацією та сесіями користувачів), Flask-Migrate (для виконання міграцій структури бази даних). Фронтенд-представлення побудовано із застосуванням шаблонізатора Jinja2 та CSS-фреймворку Bootstrap 5, що гарантувало адаптивність та сучасний вигляд інтерфейсу на різних пристроях.

Організація роботи над проєктом велася у середовищі розробки Visual Studio Code, яке було обране через його гнучкість, підтримку розширень для Python, Git і автодоповнення коду. Для контролю версій використовувався сервіс GitHub, де створено окремий репозиторій проєкту [30]. Це забезпечило збереження історії змін, можливість швидкого повернення до попередніх версій і командну роботу за потреби. Усі основні дії щодо розробки, запуску та тестування додатку виконувались через вбудований термінал bash у середовищі Linux Mint, що дозволяє оперативно керувати залежностями, активувати

віртуальне середовище (venv), запускати сервер у режимі розробки та працювати з міграціями бази даних.

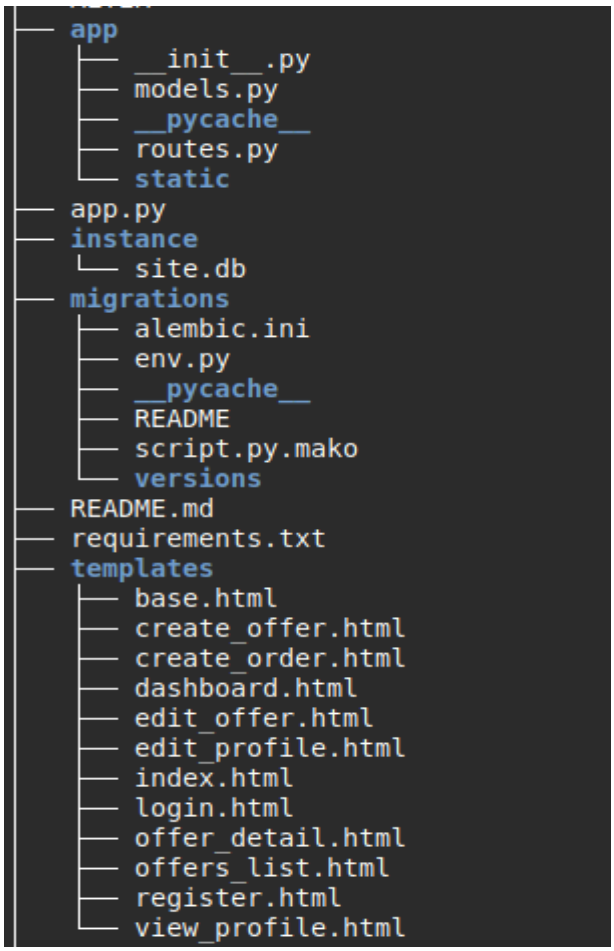
Налаштування оточення для запуску додатку починалося зі створення та активації віртуального середовища Python, встановлення всіх залежностей із файлу requirements.txt, після чого запуск сервера здійснювався стандартною командою python app.py. Для тестування роботи використовувався стандартний Flask-сервер у режимі debug, що дозволяло оперативно виявляти та виправляти помилки ще на етапі розробки. Основна база даних створювалась локально у вигляді файлу SQLite[29], що значно спрощувало розгортання й обслуговування MVP.

Завдяки комплексному підходу до вибору інструментів та правильній організації середовища вдалося забезпечити не лише простоту й швидкість розробки, але й надійну роботу платформи під час тестування, що важливо для подальшого розгортання сервісу у виробничому середовищі.

3.2 Архітектура та реалізація системи

3.2.1. Структура проєкту

Архітектура програмного коду платформи побудована на модульному принципі, що дає змогу чітко розмежувати функціонал, забезпечити гнучкість розвитку й простоту підтримки. У кореневій директорії проєкту розміщено головні файли конфігурації, а також піддиректорії для основних компонентів застосунку.



Малюнок 3.2.1 Дерево структури файлів

Ключовим елементом є папка **app**, у якій містяться всі основні модулі застосунку. Саме тут розміщено такі файли:

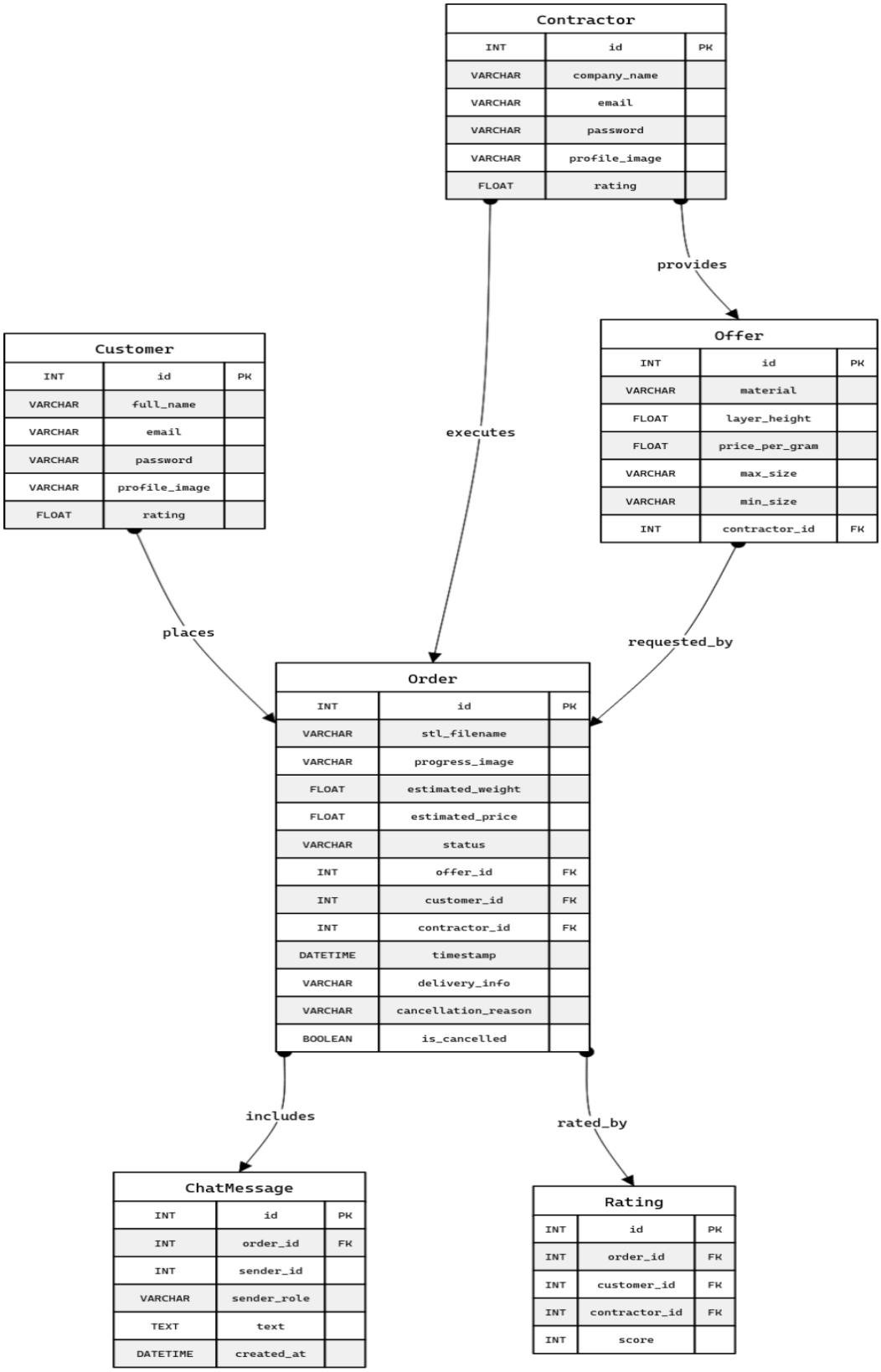
- **__init__.py** — містить функцію create_app() для створення екземпляра Flask-додатку, підключення розширень (ORM, сесій, міграцій) та реєстрації blueprint-ів. Також тут відбувається ініціалізація бази даних та налаштування менеджера сесій користувачів.
- **models.py** — файл із визначенням усіх основних моделей SQLAlchemy (Customer, Contractor, Offer, Order, ChatMessage, Rating). Тут описано структуру таблиць, зв’язки між ними та основні методи роботи з даними.

- **routes.py** — містить основну бізнес-логіку та маршрутизацію, розділену на окремі функції для кожного типу операцій: реєстрація, логін, створення та перегляд оферів, оформлення замовлень, обробка чатів, завантаження фото тощо. Саме в цьому файлі зосереджено “ядро” обробки запитів із фронтенду.
- **static/** — папка зі статичними ресурсами: зображеннями, CSS- і JS-файлами, а також збереженими фото-зразками оферів та STL-файлами для друку. Забезпечує швидкий доступ до всіх медіа, необхідних для роботи платформи.
- **templates/** — шаблони Jinja2 для всіх основних сторінок застосунку: реєстрації, авторизації, головної сторінки, каталогу оферів, особистого кабінету, перегляду деталей замовлення тощо. Кожен шаблон отримує динамічні дані від бекенду й відповідає за адаптивне відображення контенту.
- У кореневій папці проєкту розташовано:
- **app.py** — точка входу, яка викликає функцію створення застосунку та запускає локальний сервер для тестування й розробки.
- **requirements.txt** — список залежностей, що гарантує ідентичність середовища для всіх учасників проєкту.
- **README.md** — коротка інструкція для запуску та опис структури.

Така побудова дає змогу швидко орієнтуватися у проєкті, а також за потреби масштабувати систему: додавати нові модулі (наприклад, для інтеграції з платіжними системами чи зовнішніми сервісами) без зміни базової логіки. Кожен файл або папка має чітко визначене призначення, що знижує ризик дублювання коду і спрощує колективну розробку.

3.2.2. Реалізація моделей бази даних

У цьому проєкті для збереження всіх даних про користувачів, пропозиції, замовлення та переписку застосовується реляційна база даних, яку я реалізував за допомогою SQLAlchemy – це зручно, бо можна працювати не напряму із SQL, а через Python-класи. Усі ключові об’єкти – замовники, виконавці (друкарі), офери (пропозиції), замовлення, повідомлення у чаті та оцінки – представлені окремими моделями, і кожна така модель повністю відповідає таблиці у базі.



Малюнок 3.2.2 Діаграма класів

Основна ідея тут у тому, що кожен користувач має свою роль: **Customer** – це звичайний клієнт, а **Contractor** – це виконавець, який друкує моделі. Для кожної ролі створена своя модель з різними полями, наприклад, замовник має ім’я та e-mail, а друкар ще й назву компанії. Усі паролі зберігаються тільки у зашифрованому вигляді, а для аватара користувача є поле для збереження шляху до фото. Для того, щоб не було дублювання, вся авторизація зроблена через стандартний Flask-Login.

Зв’язки між моделями прості: кожен замовник може створити багато замовлень, а кожен друкар – додати багато своїх пропозицій (оферів) і отримати багато замовлень від різних клієнтів. Пропозиція (Offer) завжди належить одному конкретному виконавцю, і до неї можуть прив’язуватися різні замовлення. В самому замовленні (Order) зберігається посилання і на офер, і на клієнта, і на виконавця – це дуже зручно, бо в один запит можна дістати все, що потрібно для відображення на сторінці.

Для кожного замовлення ведеться історія листування у вигляді чату: всі повідомлення зберігаються в таблиці **ChatMessage**, де фіксується, хто і коли відправив текст або фото. Окремо організована система оцінювання – після завершення друку клієнт може залишити оцінку від 1 до 5 балів; всі оцінки автоматично рахуються для підсумкового рейтингу виконавця, і це одразу відображається у його профілі. Всі ключові зв’язки у моделях оголошені як зовнішні ключі – це дозволяє зберігати цілісність і швидко робити вибірки по всіх потрібних сутностях.

Міграції бази даних виконуються через Flask-Migrate[25]. Я просто змінюю або додаю поле в коді моделей, потім запускаю дві команди в терміналі: flask db migrate і flask db upgrade – база сама підлаштовується під нові вимоги, і всі

таблиці завжди актуальні. Такий підхід дуже спрощує життя, бо не треба вручну переписувати структуру бази після кожної зміни.

Загалом, обрана структура моделей вийшла простою, логічною і без зайвого дублювання. Всі зв’язки очевидні, дані легко знаходити і відображати, а будь-які нові функції можна додати без болю і хаосу у структурі. Це дозволяє швидко розвивати платформу, зберігаючи стабільність і простоту підтримки коду.

3.2.3. Реалізація бекенду

У цьому підрозділі я опишу, як у моєму застосунку працюють ключові маршрути, як відбувається валідація даних, перевірка ролей і захист доступу, а також покажу прості приклади логіки обробки форм та відповіді серверу.

Вся серверна логіка побудована на Flask. Для кожної дії — реєстрації, входу, створення пропозиції, оформлення замовлення, чату та оцінки — створено окремі маршрути. Кожен маршрут приймає дані з форми (POST) або повертає сторінку з потрібною інформацією (GET).

Реєстрація та логін. Для реєстрації та входу використовуються маршрути /register та /login. Тут проводиться перевірка — чи вказано роль (customer чи contractor), чи не зайнятий email, чи співпадають паролі. Пароль хешується через werkzeug.security. При успішній реєстрації користувач додається в базу, після входу — авторизується через Flask-Login.

```python
#  Реєстрація
@main.route('/register', methods=['GET', 'POST'])
def register():
    role = request.args.get('role')  # ?role=customer | ?role=contractor
    if not role or role not in ['customer', 'contractor']:
        return "Потрібен ?role=customer чи ?role=contractor", 400

    if request.method == 'POST':
        name = request.form.get('name')
        email = request.form.get('email')
        password = request.form.get('password')
        confirm = request.form.get('confirm')
        file = request.files.get('profile_image')

        if password != confirm:
            return "Паролі не співпадають!"

        model = Customer if role == 'customer' else Contractor
        if model.query.filter_by(email=email).first():
            return "Email вже використовується!"

        hashed_password = generate_password_hash(password, method='pbkdf2:sha256')
        filename = 'default-avatar.png'

        # Якщо додали фото
        if file and file.filename != '':
            filename = secure_filename(file.filename)
            save_path = os.path.join('app', 'static', 'uploads', filename)
            file.save(save_path)

        if role == 'customer':
            new_user = Customer(full_name=name, email=email, password=hashed_password, profile_image=filename)
        else:
            new_user = Contractor(company_name=name, email=email, password=hashed_password, profile_image=filename)

        db.session.add(new_user)
        db.session.commit()

        return redirect(url_for('main.login') + f'?role={role}')

    return render_template('register.html', role=role)
```

Малюнок 3.2.3.1 функціональне рішення реєстрації**Валідація** тут дуже проста: якщо користувач ввів неправильні дані або спробував зареєструвати вже існуючий email — сервер повертає коротке повідомлення про помилку.

Створення пропозиції (create_offer) Тільки користувач з роллю «друкар» може створити пропозицію. Для цього перевіряється роль (у сесії), інакше — повертається помилка. При створенні оферу перевіряються обов’язкові поля: матеріал, ціна, розмір, а також обмеження на кількість фото.

```python
#  Створення пропозиції
@main.route('/contractor/create_offer', methods=['GET', 'POST'])
@login_required
def create_offer():
    if session.get('role') != 'contractor':
        return "Доступ лише для друкарів", 403

    if request.method == 'POST':
        # якщо обрано other — забираємо текст із поля material_other
        material = request.form.get('material')
        if material == 'other':
            material = request.form.get('material_other', '').strip() or 'Other'

        layer_height   = float(request.form['layer_height'])
        price_per_gram = float(request.form['price_per_gram'])
        max_size       = request.form['max_size']
        min_size       = request.form['min_size']

        offer = Offer(
            material       = material,
            layer_height   = layer_height,
            price_per_gram = price_per_gram,
            max_size       = max_size,
            min_size       = min_size,
            contractor_id  = current_user.id
        )
        db.session.add(offer)
        db.session.commit()

        # зберігаємо до 10 фотографій-прикладів
        files         = request.files.getlist('images')
        examples_path = os.path.join('app', 'static', 'examples')
        os.makedirs(examples_path, exist_ok=True)

        for idx, f in enumerate(files[:10], start=1):
            if f and f.filename:
                fname = f'offer_{offer.id}_example_{idx}.jpg'
                f.save(os.path.join(examples_path, fname))

        return redirect(url_for('main.dashboard', role='contractor'))

    # GET — показуємо форму зі списком MATERIALS
    return render_template('create_offer.html', materials=MATERIALS)
```

Малюнок 3.2.3.2 функціональне рішення створення пропозицій

Замовлення може створити тільки замовник. Файл обов’язково повинен бути у форматі STL, це перевіряється перед збереженням. Якщо STL немає — повертається помилка.

```python
#  Створення / повторне створення замовлення
@main.route('/order/create/<int:offer_id>', methods=['GET', 'POST'])
@login_required
def create_order(offer_id):
    if session.get('role') != 'customer':
        return "Доступ лише для замовників", 403

    offer = Offer.query.get_or_404(offer_id)

    #  POST
    if request.method == 'POST':
        file = request.files.get('stl_file')
        if not file or not file.filename.endswith('.stl'):
            return "Потрібен STL-файл!", 400

        # шукаємо «активне» замовлення (Draft / Очікує … / …)
        active = (Order.query…

        # якщо є активне — просто перейдемо до нього (не даємо плодити дублікати)
        if active:
            return redirect(url_for('main.offer_detail',…

        # якщо ж останнє замовлення було **відхилено**, — ПЕРЕВИКОРИСТОВУЄМО його
        rejected = (Order.query
                    .filter_by(offer_id      = offer_id,
                               customer_id   = current_user.id,
                               contractor_id = offer.contractor.id,
                               status        = 'Відхилено')
                    .order_by(Order.timestamp.desc())
                    .first())

        #  зберігаємо файл
        stl_dir  = os.path.join('app', 'static', 'stl_files')
        os.makedirs(stl_dir, exist_ok=True)
        filename = secure_filename(file.filename)
        file.save(os.path.join(stl_dir, filename))

        # TODO: обчислюємо вагу реально
        estimated_weight = 50.0
        estimated_price  = estimated_weight * offer.price_per_gram

        if rejected:                # ▲ оновлюємо «відхилене» замовлення…
        else:                        # ▼ створюємо геть новий Order
            new_order = Order(…
            db.session.add(new_order)
            db.session.commit()
            oid = new_order.id

        return redirect(url_for('main.offer_detail',
                                offer_id = offer.id,
                                order    = oid))
    #  GET
    return render_template('create_order.html', offer=offer)
```

Малюнок 3.2.3.3 функціональне рішення створення замовлень

Чат та відправка повідомлень

Чат працює тільки між тими, хто має доступ до конкретного замовлення. Для кожного повідомлення перевіряється, чи користувач справді є учасником цього чату.

```python
#  CHAT
def _allowed(order):
    return ((session['role']=='customer'   and order.customer_id==current_user.id) or
            (session['role']=='contractor' and order.contractor_id==current_user.id))
```

Малюнок 3.2.3.4 реалізація перевірки користувача

Оцінка (rating)

Тільки замовник, який отримав замовлення, може виставити оцінку. Перевіряється статус замовлення і чи не було вже оцінки для цього замовлення.

```python
#  Оцінити друкара
@main.route('/order/<int:order_id>/rate', methods=['POST'])
@login_required
def rate_order(order_id):
    order = Order.query.get_or_404(order_id)
    #  доступ лише для клієнта цього
    if session.get('role') != 'customer' or order.customer_id != current_user.id:
        return "Доступ заборонено", 403
      You, 2 weeks ago • Обов'язкове фото для завершення друку, виправлено warnin…
    if order.status != 'Відправлено':
        return "Оцінити можна тільки після відправки", 400

    if Rating.query.filter_by(order_id=order.id, customer_id=current_user.id).first():
        return "Оцінку вже виставлено", 400

    try:
        score = int(request.form.get('score', 0))
    except ValueError:
        score = 0
    score = max(1, min(score, 5))

    r = Rating(order_id=order.id,
               customer_id=current_user.id,
               contractor_id=order.contractor_id,
               score=score)
    db.session.add(r)

    # Перерахунок середнього рейтингу для цього друкаря
    avg = db.session.query(db.func.avg(Rating.score)) \
                    .filter(Rating.contractor_id == order.contractor_id) \
                    .scalar() or 0
    contractor = Contractor.query.get(order.contractor_id)
    contractor.rating = round(avg, 2)
    db.session.commit()

    if request.headers.get('X-Requested-With') == 'XMLHttpRequest':
        return '', 204
    return redirect(url_for('main.offer_detail', offer_id=order.offer_id, order=order.id))
```

Малюнок 3.2.3.5 функціональне рішення оцінки

Уся обробка форм у бекенді побудована так, щоб користувач завжди отримував просту та зрозумілу відповідь. Якщо дія виконана успішно, наприклад, створено замовлення чи відправлено повідомлення, сервер повертає або редірект на потрібну сторінку, або просто коротку відповідь без зайвих даних, наприклад, return '', 204 для AJAX-запитів. Якщо виникає помилка — наприклад, не співпадають паролі, невірна роль або користувач не має доступу до певної дії — повертається короткий текст з поясненням помилки, або стандартний код помилки (400 чи 403). Для захисту всіх важливих дій використовується Flask-Login: кожен маршрут, який працює з особистими даними або дозволяє щось змінювати, позначений декоратором @login_required. Усередині маршруту додатково перевіряється, чи належить дія саме цьому користувачу та чи відповідає роль очікуваній. Будь-яка спроба доступу до чужих файлів або сторінок завершується поверненням помилки 403. Завдяки такому підходу бекенд залишається максимально простим і логічним: усі основні перевірки виконуються прямо в маршрутах, що дає змогу легко знаходити помилки, швидко реагувати на проблеми і гарантувати безпеку для кожного користувача.

3.2.4. Реалізація фронтенду

Фронтенд платформи побудований на шаблонах Jinja2, що дозволяє динамічно вставляти дані у HTML-сторінки для кожного користувача. Основні сторінки — це реєстрація (register.html), вхід (login.html), каталог пропозицій (offers_list.html), детальна сторінка офера (offer_detail.html), чат під замовленням, сторінка створення й перегляду замовлень.

Всі форми (наприклад, реєстрація, створення офера чи замовлення) мають просту структуру: поля вводу, кнопки, повідомлення про помилки. Для фільтрації у каталозі використано HTML-форми з селектами та інпутами для ціни,

матеріалу, рейтингу. Галерея фотографій для оферів реалізована через Bootstrap Carousel або просту сітку зображень. Для рейтингу використовується JS-бібліотека зі зручними зірочками, що дозволяє клієнту обрати оцінку без перезавантаження сторінки.

Чат під кожним замовленням — це простий блок з історією повідомлень та формою для введення нового тексту. Всі повідомлення підтягуються через AJAX (fetch)[26], і нові відображаються одразу після відправки, без перезавантаження сторінки.

Інтерфейс оформлений через Bootstrap: сітка, кнопки, модальні вікна, вкладки, форми — все максимально адаптивне й виглядає однаково добре на ПК та мобільних пристроях. Додатково використана стороння бібліотека jsdelivr[28] для анімованих рейтингів, підключенна star-rating-svg[27].В Index-сторінці були застосовані сучасні блоки та анімовані елементи для кращої першої взаємодії, з реальними фото-робіт і відгуками, а також стильними кнопками входу.

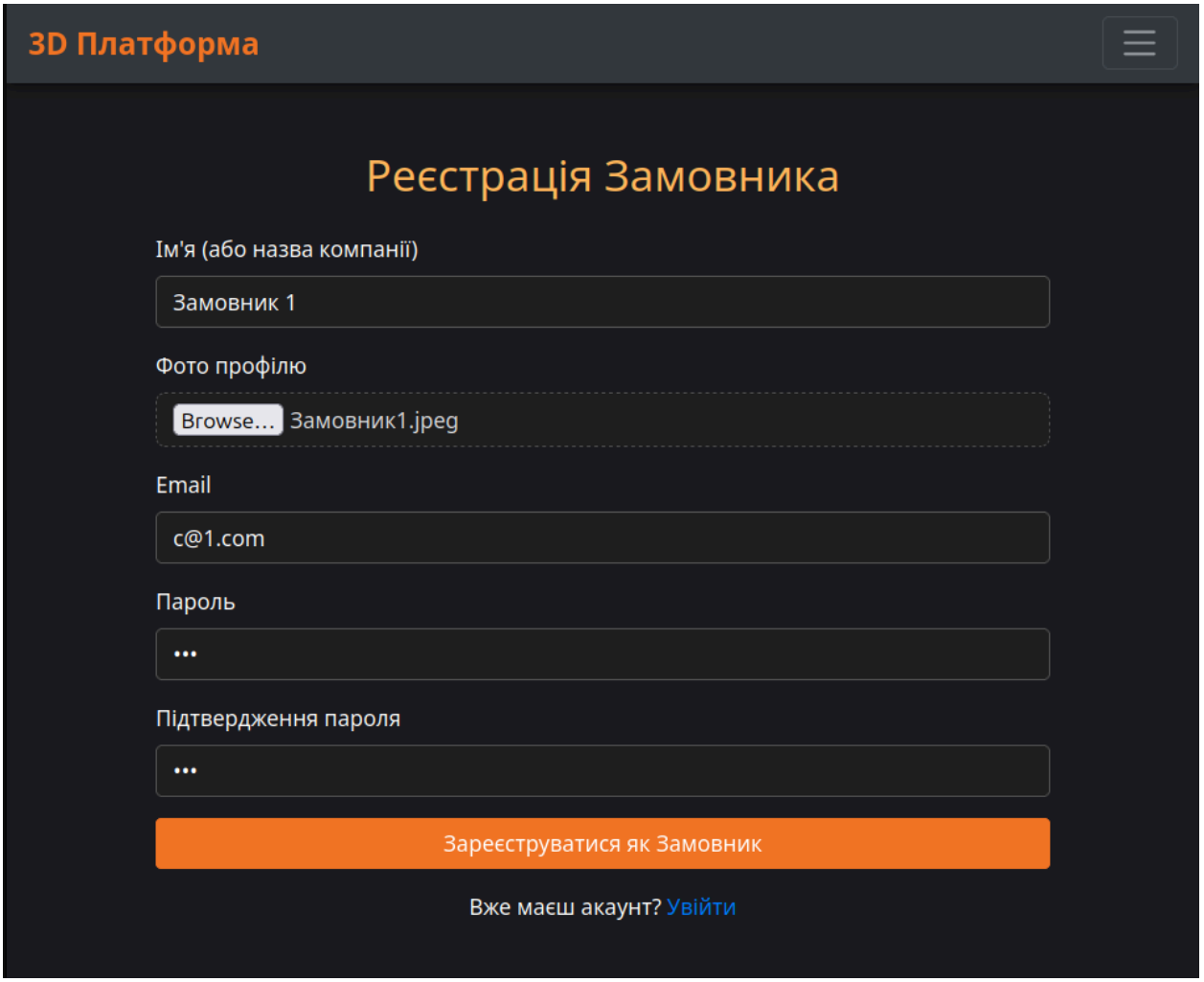
Загалом, фронтенд зроблено простим, легким для сприйняття, без складних скриптів, з акцентом на швидкість завантаження і зручність користувача. Кожен шаблон містить лише потрібний мінімум елементів для свого сценарію, що дозволяє швидко орієнтуватися і не перевантажує інтерфейс.

3.3. Покроковий сценарій роботи користувача з платформою

Щоб наочно показати, як працює система, опишу типові сценарії для обох ролей з реальними скріншотами кожного кроку.

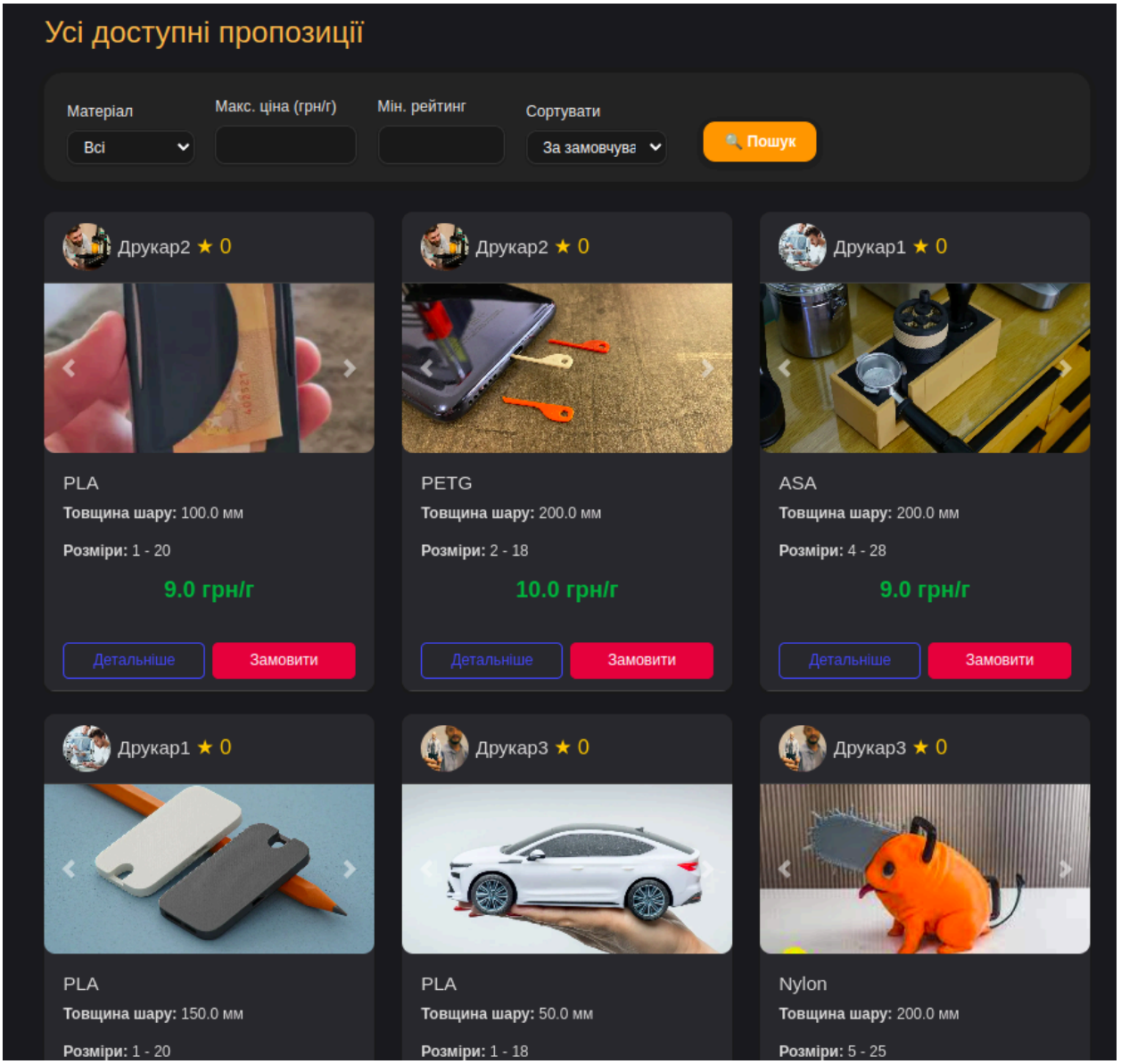
Для замовника

Все починається зі сторінки реєстрації, де користувач обирає свою роль, вводить ім’я, email, пароль та за бажанням додає фото профілю.



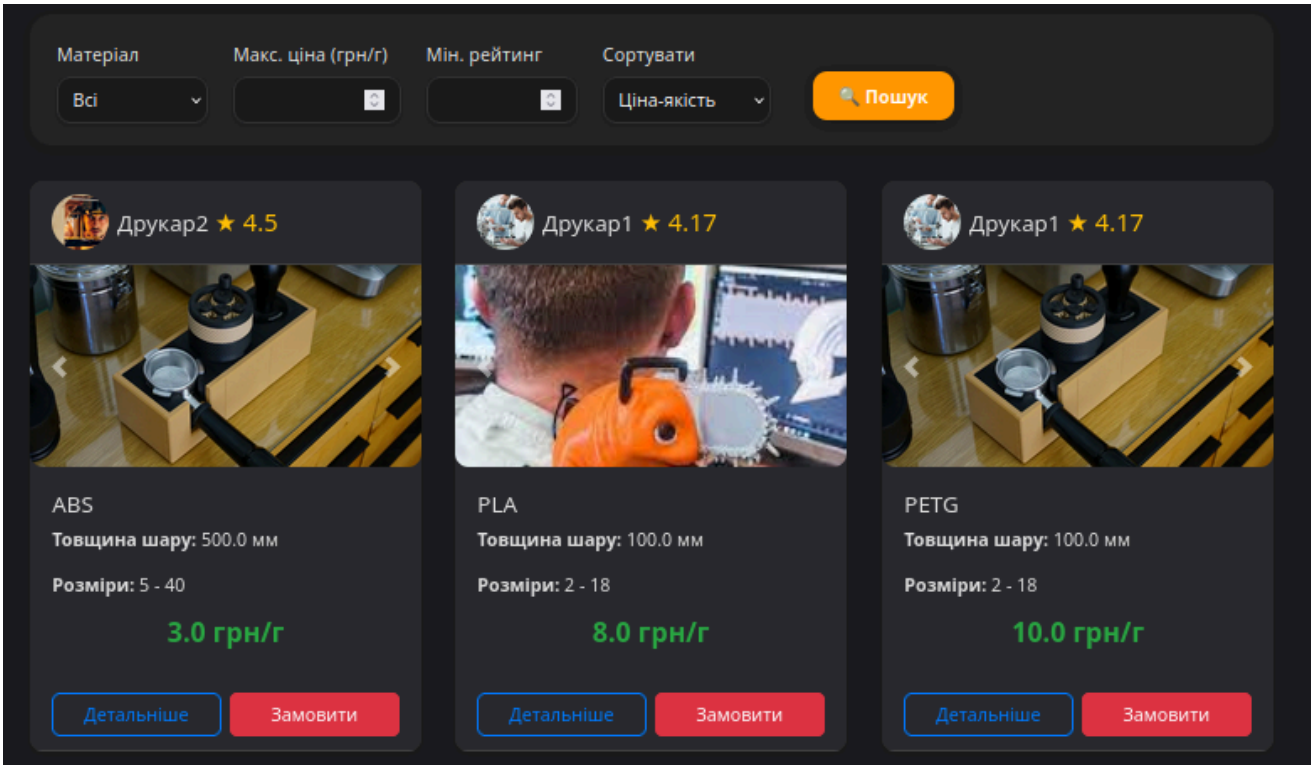
Малюнок 3.3.1 Демонстрація реєстрації

Після підтвердження реєстрації користувач входить у свій профіль і одразу потрапляє у каталог доступних оферів — тут можна побачити фото, ціну, матеріали, рейтинг кожного виконавця.



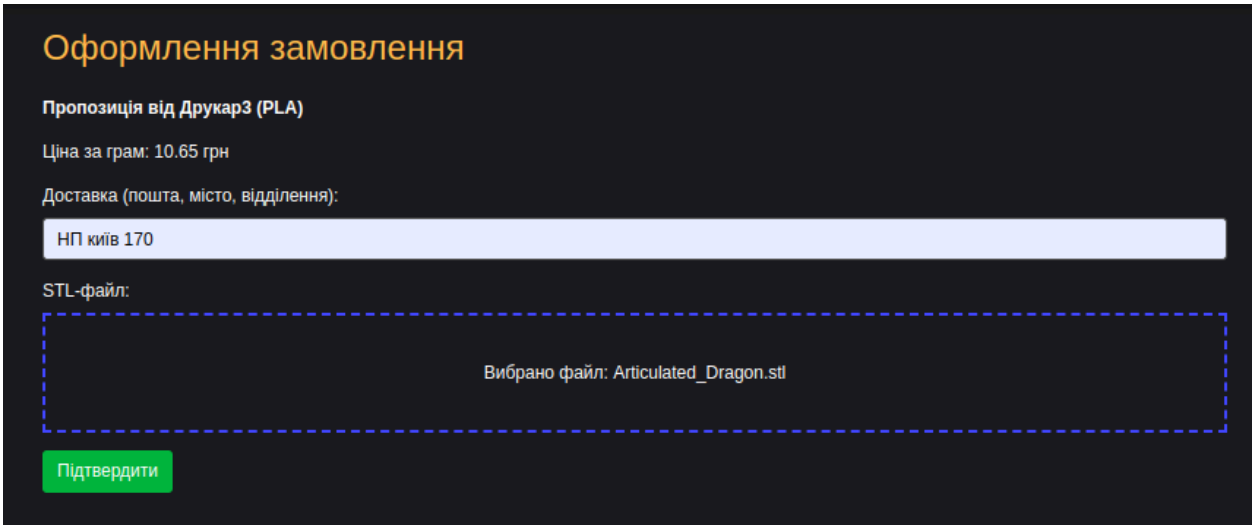
Малюнок 3.3.2 Демонстрація доски з доступними пропозиціями

Далі замовник може скористатися фільтрами по матеріалу, ціні чи рейтингу — це дає змогу швидко знайти оптимальний варіант серед усіх пропозицій.



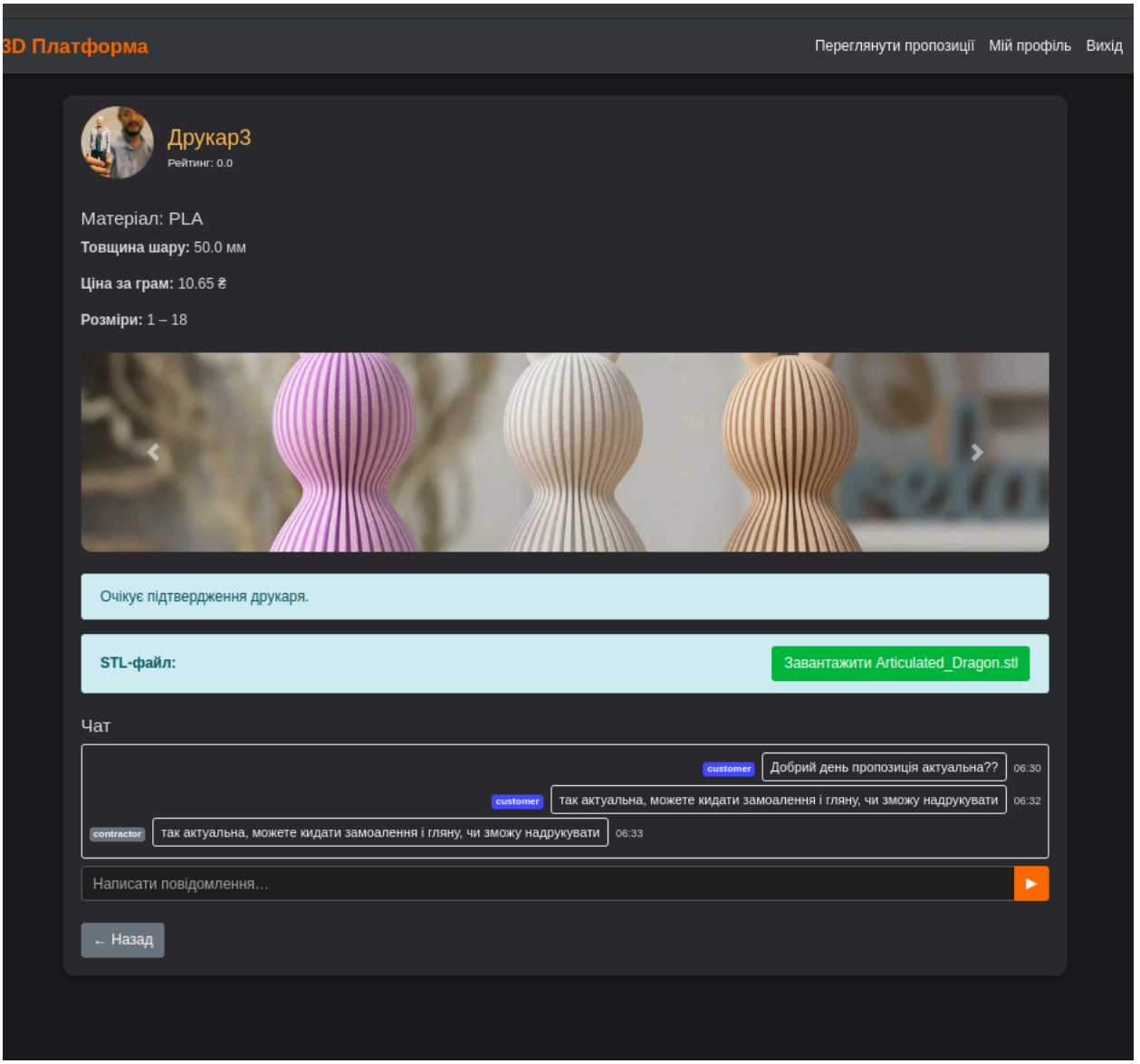
Малюнок 3.3.3 Демонстрація фільтрації

Обравши цікавий офер, користувач переходить на його детальну сторінку й натискає “Замовити”, після чого відкривається проста форма — потрібно завантажити STL-файл моделі, вказати деталі доставки чи коментар.



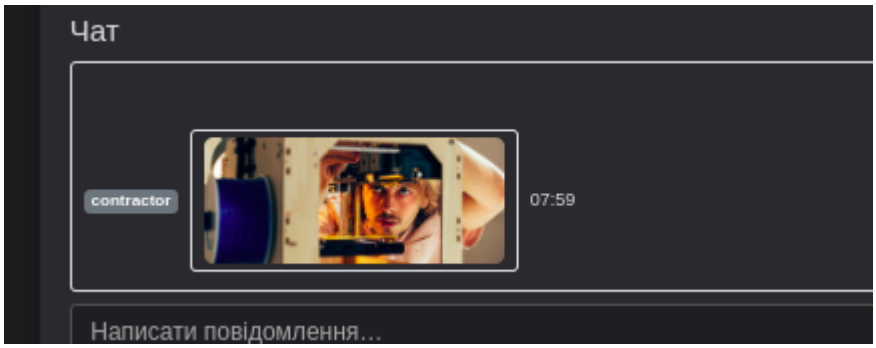
Малюнок 3.3.4 Демонстрація фільтрації

Після відправлення заявки замовник одразу потрапляє у чат із виконавцем — тут зручно уточнювати деталі, домовлятися про терміни чи матеріали, обмінюватися повідомленнями.



Малюнок 3.3.5 Демонстрація роботи чату

Коли друк завершено, друкар завантажує фото готової деталі — знімок відразу з’являється у чаті й на сторінці замовлення, тож замовник бачить, як виглядає його виріб ще до відправлення.



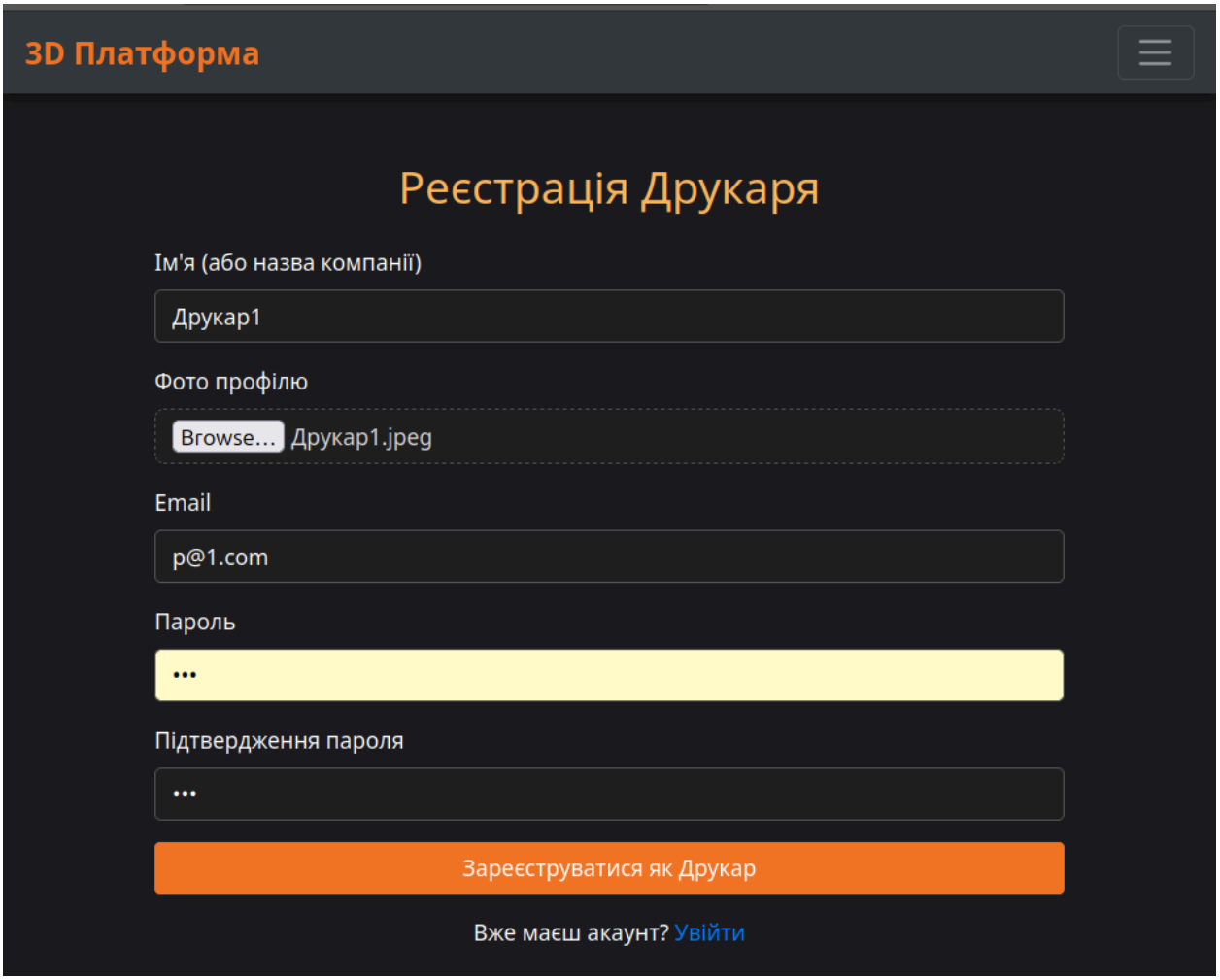
Малюнок 3.3.6 Демонстрація фотозвітності

Після підтвердження отримання користувач може оцінити роботу через просту форму із зірочками — це впливає на рейтинг виконавця.

Малюнок 3.3.7 Демонстрація елемнтів рейтинку

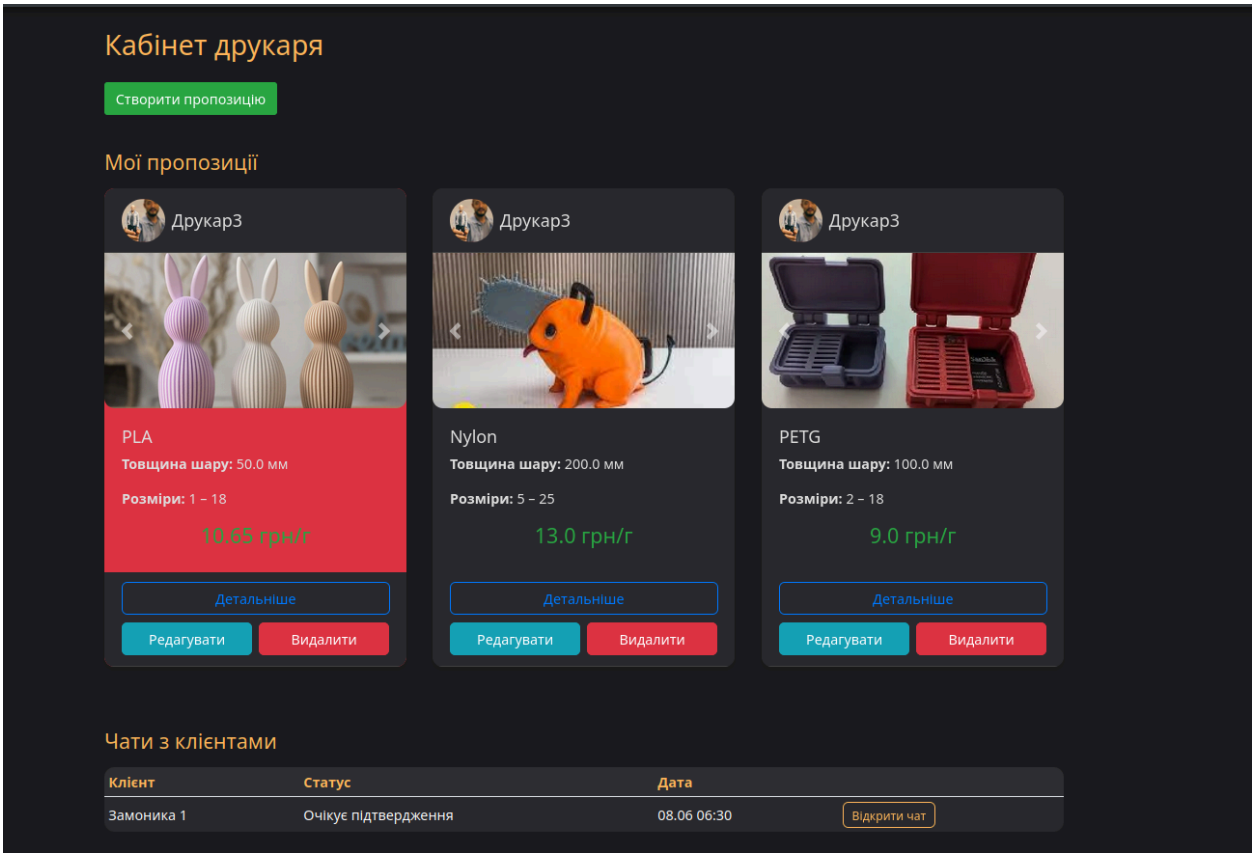
Для друкаря

Для виконавця все також починається з реєстрації, тільки тут вже треба вказати назву компанії або ім’я, email, пароль і можна додати свій логотип чи фото.



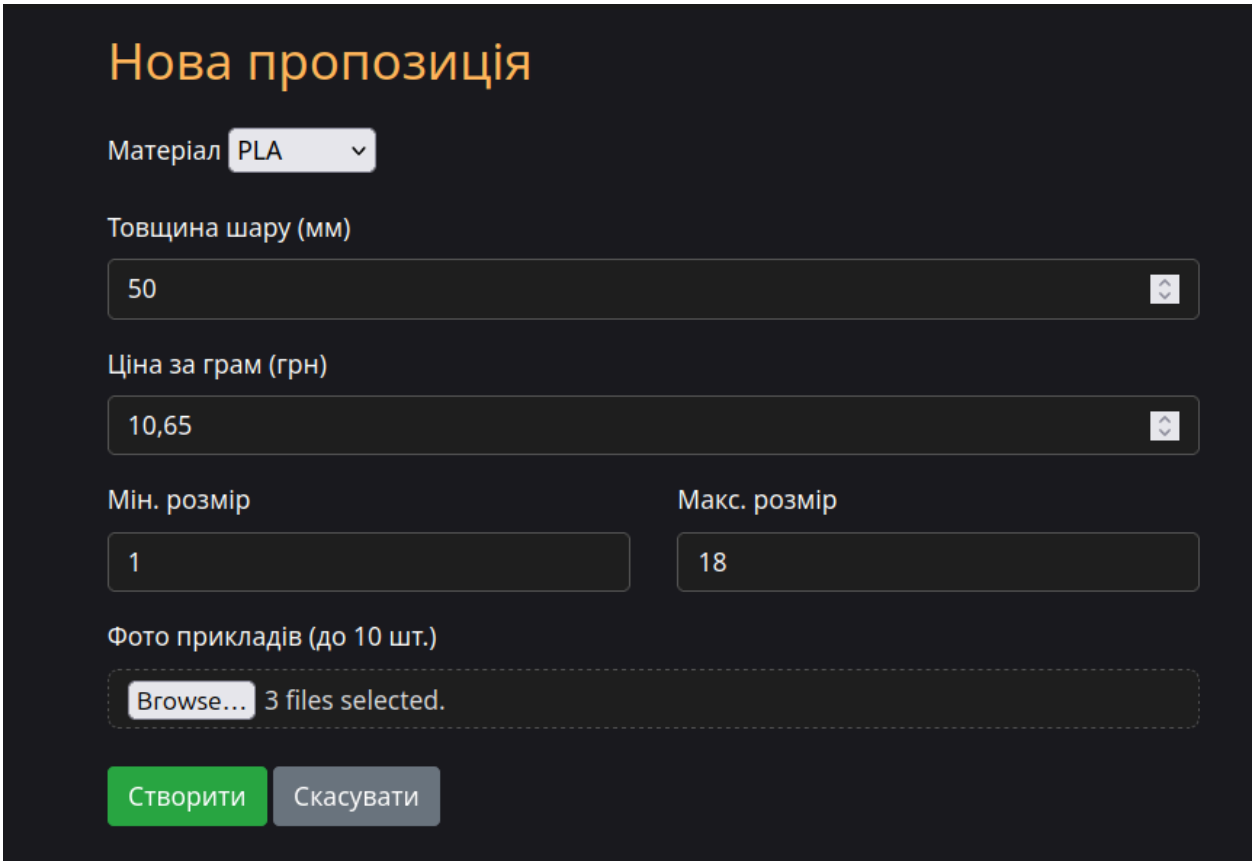
Малюнок 3.3.8 Демонстрація реєстрації

Після входу друкар бачить свою панель керування — тут є список усіх оферів та замовлень, кнопка для створення нової пропозиції.



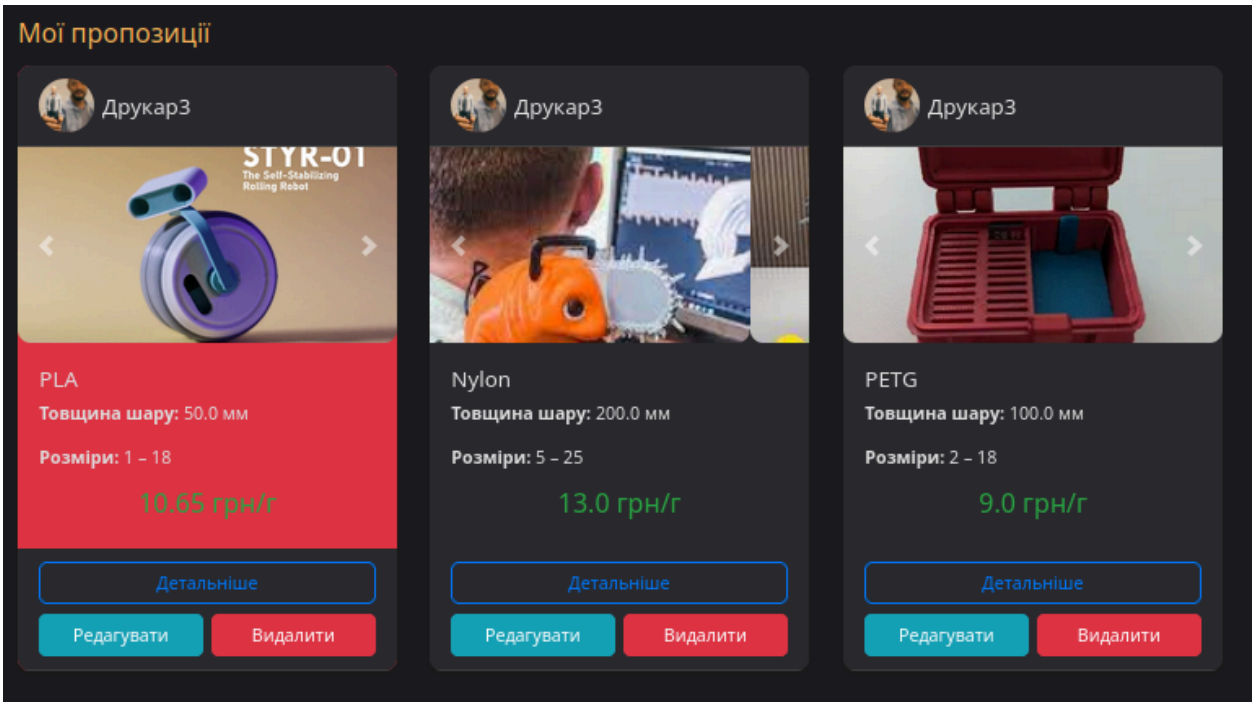
Малюнок 3.3.9 Демонстрація кабінета друкаря

Створення нового оферу — це заповнення форми з вибором матеріалу, ціни, розмірів та додаванням прикладів робіт (фото).



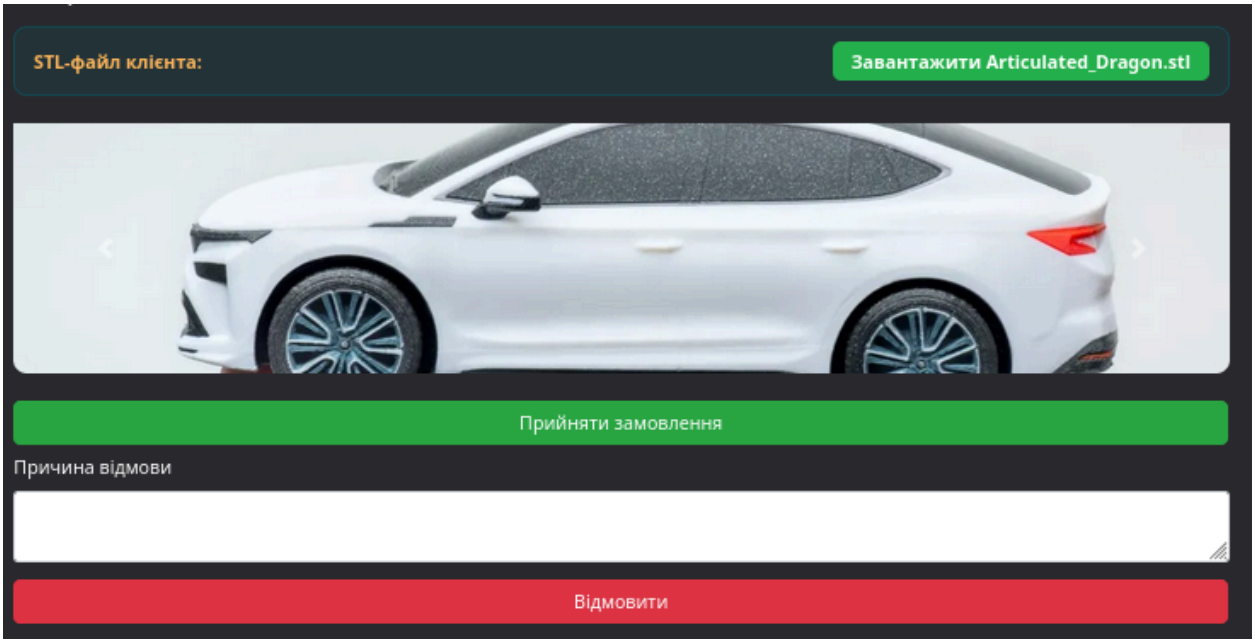
Малюнок 3.3.10 Демонстрація створення пропозицій

Коли на офер приходить замовлення, воно відображається у дашборді червоними кольором, що індукує про кинутий запит



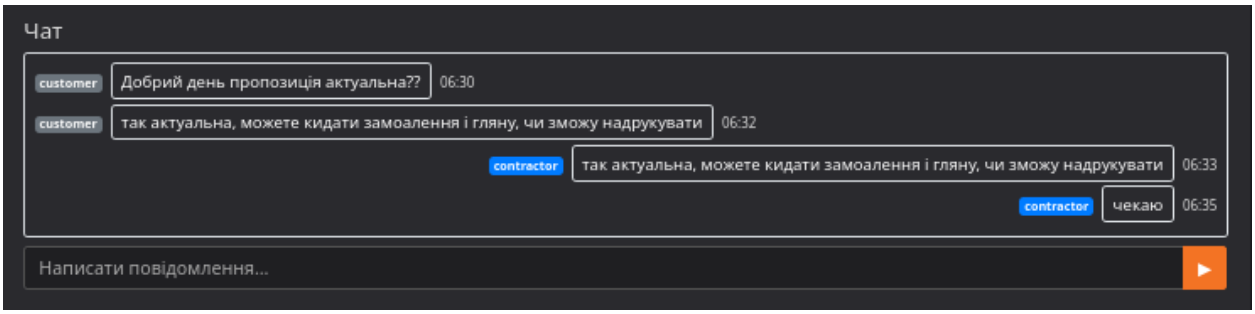
Малюнок 3.3.11 Демонстрація кольорової індекації

У друкаря є можливість прийнятт і відмови від замовлення



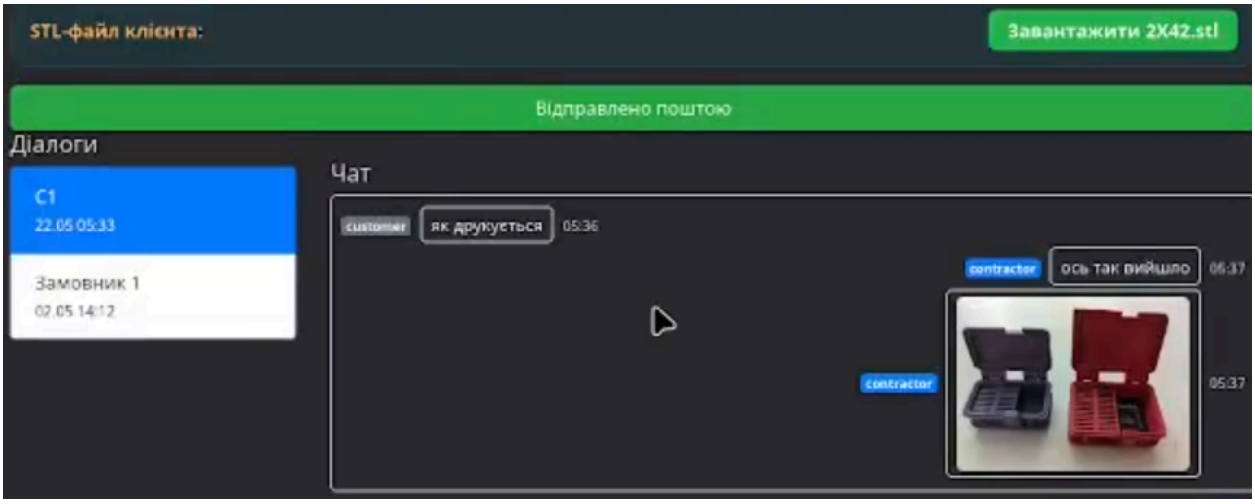
Малюнок 3.3.12 Демонстрація вибору прийняття завмовлення

У самому замовленні є чат для швидкого узгодження всіх питань.



Малюнок 3.3.13 *Демонстрація чату з боку друкаря*

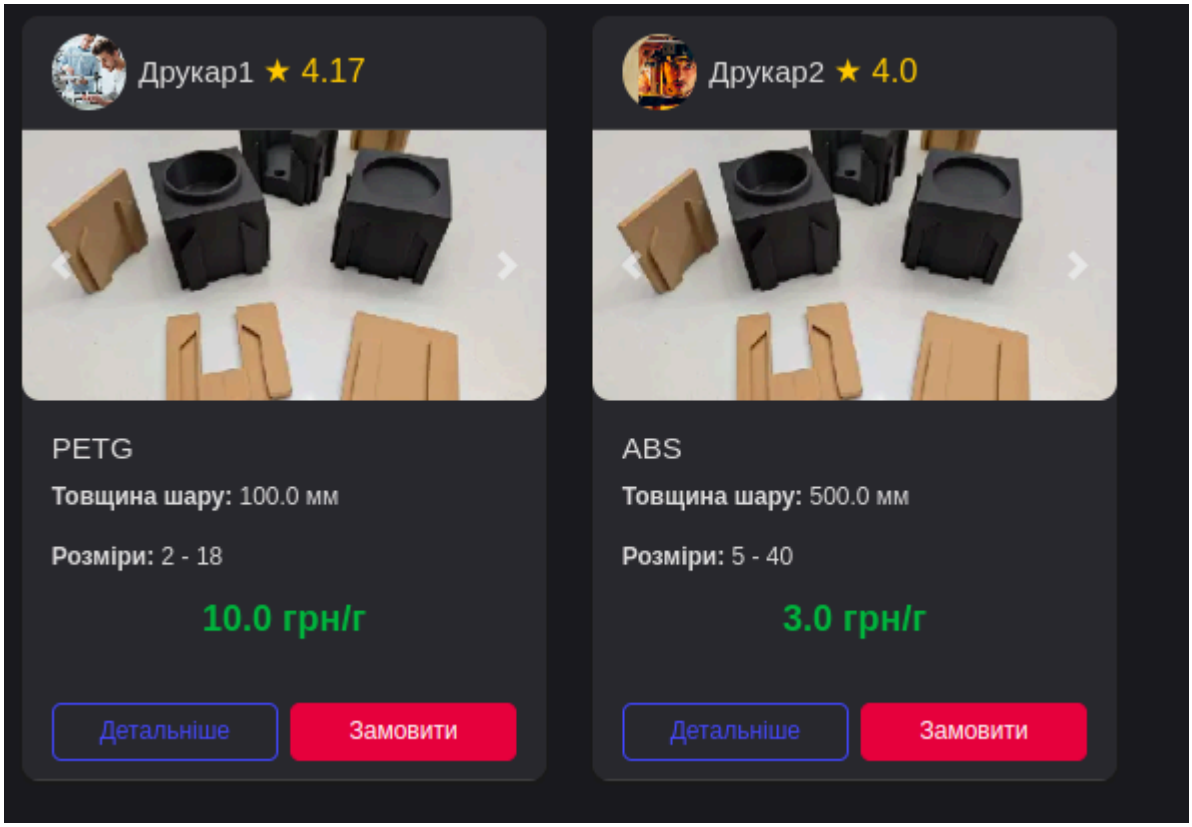
Після виконання друку друкар додає фото готової деталі через відповідну форму.



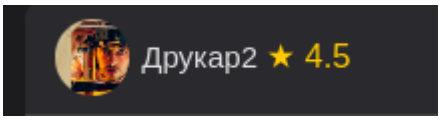
Малюнок 3.3.14 *Демонстрація відправки обрвязкового звіту*

Далі він змінює статус на “Друк завершено” або “Відправлено”,

А також при оцінці, одразу обраховується арифметичне сердній рейтинг



Малюнок 3.3.15 *Демонстрація початкового рейтингу до оцінки*



Малюнок 3.3.15 *Демонстрація кінцевого рейтингу після оцінки*

3.4. Результати тестування

Всі основні функції були протестовані. Таблиця результатів:

Функція	**Сценарій/Дія**	**Статус**	**Коментар**
Реєстрація	Успішна/з помилкою	OK	Валідація працює
Вхід	Вірний/невірний пароль	OK	Помилки коректно показує
Створення оферу	Новий/редагування/видалення	OK	Фото зберігається
Оформлення замовлення	STL, без STL, дублікати	OK	STL обов'язковий
Чат	Відправка, отримання	OK	Всі повідомлення відображ.

Рейтинг	Виставлення, повторне	OK	Неможливо оцінити двічі

Під час тестування були виявлені кілька дрібних багів:

1. Некоректна робота фільтрації при вводі нечислових значень у поле “ціна” — виправлено додатковою перевіркою на фронті.
2. Дублювання фото під час повторного завантаження оферу — додано обмеження до 10 фото.
3. Можливість зайти в чат без замовлення — закрито через перевірку прав доступу.

Після виправлення всі сценарії спрацювали коректно, система готова до розгортання для реальних користувачів.

Висновок до розділу «Постановка задачі та проєктування застосунку»

У цьому розділі я визначив головну мету — створити просту, зрозумілу та ефективну платформу для замовлення послуг 3D-друку онлайн. Детально проаналізовано потреби користувачів, основні функціональні вимоги, обрано оптимальний стек технологій та спроєктовано логічну архітектуру системи. Я описав, як кожен блок працює і взаємодіє між собою: від моделі бази даних до ролей, маршрутизації й інтерфейсу. Завдяки такому підходу платформа отримала зрозумілу структуру, простий спосіб розширення та зручність для різних типів користувачів. Усі ключові сценарії роботи вже враховано на етапі проєктування, тож подальша реалізація була швидкою і логічною.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено й впроваджено сучасну веб-платформу для онлайн-замовлення послуг 3D-друку, яка повністю відповідає заявленим вимогам і відображає сучасний рівень розвитку IT-сервісів. У ході роботи проведено аналіз предметної області та ринку, сформульовано функціональні та нефункціональні вимоги, спроєктовано архітектуру системи, розроблено модулі для роботи з користувачами, пропозиціями, замовленнями, чатом і системою оцінювання.

Платформа підтримує дві основні ролі: замовника та друкаря, що дозволяє максимально чітко розмежувати функціонал і забезпечити гнучкість у взаємодії. Реалізовано зручну процедуру реєстрації та входу, особисті кабінети для кожної ролі, можливість створювати й переглядати пропозиції, здійснювати замовлення з прикріпленням STL-файлів, вести комунікацію через вбудований чат та здійснювати оцінювання якості виконаних робіт. Особливу увагу приділено інтуїтивності інтерфейсу, простоті навігації та адаптивності дизайну завдяки використанню сучасних бібліотек і фреймворків.

Під час тестування всі основні сценарії (реєстрація, вхід, створення та редагування пропозицій, подання й обробка замовлень, обмін повідомленнями, виставлення оцінок) продемонстрували стабільність роботи платформи, відсутність критичних помилок і високу зручність для кінцевого користувача. Виявлені у процесі розробки й тестування незначні недоліки були оперативно усунуті.

Завдяки впровадженій системі рейтингів і фільтрації, користувачі мають можливість швидко знаходити надійних виконавців або вигідні пропозиції, що сприяє формуванню конкурентного й прозорого ринку 3D-друку в Україні. Запропонована архітектура та структура коду забезпечують легкість подальшого масштабування платформи, можливість додавання нових функцій (наприклад,

інтеграції з платіжними системами, підтримки додаткових технологій — ЧПК, лазерне різання тощо) та інтеграції із зовнішніми сервісами.

У підсумку, поставлені завдання дипломної роботи виконані повністю. Створена платформа не лише вирішує актуальні задачі ринку, а й є гнучкою основою для подальшого розвитку й адаптації до нових вимог галузі, сприяючи цифровій трансформації виробництва в Україні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 3D Printing Market worth $37.4 billion by 2029 – exclusive report by MarketsandMarkets [Електронний ресурс] // PR Newswire. – Режим доступу: https://www.prnewswire.com/news-releases/3d-printing-market-worth-37-4-billion-by-2029---exclusive-report-by-marketsandmarkets-302123055.html . – Дата доступу: 10.11.2024.
2. Яскевич, Владислав Олександрович (2023) *JavaScriptбібліотека React Навчальний посібник для студентів спеціальності Комп'ютерні науки* Київський університет імені Бориса Грінченка, Україна, Київ. https://elibrary.kubg.edu.ua/id/eprint/48587/
3. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.
4. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
5. Volunteers Worldwide With 3D Printers Are Aiding Ukraine's War [Електронний ресурс] // Forbes. – Режим доступу: https://www.forbes.com/sites/davidhambling/2024/06/07/how-volunteers-worldwide-are-helping-ukraines-war-with-3d-printers/. – Дата доступу: 22.02.2025.
6. Офіційний сайт волонтерської спільноти Druk Army [Електронний ресурс]. – Режим доступу: https://drukarmy.org.ua. – Дата доступу: 05.03.2025.

7. Printables now reaches 10 million monthly visits [Електронний ресурс] // Prusa Blog. – 2023. – Режим доступу: https://blog.prusa3d.com/support-the-designers-you-love-coming-soon-to-printables-com_78461/. – Дата доступу: 18.04.2025.
8. Офіційний сайт платформи Printables [Електронний ресурс]. – Режим доступу: https://www.printables.com. – Дата доступу: 29.03.2025.
9. Онлайн-спільнота моделей Thingiverse [Електронний ресурс]. – Режим доступу: https://www.thingiverse.com. – Дата доступу: 11.05.2025.
10. Приклад реалізації калькулятора, що вимагає реєстрації [Електронний ресурс] // 3D Device. – Режим доступу: https://3ddevice.com.ua/3d-друк/?srsltid=AfmBOoqGWj928wWEoqUVNF8B2fn7Ax5q1p1EhtXzo9QMP3kOMM2VxpcH. – Дата доступу: 27.02.2025.
11. Volnov Yevgen. PrintArmy: When Small Things Matter [Електронний ресурс] // UkraineWorld. – 29.08.2023. – Режим доступу: https://ukraineworld.org/en/articles/analysis/printarmy. – Дата доступу: 03.05.2025.
12. Ukraine 3D Printing High Performance Market (2025–2031) [Електронний ресурс] // 6Wresearch. – Режим доступу: https://www.6wresearch.com/industry-report/ukraine-3d-printing-high-performance-market. – Дата доступу: 17.01.2025.
13. How to Maximize Efficiency with 3D Printing Software: User Management and Workflow Management [Електронний ресурс] // 3DPrinterOS. – 02.02.2023. – Режим доступу: https://www.3dprinteros.com/articles/how-to-maximize-efficiency-with-3d-printing-software-user-management-and-workflow-management. – Дата доступу: 18.04.2025.

14. 10 UI Design Best Practices for Online Marketplaces 2024 [Електронний ресурс] // Fleexy. – Режим доступу: https://fleexy.dev/blog/10-ui-design-best-practices-for-online-marketplaces-2024/. – Дата доступу: 06.05.2025.
15. Human interface guidelines [Електронний ресурс] // Wikipedia. – Режим доступу: https://en.wikipedia.org/wiki/Human_interface_guidelines. – Дата доступу: 22.01.2025.
16. Flask Documentation [Електронний ресурс] // Pallets Projects. – Режим доступу: https://flask.palletsprojects.com/. – Дата доступу: 30.04.2025.
17. Flask-SQLAlchemy Documentation [Електронний ресурс]. – Режим доступу: https://flask-sqlalchemy.palletsprojects.com/. – Дата доступу: 12.03.2025.
18. Bootstrap 5 Documentation [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/. – Дата доступу: 05.02.2025.
19. Bootstrap Components (Forms, Modal, Carousel) [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.0/components/. – Дата доступу: 05.02.2025.
20. 47 важливих порад для UI та UX дизайнерів [Електронний ресурс] // UXpub. – Режим доступу: https://ux.pub/editorial/47-vazhlivikh-porad-dlia-ui-ta-ux-dizainieriv-4h0j. – Дата доступу: 12.05.2025.
21. Grinberg M. The Flask Mega-Tutorial – Part XV: A Better Application Structure [Електронний ресурс]. – Режим доступу: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-xv-a-better-application-structure. – Дата доступу: 15.03.2025.Flask-Login Documentation [Електронний ресурс]. – Режим доступу: https://flask-login.readthedocs.io/en/latest/. – Дата доступу:

27.04.2025.SQLAlchemy ORM Query Guide [Електронний ресурс]. – Режим доступу: https://docs.sqlalchemy.org/en/20/orm/queryguide/relationships.html. – Дата доступу: 19.02.2025.MVP in App Development [Електронний ресурс] // Softlancer. – Режим доступу: https://www.softlancer.co/blog/mvp-in-app-development. – Дата доступу: 24.01.2025.

22. Dynamic vs Static vs SPA [Електронний ресурс] // Academind. – Режим доступу: https://academind.com/tutorials/dynamic-vs-static-vs-spa. – Дата доступу: 29.03.2025.
23. How to Use Templates in a Flask Application [Електронний ресурс] // DigitalOcean. – Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-use-templates-in-a-flask-application. – Дата доступу: 10.04.2025.
24. What is an ORM? [Електронний ресурс] // Prisma Data Guide. – Режим доступу: https://www.prisma.io/dataguide/types/relational/what-is-an-orm. – Дата доступу: 05.05.2025.
25. Flask-Migrate Documentation [Електронний ресурс]. – Режим доступу: https://flask-migrate.readthedocs.io/en/latest/. – Дата доступу: 20.04.2025.
26. AJAX and Fetch API Crash Courses [Електронний ресурс] // The Dev Drawer. – Режим доступу: https://thedevdrawer.medium.com/ajax-and-fetch-api-crash-courses-82be25989bb5. – Дата доступу: 14.02.2025.
27. Star-rating-SVG Package [Електронний ресурс] // jsDelivr. – Режим доступу: https://www.jsdelivr.com/package/npm/star-rating-svg. – Дата доступу: 01.05.2025.
28. jsDelivr CDN [Електронний ресурс]. – Режим доступу: https://www.jsdelivr.com/. – Дата доступу: 08.03.2025.

29. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://sqlite.org/docs.html. – Дата доступу: 12.01.2025.
30. Ilinov K. 3d-platform prototype (GitHub repository) [Електронний ресурс]. – 2025. – Режим доступу: https://github.com/Kyrylo05/3d-platform. – Дата доступу: 25.05.2025.