НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.7:004.658_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інтегровані інформаційні системи»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: « Інформаційна система обміну товарами та послугами на основі P2P-архітектури з використанням технології блокчейн»

Виконав:
студент 2 курсу, групи ІА-42мп
Кухарук Олександр Сергійович _____

Керівник:
доцент каф. ІСТ, к.ф.-м.н., доц.
Рибачук Людмила Віталіївна _____

Рецензент:
доцент каф. ІПІ, к.т.н., доц.
Олійник Юрій Олександрович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти — другий (магістерський)

Спеціальність — 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Кухаруку Олександру Сергійовичу

1. Тема дисертації «Інформаційна система обміну товарами та послугами на основі P2P-архітектури з використанням технології блокчейн», науковий керівник дисертації Рибачук Людмила Віталіївна, к.ф.-м.н., доц., затверджені наказом по університету від «06» 11 2025 р. № 4841-с.

2. Термін подання студентом дисертації «15» 12 2025 р.

3. Об’єкт дослідження: процес обміну товарами та послугами в умовах децентралізованих P2P-систем з використанням блокчейн-технологій, включає вивчення архітектурних рішень для гібридних P2P-маркетплейсів, механізмів забезпечення довіри через Smart Contracts (Escrow), оптимізації продуктивності та економічної ефективності децентралізованих систем обміну.

4. Вихідні дані: безпека транзакцій через криптографічну цілісність Escrow SC та захист від Reentrancy-атак, децентралізація критичних операцій через блокчейн для зберігання стану угод та репутації, гібридна архітектура, що поєднує швидкість централізованих систем з безпекою блокчейну, off-chain API latency ≤150 мс, on-chain транзакція ≤10, економічна ефективність через мінімізацію Gas Cost, масштабованість до 1000 активних користувачів/хв, uptime ≥99.9%, Eventually Consistency з затримкою синхронізації ≤5 с, відповідність стандартам безпеки інформації та законодавству щодо захисту персональних даних.

5. Перелік завдань, які потрібно розробити: провести аналіз предметної області та огляд існуючих рішень, визначити функціональні та нефункціональні вимоги до системи, розробити гібридну архітектурну модель та провести моделювання функціональних процесів, спроєктувати логічну структуру системи та схему бази даних, реалізувати Escrow Smart Contract з оптимізацією Gas Cost, розробити серверну та клієнтську частини системи, розробити маркетингову стратегію та бізнес-план стартап-проєкту, оцінити ризики та оформити результати дослідження.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма прецедентів, діаграма послідовності процесу створення угоди, діаграма активності процесу виконання угоди через Escrow, діаграма станів Smart Contract, діаграма компонентів системи, діаграма класів, ER-діаграма бази даних, структурна схема гібридної архітектури системи, мережева діаграма взаємодії компонентів, діаграма безпеки системи, структурні схеми front-end та back-end складових, демонстраційні скріншоти роботи системи.

7. Дата видачі завдання 01.09.2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області та огляд існуючих рішень	01.09.25 – 22.09.25	
2.	Розробка технічного завдання та визначення вимог	22.09.25 – 29.09.25	
3.	Проєктування архітектури системи та моделювання	29.09.25 – 06.10.25	
4.	Розробка Smart Contract та серверної частини	06.10.25 – 27.10.25	
5.	Розробка клієнтської частини та інтеграція	27.10.25 – 24.11.25	
6.	Розробка маркетингової стратегії та бізнес-плану	24.11.25 – 01.12.25	
7.	Оформлення пояснювальної записки	01.12.25 – 08.12.25	
8.	Складання заліку	22.12.2025 - 26.12.2025	

Студент Олександр КУХАРУК

Науковий керівник Людмила РИБАЧУК

РЕФЕРАТ

Інформаційна система обміну товарами та послугами на основі P2P-архітектури з використанням технології блокчейн: 122 с., 24 табл., 21 рис., 9 дод., 16 джерел.

P2P-АРХІТЕКТУРА, БЛОКЧЕЙН, ГІБРИДНА СИСТЕМА, SMART CONTRACT, ESCROW, ДЕЦЕНТРАЛІЗАЦІЯ, TRUSTLESS-СИСТЕМИ, ІНФОРМАЦІЙНА СИСТЕМА, МАРКЕТПЛЕЙС, WEB3.

Актуальність — централізовані P2P-маркетплейси мають системні недоліки, зокрема SPOF та залежність від посередника, застосування блокчейну та смарт контрактів дозволяє забезпечити криптографічні гарантії виконання угод.

Мета роботи — підвищити безпеку та ефективність P2P обміну товарами і послугами шляхом створення інформаційної системи на основі гібридної P2P архітектури з інтеграцією блокчейн технологій.

Об’єкт дослідження — процеси обміну товарами та послугами в P2P економіці та інформаційні системи, що забезпечують таку взаємодію.

Предмет дослідження — архітектурні моделі та підходи до реалізації гібридних P2P маркетплейсів із використанням блокчейн технологій для забезпечення довіри та безпеки транзакцій.

Методи дослідження — системний і порівняльний аналіз централізованих та децентралізованих рішень, імітаційне моделювання для експериментальної оцінки продуктивності та економічної ефективності.

Наукова новизна — створено модель оптимального розподілу функціоналу між централізованими та децентралізованими компонентами в гібридній P2P архітектурі, відмінністю підходу є формалізація критеріїв визначення меж між on chain та off chain компонентами за принципами критичності до довіри та продуктивності.

Практичне значення — прототип є готовою основою для безпечного маркетплейсу з escrow механізмом і може бути використаний для комерційного впровадження.

ABSTRACT

Information system for exchange of goods and services based on P2P-architecture using blockchain technology: 122 p., 24 tab., 21 draw., 9 app., 16 sources.

P2P-ARCHITECTURE, BLOCKCHAIN, HYBRID SYSTEM, SMART CONTRACT, ESCROW, DECENTRALIZATION, TRUSTLESS SYSTEMS, INFORMATION SYSTEM, MARKETPLACE, WEB3.

Relevance — centralized P2P marketplaces have systemic drawbacks, including SPOF and dependence on an intermediary, the use of blockchain and smart contracts makes it possible to provide cryptographic guarantees of agreement execution.

Objective — to increase the security and efficiency of P2P exchange of goods and services by developing an information system based on a hybrid P2P architecture with blockchain integration.

Object of research — processes of exchange of goods and services in the P2P economy and information systems that provide such interaction.

Subject of research — architectural models and approaches to implementing hybrid P2P marketplaces using blockchain technologies to ensure trust and transaction security.

Research methods — systems and comparative analysis of centralized and decentralized solutions, simulation modeling for experimental evaluation of performance and economic efficiency.

Scientific novelty — a model for optimal distribution of functionality between centralized and decentralized components in a hybrid P2P architecture has been developed, the distinctive feature of the approach is the formalization of criteria for defining boundaries between on chain and off chain components based on trust criticality and performance principles.

Practical significance — the prototype is a ready foundation for a secure marketplace with an escrow mechanism and can be used for commercial implementation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API — Application Programming Interface (інтерфейс програмування додатків)

BNB — Binance Coin (BNB Smart Chain)

CORS — Cross-Origin Resource Sharing (обмін ресурсами між джерелами)

CSS — Cascading Style Sheets (каскадні таблиці стилів)

DeFi — Decentralized Finance (децентралізовані фінанси)

DMP — Decentralized Marketplace Platform (децентралізований маркетплейс)

ER — Entity-Relationship (сутність-зв'язок)

EVM — Ethereum Virtual Machine (віртуальна машина Ethereum)

GDPR — General Data Protection Regulation (загальний регламент захисту даних)

HTML — HyperText Markup Language (мова розмітки гіпертексту)

HTTP — HyperText Transfer Protocol (протокол передачі гіпертексту)

HTTPS — HyperText Transfer Protocol Secure (захищений протокол передачі гіпертексту)

IT — Information Technology (інформаційні технології)

JSON — JavaScript Object Notation (нотація об'єктів JavaScript)

JWT — JSON Web Token (веб-токен JSON)

P2P — Peer-to-Peer (рівний до рівного)

PoS — Proof of Stake (доказ частки)

PoW — Proof of Work (доказ роботи)

REST — Representational State Transfer (передача стану представлення)

SaaS — Software as a Service (програмне забезпечення як послуга)

SC — Smart Contract (розумний контракт)

SPOF — Single Point of Failure (єдина точка відмови)

SSL/TLS — Secure Sockets Layer / Transport Layer Security (протоколи безпеки)

UML — Unified Modeling Language (уніфікована мова моделювання)

UI — User Interface (інтерфейс користувача)

UX — User Experience (досвід користувача)

WebRTC — Web Real-Time Communication (веб-комунікація в реальному часі)

WebSocket — протокол двостороннього зв'язку

Web3 — третє покоління веб-технологій на базі блокчейну

ВСТУП

У сучасному світі, де цифровізація охоплює різні сфери економіки, обмін товарами та послугами між користувачами також потребує нових технологічних підходів. Традиційна модель централізованих P2P-маркетплейсів базується на посередництві платформи, що стягує високі комісії, контролює транзакції та може здійснювати цензуру, що часто спричиняє залежність користувачів від однієї точки відмови (SPOF), непрозорість процесів та втрату контролю над власними даними. Тому актуальною стає потреба у створенні децентралізованої інформаційної системи, яка дозволяє забезпечити безпеку транзакцій через блокчейн-технології, усунути системні ризики централізованих платформ та покращити економічну ефективність P2P-обміну.

Розвиток блокчейн-технологій та децентралізованих систем відкриває нові можливості для створення справедливих та безпечних платформ обміну. Використання Smart Contracts дозволяє автоматизувати виконання угод, забезпечити криптографічні гарантії безпеки транзакцій, створити прозору систему репутації та усунути необхідність довіри до посередника. Однак чисті децентралізовані рішення мають обмеження щодо продуктивності та масштабованості. Гібридна архітектура, що поєднує швидкість централізованих систем з безпекою блокчейну, є актуальним напрямком для інформаційних систем обміну товарами та послугами.

Метою роботи є розробка та реалізація інформаційної системи обміну товарами та послугами на основі гібридної P2P-архітектури з інтеграцією блокчейн-технологій для забезпечення безпеки транзакцій та усунення системних ризиків централізованих платформ.

Для досягнення мети поставлено такі завдання:

– провести системний аналіз централізованих та децентралізованих P2P-маркетплейсів та визначити їхні переваги та недоліки;

– обґрунтувати вибір гібридної архітектурної моделі та методологію розподілу функціоналу між централізованими та децентралізованими компонентами;

– спроєктувати функціональну та нефункціональну архітектуру системи з використанням UML-моделювання;

– розробити Escrow Smart Contract для автоматизації угод та забезпечення безпеки транзакцій;

– реалізувати прототип системи з інтеграцією централізованих компонентів (сервер, база даних) та децентралізованих (блокчейн, Smart Contract);

Об'єктом дослідження є процеси обміну товарами та послугами в P2P економіці та інформаційні системи, що забезпечують таку взаємодію.

Предметом дослідження є архітектурні моделі та методи реалізації гібридних P2P-маркетплейсів з використанням блокчейн-технологій для забезпечення безпеки транзакцій та усунення системних ризиків.

Методами дослідження є системний аналіз для дослідження архітектурних моделей, порівняльний аналіз для оцінки переваг та недоліків існуючих рішень, UML-моделювання для проєктування функціональних процесів та архітектури системи, методологія розробки програмного забезпечення для реалізації компонентів, імітаційне моделювання для експериментальної оцінки продуктивності та економічної ефективності.

У першому розділі розглянуто предметну область та теоретичні засади децентралізованих систем, проаналізовано проблематику P2P-економіки в умовах централізованих платформ, досліджено концептуальні основи технології блокчейн та механізми Smart Contracts для автоматизації угод.

Другий розділ присвячено огляду існуючих рішень, критичному аналізу архітектур децентралізованих P2P-маркетплейсів, порівнянню аналогів за критеріями продуктивності та надійності, обґрунтуванню вибору гібридної моделі та програмно-технічної платформи.

У третьому розділі описано функціональні етапи проєктування та моделювання системи: декомпозицію функціональних вимог, розробку

гібридної архітектурної моделі, моделювання функціональних процесів та проєктування логічної структури системи.

Четвертий розділ містить нефункціональні етапи реалізації: визначення вимог до безпеки, продуктивності та надійності, реалізацію Escrow Smart Contract, методику проведення імітаційного моделювання та експериментальної оцінки.

П'ятий розділ присвячено розробці P2P-маркетплейсу: опису серверної та клієнтської частин системи, використаних технологій та демонстрації роботи системи.

Шостий розділ представлено у вигляді стартап-проєкту, що розкриває опис ідеї проєкту, технологічний аудит, аналіз ринкових можливостей запуску та маркетингову стратегію розробленої системи.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕОРЕТИЧНИХ ЗАСАД ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМ

1.1. Сутність та проблематика P2P-економіки в умовах централізованих платформ

1.1.1. Системний аналіз централізованих P2P-маркетплейсів (ЦМП)

P2P-економіка (Peer-to-Peer, від особи до особи) є парадигмою обміну, в якій взаємодія щодо товарів, послуг або активів відбувається безпосередньо між двома кінцевими користувачами [1]. У контексті інформаційних систем (ІС), реалізація цієї взаємодії найчастіше відбувається через посередника – Централізований Маркетплейс (ЦМП). Прикладами таких систем є великі комерційні платформи, що надають лише інфраструктуру для пошуку та комунікації, але утримують повний контроль над транзакційним середовищем та даними [2].

З точки зору архітектури ІС, ЦМП використовують класичну трирівневу модель (клієнт-сервер-база даних). Центральний сервер є ядром системи, що виконує функції агрегації даних, авторизації, арбітражу та модерації.

Така архітектура надає значні переваги у швидкості розгортання та зручності користувача (UX/UI). Однак, магістерське дослідження вимагає критичного системного аналізу структурних недоліків, які породжують високі ризики для користувачів та економічної стійкості:

– проблема єдиної точки відмови (Single Point of Failure, SPOF) — центральна база даних та сервери є критичним активом, вихід з ладу або компрометація якого (через кібератаки або технічні збої) призводить до непрацездатності всієї системи та потенційної втрати всіх даних користувачів;

– проблема Високих Транзакційних Витрат та Комісій: Оскільки ЦМП є монополістом у наданні послуги довіри та арбітражу, він має можливість встановлювати високі комісійні збори (Fees), які включають не лише операційні витрати, а й маржу посередника, це знижує економічну ефективність P2P-обміну для кінцевих користувачів;

– проблема Довіри до Посередника (Intermediary Trust) — користувачі повинні повністю довіряти ЦМП, що той буде чесно виконувати функції арбітра, правильно зберігати особисті дані (GDPR-ризики) та об'єктивно підтримувати систему рейтингів, ця довіра ґрунтується на юридичних та репутаційних механізмах, а не на криптографічних гарантіях, що створює передумови для маніпуляцій.

Таким чином, незважаючи на поширеність, архітектура ЦМП не відповідає сучасним вимогам до розподіленості, прозорості та стійкості. Це створює необхідність дослідження та розробки альтернативних рішень, що інтегрують технології децентралізованих розподілених реєстрів.

1.1.2. Ідентифікація системних ризиків: SPOF, цензура та непрозорість довіри

Системний аналіз ЦМП, проведений у п. 1.1.1, дозволяє чітко ідентифікувати ключові системні ризики, які безпосередньо впливають на надійність та економічну стійкість платформ P2P-обміну. Ці ризики можуть бути класифіковані як архітектурні, операційні та ризики довіри.

SPOF є фундаментальним недоліком централізованої архітектури. У контексті ЦМП, це означає, що відмова, компрометація або виведення з ладу центральної бази даних або обчислювального вузла призводить до повної зупинки функціонування всієї системи. Наслідки SPOF виходять за межі простого простою (downtime):

– втрата цілісності та конфіденційності даних — усі користувацькі записи, транзакційні історії та, що критично важливо, особисті дані, зберігаються централізовано, успішна атака на цей вузол призводить до масового витоку даних, що є порушенням регуляторних норм (наприклад, GDPR);

– відсутність відмовостійкості — резервування даних, хоч і застосовується, не гарантує безперервної доступності сервісу в умовах потужних DDoS-атак або значних регіональних збоїв інфраструктури.

Операційний Ризик є наслідком монополії на арбітраж та модерацію. Центральний оператор має можливість в односторонньому порядку:

– цензурувати контент та угоди — блокування, видалення або модифікація оголошень, що суперечать внутрішній політиці платформи, навіть якщо вони є законними, це становить проблему свободи економічної діяльності;

– заморожувати активи та блокувати користувачів: Втручання в фінансові потоки та арбітражні спори, що може призвести до неправомірного блокування коштів користувачів без прозорого механізму оскарження;

– рішення посередника є остаточним, оскільки відсутній механізм, заснований на об'єктивних, автоматизованих та незмінних правилах.

Ефективність будь-якої P2P-системи критично залежить від системи довіри та репутації. У ЦМП ця система є непрозорою (Trust-based, а не Trustless), оскільки:

– рейтингові дані зберігаються централізовано, що дозволяє теоретичну можливість їхньої внутрішньої модифікації або маніпуляції (наприклад, видалення негативних відгуків для пріоритетних продавців);

– система довіри вразлива до атаки Sybil, коли зловмисники створюють множинні фальшиві облікові записи для штучного підвищення або зниження репутації;

– відсутність криптографічно підтвердженої історії транзакцій унеможливлює надійну верифікацію учасників.

Таким чином, для створення стійкої, прозорої та економічно ефективної інформаційної системи обміну необхідний перехід до архітектурного рішення, яке мінімізує вплив централізованого посередника. Технологія блокчейн, що є предметом подальшого дослідження, пропонує механізми (розподілений реєстр та Smart Contracts) для перенесення функцій довіри та арбітражу з центрального вузла на децентралізовану мережу.

1.2. Концептуальні основи технології блокчейн для забезпечення довіри

1.2.1. Архітектурні принципи та механізми консенсусу (PoS/PoW)

Технологія блокчейн (Blockchain) є різновидом технології розподіленого реєстру (DLT), що архітектурно вирішує проблему довіри в P2P-взаємодіях шляхом перенесення функцій верифікації та зберігання даних від центрального посередника до децентралізованої мережі вузлів [3]. Основними архітектурними принципами, які забезпечують цю трансформацію, є:

– відсутність єдиної центральної точки контролю, кожен вузол мережі зберігає копію реєстру, що усуває проблему SPOF (Single Point of Failure) [4];

– дані організовані у блоки, які криптографічно пов'язані між собою у ланцюг за допомогою хеш-функцій, після запису транзакції в блок змінити її постфактум неможливо без зміни всіх наступних блоків, що робить маніпуляції економічно та обчислювально недоцільними;

– хоча особисті дані користувачів можуть бути псевдонімними, усі транзакції та правила їх виконання (Smart Contracts) є відкритими та публічно перевіряються.

Ключовим елементом архітектури блокчейну є механізм консенсусу. Його роль полягає у забезпеченні узгодженості стану розподіленого реєстру та захисті від зловмисних дій. У контексті розробки інформаційних систем (спеціальність 126), вибір механізму консенсусу критично впливає на нефункціональні вимоги: продуктивність, масштабованість та економічну ефективність.

Найпоширеніші механізми консенсусу:

– Proof-of-Work (PoW) — механізм, заснований на обчислювальній роботі (майнінгу) для створення нового блоку. Він забезпечує високу безпеку та стійкість до атаки 51%, але має низьку швидкість підтвердження транзакцій (*Latency*) та високі транзакційні витрати (*Gas Cost*) [5];

– Proof-of-Stake (PoS) — механізм, де валідатори обираються пропорційно до кількості криптоактивів, які вони "застейкали" (заблокували) як забезпечення. PoS пропонує значно вищу швидкість транзакцій (вищий *Throughput*), нижчі енерговитрати та значно менші Gas Costs, що робить його архітектурно

привабливішим для створення децентралізованих маркетплейсів з великою кількістю мікротранзакцій.

Вибір PoS або його модифікацій (наприклад, Delegated Proof-of-Stake, DPoS) є більш обґрунтованим для архітектури інформаційної системи обміну, оскільки пріоритетом є економічна ефективність та швидкість підтвердження P2P-угод, а не лише абсолютна криптографічна стійкість як у фінансових системах.

1.2.2. Поняття "Trustless-системи" та їхня роль у P2P-обміні

У контексті інформаційних систем, ключовим теоретичним перевагою інтеграції блокчейну є перехід від Trust-based (системи, засновані на довірі до посередника) до Trustless-систем (системи, які не потребують довіри до жодного з учасників чи посередників). Цей концептуальний зсув є фундаментальним для розробки стійкої та справедливої P2P-платформи обміну.

Trustless-система — це архітектурне рішення, де функція довіри замінюється криптографічними гарантіями та програмним кодом. Замість того, щоб покладатися на репутацію центрального органу (як у ЦМП), користувачі покладаються на математичну доведеність та незмінність логіки, що зафіксована у розподіленому реєстрі.

Роль Trustless-систем у P2P-обміні:

– усунення ризику цензури — правила взаємодії та модерації (закладені у Smart Contracts) є публічними та не можуть бути змінені однією стороною, це усуває можливість одностороннього блокування користувачів чи заморожування активів, забезпечуючи архітектурний захист свободи торгівлі;

– забезпечення прозорості та аудиту — усі транзакції та зміни стану угоди фіксуються у незмінному реєстрі, кожен учасник мережі може верифікувати історію без звернення до центрального посередника, що мінімізує ризик шахрайства;

– створення істинної системи довіри — традиційні рейтинги в ЦМП вразливі до маніпуляцій (атака Sybil), у Trustless-системах репутація може бути пов’язана з

криптографічно підтвердженими успішними транзакціями, це дозволяє формувати надійну, незмінну репутацію користувача, що є об'єктивнішою мірою довіри у P2P-обміні [6];

– автоматизація угод — замість ручного контролю та арбітражу, угоди виконуються автоматично, коли виконуються заздалегідь визначені програмні умови, ця функція реалізується через Smart Contracts, які стають ключовим елементом архітектури ІС.

Таким чином, для створення стійкої інформаційної системи обміну, що відповідає сучасним вимогам до децентралізації та безпеки, необхідно архітектурно впровадити принципи Trustless-моделі, використовуючи можливості технології блокчейн.

1.3. Дослідження механізмів Smart Contracts для автоматизації угод (Escrow-механізм)

1.3.1. Функціональні можливості та обмеження Smart Contracts у сфері обміну

Smart Contract (Смарт-контракт) є фундаментальним архітектурним елементом, який дозволяє реалізувати принципи Trustless-системи на рівні застосунків [7]. З точки зору інформаційних систем, смарт-контракт являє собою автономний, самовиконуваний код, що зберігається та виконується в розподіленому реєстрі (блокчейні). Він автоматично виконує умови угоди, прописані в коді, коли виконуються заздалегідь визначені події, усуваючи необхідність у посереднику для верифікації та виконання домовленостей.

Функціональні можливості у сфері обміну:

– автоматизація виконання угод — смарт-контракти можуть безпосередньо управляти цифровими активами, у контексті P2P-обміну вони дозволяють програмувати логіку транзакцій (наприклад, умовне депонування коштів, автоматичний розрахунок комісій, перерахування активів після підтвердження отримання товару), забезпечуючи виконання зобов'язань обома сторонами;

– забезпечення незмінності логіки, на відміну від коду на централізованому сервері, логіка Smart Contract після розгортання є незмінною (Immutable), це гарантує, що правила, погоджені сторонами на момент укладання угоди, не можуть бути змінені оператором системи або іншою третьою стороною;

– створення децентралізованих систем довіри — смарт-контракти можуть автоматично записувати результат угоди (успіх/неуспіх) у блокчейн, формуючи прозору та незмінну історію репутації користувачів, це є основою для розробки надійного рейтингового механізму, захищеного від маніпуляцій.

Хоча Smart Contracts є потужним інструментом, їхнє застосування в ІС обміну має низку суттєвих обмежень, які необхідно враховувати на етапі проєктування:

– неможливість внесення змін (Bugs are Permanent) — незмінність коду означає, що будь-яка помилка, вразливість або непередбачена умова, закладена в логіку контракту, не може бути виправлена після розгортання, це вимагає надзвичайно ретельного аудиту та тестування на етапах функціонального проєктування;

– проблема оракулів (Oracle Problem) — смарт-контракти оперують лише даними, які вже знаходяться в блокчейні, для угод обміну товарами та послугами часто потрібна інформація із зовнішнього світу (наприклад, підтвердження доставки, якість товару), ця залежність від зовнішніх джерел (*оракулів*) створює нову точку централізації та вимагає впровадження надійних децентралізованих рішень для подачі зовнішніх даних;

– обмеження продуктивності та вартості (Gas Cost) — виконання коду Smart Contract вимагає плати — газу (Gas), складні операції або велика кількість транзакцій можуть спричинити високі витрати та низьку швидкість підтвердження (*Latency*), що є критичним нефункціональним обмеженням для масового P2P-маркетплейсу.

З огляду на ці обмеження, подальше дослідження буде присвячене конкретному механізму — Escrow Smart Contract — який є центральним у

забезпеченні безпеки обміну та потребує оптимізації для подолання проблеми Gas Cost.

1.3.2. Аналіз алгоритмів Escrow-механізмів у децентралізованих додатках

Для ефективної та безпечної реалізації обміну товарами та послугами в децентралізованій P2P-архітектурі критично важливим є використання механізму Escrow (умовного депонування). У традиційних системах Escrow виступає центральний посередник, який утримує активи до моменту, коли обидві сторони підтвердять виконання своїх зобов'язань. У Trustless-системі ця функція переноситься на Escrow Smart Contract.

Escrow Smart Contract — це програмний алгоритм, який фіксує кошти покупця на блокчейні та автоматично звільняє їх продавцю лише після виконання заздалегідь визначених умов (наприклад, підтвердження доставки або відсутності претензій протягом терміну). Цей механізм архітектурно вирішує ризик контрагента (counterparty risk), оскільки усуває необхідність довіряти безпосередньо іншій стороні чи центральному посереднику [8].

Класифікація та аналіз алгоритмів Escrow:

– двосторонній Escrow (Two-Party Escrow) — найпростіший алгоритм, де кошти звільняються лише за умови спільної згоди обох сторін (покупця та продавця), його перевага — мінімальні витрати Gas Cost, але він має суттєвий недолік у вирішенні спорів: у разі конфлікту (відсутність згоди) кошти залишаються заблокованими на контракті назавжди;

– багатосторонній Escrow з Арбітром (Multi-Party Escrow with Arbitrator) — цей алгоритм є основою для багатьох децентралізованих маркетплейсів, у ньому бере участь третя, незалежна сторона — арбітр;

– Hashed Time-Lock Contracts (HTLCs) — переважно використовуються для атомарних свопів (обміну криптовалют) і не повністю застосовні для обміну фізичними товарами, оскільки вимагають наявності криптографічного секрету для завершення транзакції.

Висновки до розділу 1

У розділі 1 проаналізовано предметну область та теоретичні засади децентралізованих систем для P2P-обміну товарами та послугами. Проведено системний аналіз архітектурних моделей та визначено теоретичні основи для подальшого проєктування гібридної інформаційної системи.

Визначено системні проблеми централізованих P2P-маркетплейсів: єдина точка відмови (SPOF), високі комісії, цензура та непрозорість системи довіри. Ці обмеження знижують надійність, економічну ефективність та свободу економічної діяльності користувачів. Централізовані платформи створюють залежність від посередника, що суперечить принципам децентралізованої економіки та обмежує можливості справедливого P2P-обміну.

Розглянуто концептуальні основи блокчейну: децентралізація, незмінність, прозорість. Ці принципи забезпечують архітектурну основу для усунення системних ризиків централізованих платформ. Механізми консенсусу (PoW та PoS) проаналізовано з точки зору продуктивності, масштабованості та економічної ефективності. Для P2P-маркетплейсів PoS є більш прийнятним через вищу швидкість транзакцій, нижчі енерговитрати та менші транзакційні витрати, що критично важливо для мікротранзакцій та масового використання.

Визначено перехід від Trust-based до Trustless-систем як основу для створення стійких P2P-платформ. Trustless-системи забезпечують криптографічні гарантії замість довіри до посередника, усувають ризик цензури та забезпечують прозорість та аудитування транзакцій. Цей концептуальний зсув є фундаментальним для розробки справедливих та безпечних платформ обміну, де правила взаємодії зафіксовані в незмінному програмному коді, а не залежать від довіри до центрального органу.

Досліджено механізми Smart Contracts для автоматизації угод, зокрема Escrow-механізм. Визначено функціональні можливості (автоматизація виконання угод, незмінність логіки, створення децентралізованих систем довіри)

та обмеження (неможливість внесення змін після розгортання, проблема оракулів для зовнішніх даних, обмеження продуктивності та вартості через Gas Cost). Проаналізовано алгоритми Escrow-механізмів: двосторонній Escrow (мінімальні витрати, але проблеми з вирішенням спорів), багатосторонній Escrow з арбітром (надійність, але складність реалізації) та HTLCs (для атомарних свопів), визначено їх переваги та недоліки в контексті P2P-обміну товарами та послугами.

Для створення стійкої, прозорої та економічно ефективної інформаційної системи обміну необхідно інтегрувати технологію блокчейну з механізмами Smart Contracts, зокрема Escrow-механізмом. Це дозволить усунути недоліки централізованих платформ та забезпечити децентралізовану, безпечну та справедливу P2P-взаємодію з криптографічними гарантіями виконання угод. Проведений аналіз створює теоретичну основу для подальшого проєктування та розробки гібридної архітектури інформаційної системи обміну, що поєднує переваги централізованих систем (швидкість, зручність) та децентралізованих систем (безпека, прозорість, відсутність цензури).

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1. Критичний аналіз архітектур децентралізованих P2P-маркетплейсів

2.1.1. Аналіз чистої P2P-архітектури (на прикладі Bisq, OpenBazaar)

Пошук альтернативи Централізованим Маркетплейсам (ЦМП) природно привів до розробки платформ, заснованих на чистій P2P-архітектурі. Така модель прагне до максимально можливої децентралізації, усуваючи не лише посередника, а й єдиний сервер, який може виступати точкою контролю. З архітектурної точки зору, чиста P2P-система функціонує як мережа рівноправних вузлів, де кожен учасник (комп'ютер) одночасно є клієнтом і сервером.

Ключовими прикладами такої архітектури є торговельні платформи, такі як Bisq (для обміну криптовалют) (рисунок 2.1) та колишній OpenBazaar (для обміну товарами) (рисунок 2.2).

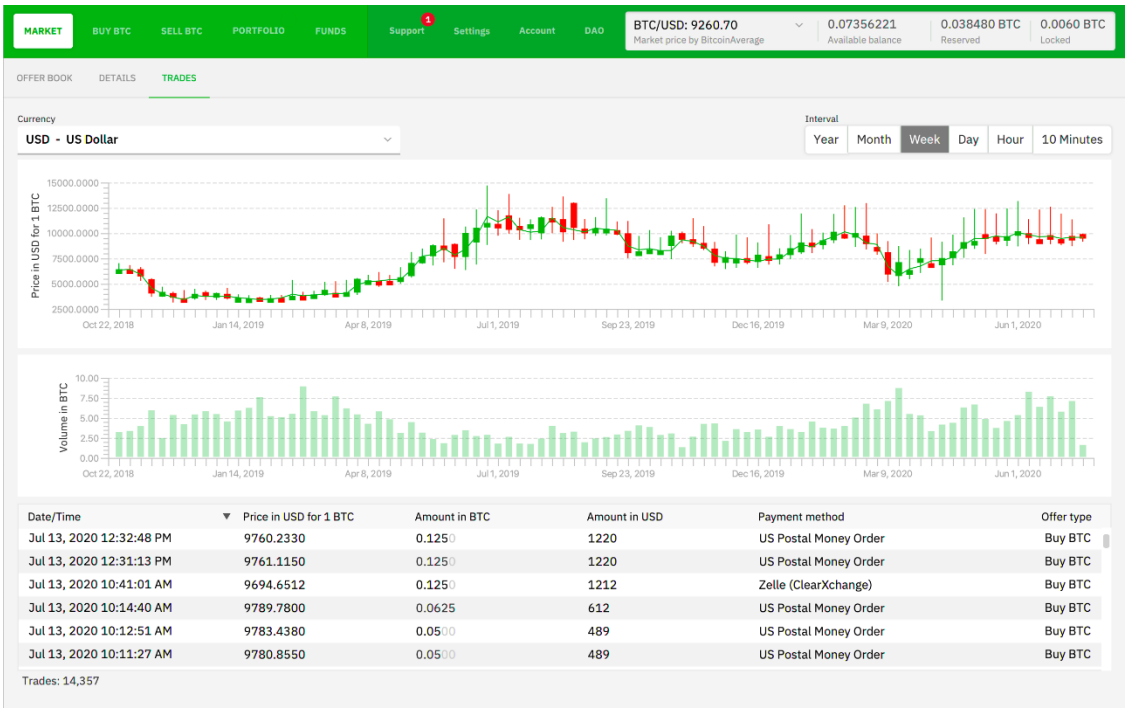

Рисунок 2.1 — Інтерфейс Bisq

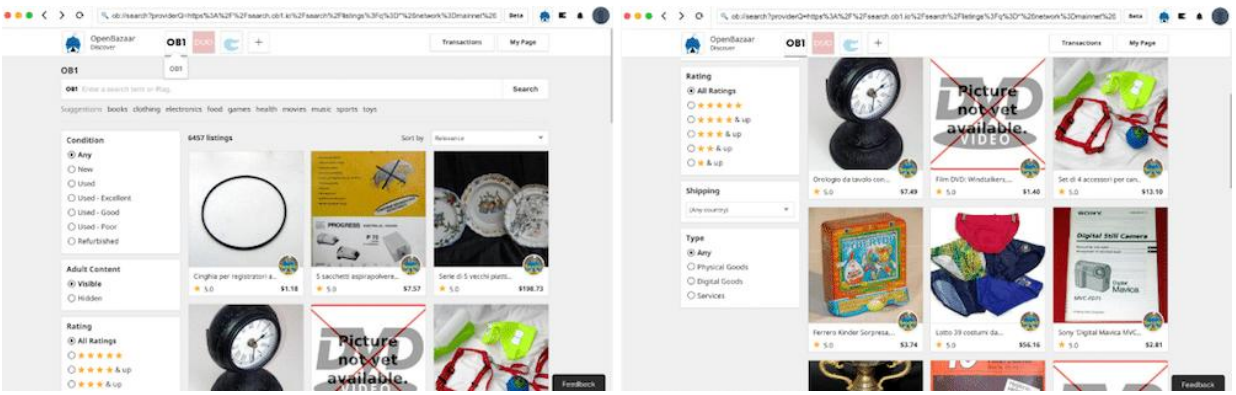

Рисунок 2.2 — Інтерфейс OpenBazaar

Аналіз архітектурних особливостей:

– повна відсутність SPOF (Single Point of Failure) — оскільки відсутній центральний сервер, система зберігає стійкість, навіть якщо значна частина вузлів тимчасово виходить з ладу, це відповідає нефункціональній вимозі до надійності;

– захист від цензури — інформація (оголошення, угоди) поширюється безпосередньо між вузлами через розподілену хеш-таблицю (DHT) або інші P2P-протоколи, жодна центральна структура не може заблокувати користувача чи видалити його оголошення;

– використання Multi-signature та Smart Contracts — у Bisq, наприклад, для забезпечення безпеки угод (Escrow) використовуються Multi-signature гаманці або прості Smart Contracts, це гарантує, що кошти блокуються до отримання підтвердження від обох сторін або арбітра.

Системні недоліки чистої P2P-архітектури:

Незважаючи на ідеальну децентралізацію, чиста P2P-архітектура має значні обмеження, які роблять її непрактичною для масового маркетплейсу, що є критичним висновком для проєктування:

– низька швидкість пошуку (Latency) — пошук товарів або послуг вимагає звернення до багатьох вузлів мережі, це значно збільшує час відгуку системи, що погіршує юзабіліті та продуктивність порівняно з ЦМП;

– низька масштабованість (Scalability) — зі зростанням кількості вузлів, навантаження на мережу (особливо на етапі ініціації транзакції) зростає непропорційно, що призводить до загального уповільнення;

– залежність від онлайну (Availability) — для того, щоб оголошення було доступне, вузол, який його розмістив, має бути онлайн, або ж дані мають зберігатися в додатковій розподіленій системі зберігання (наприклад, IPFS), що ускладнює архітектуру;

– високі Gas Costs — якщо для Escrow використовуються складніші Smart Contracts на публічних блокчейнах (наприклад, Ethereum), висока вартість транзакцій (Gas Cost) може зробити мікротранзакції (обмін дрібними товарами чи послугами) економічно невигідними.

Цей аналіз показує, що чиста P2P-архітектура, хоч і забезпечує ідеальний захист від цензури, не відповідає вимогам до продуктивності та юзабіліті сучасної інформаційної системи обміну, що обґрунтовує необхідність дослідження гібридних рішень у наступному підрозділі.

2.1.2. Дослідження гібридних блокчейн-рішень та їхніх переваг

Аналіз чистої P2P-архітектури показав, що, незважаючи на ідеальний рівень децентралізації, вона значно поступається централізованим рішенням за нефункціональними вимогами до продуктивності та юзабіліті (зокрема, через високу латентність пошуку та залежність від онлайну вузла). Це призводить до необхідності дослідження гібридних блокчейн-рішень, які прагнуть поєднати переваги обох моделей [9].

Гібридна архітектура ІС у контексті P2P-обміну визначається як модель, у якій критично важливі функції (що потребують довіри та незмінності) переносяться на блокчейн, тоді як функціонал, критичний до продуктивності та юзабіліті, залишається на централізованих або федеративних серверах.

Архітектурна декомпозиція гібридної моделі:

– децентралізований компонент (Блокчейн/Smart Contracts) відповідає за транзакційний Escrow — умовне депонування коштів до виконання умов, система довіри та репутації — незмінне зберігання та автоматичне оновлення рейтингів,

керування ідентифікацією (DID) — псевдонімні адреси та криптографічна верифікація;

– централізований/федеративний компонент (Web 2.0 Сервер) відповідає за, індексацію та пошук — високошвидкісний повнотекстовий пошук товарів, користувацький інтерфейс (UI/UX) — швидке завантаження та звична взаємодія, зберігання некритичних даних — метадані оголошень, зображення (часто з використанням CDN).

Переваги гібридної архітектури:

– оптимальне співвідношення між безпекою та продуктивністю — вирішується ключовий конфлікт чистих P2P-систем, висока швидкість пошуку (за рахунок централізованого індексу) поєднується з криптографічною безпекою транзакцій (за рахунок блокчейну);

– зниження Gas Cost та Latency — оскільки лише фінальні, критичні для довіри, транзакції (наприклад, Escrow, оплата) записуються в блокчейн, загальна кількість операцій у мережі зменшується, це дозволяє оптимізувати Gas Cost та підвищити швидкість підтвердження для основних дій користувачів;

– покращений UX/UI — користувач отримує звичний, швидкий та зручний інтерфейс, не вимагаючи постійного підключення до P2P-вузлів чи завантаження всього розподіленого реєстру.

Гібридний підхід є найбільш архітектурно обґрунтованим рішенням для розробки сучасної інформаційної системи обміну (спеціальність 126), оскільки він максимізує нефункціональні вимоги (швидкість, надійність) у частині взаємодії та безпеку у частині транзакцій.

2.2. Оцінка переваг та недоліків існуючих архітектурних рішень

2.2.1. Порівняння аналогів за критеріями Gas Cost, Latency та надійності

Для обґрунтування архітектурної моделі інформаційної системи необхідно провести кількісну та якісну оцінку існуючих рішень за ключовими нефункціональними вимогами: економічною ефективністю (Gas Cost), швидкістю

підтвердження транзакцій (Latency) та загальною надійністю. Порівняння архітектурних моделей зображене в таблиці 2.1.

Таблиця 2.1— Порівняльна таблиця аналогів

Критерій	Централізований Маркетплейс (ЦМП)	Чистий P2P (напр., Bisq)	Гібридний Блокчейн-Маркетплейс
Gas Cost (Вартість транзакції)	Низька (фіксована комісія посередника)	Висока (залежить від навантаження публічного блокчейну)	Низька/Оптимізована (критичні транзакції винесені на L2 або оптимізовані Smart Contracts)
Latency (Час підтвердження)	Низька (миттєве підтвердження на центральному сервері)	Висока (залежить від швидкості P2P-пошуку та часу генерації блоку)	Середня/Оптимальна (швидкість Web 2.0 для UI/UX, блокчейн – для фінальної угоди)
Надійність (Стійкість до відмови)	Низька (вразливість до SPOF)	Висока (відсутність SPOF, розподіл даних)	Висока (розподіл даних, стійкість до цензури)
Цензура	Високий ризик (повний контроль посередника)	Нульовий ризик	Низький ризик (усунення цензури у критичних Escrow-операціях)
Масштабованість	Висока (легке додавання серверів)	Низька (обмежена мережевими протоколами P2P)	Середня/Висока (досягається через off-chain обробку та шардінг)

Критичний аналіз результатів порівняння:

– gas cost против довіри — хоча ЦМП забезпечує низькі експлуатаційні витрати, він вимагає високої вартості довіри до посередника, чисті P2P-рішення, навпаки, надають високу довіру, але є економічно неефективними для мікротранзакцій через неоптимізовані Gas Costs, гібридні рішення вирішують цю проблему, вимагаючи плати за газ лише за найважливіші, довіроутворюючі операції;

– latency против юзабіліті — висока затримка в чистих P2P-системах робить їх непридатними для масового використання, оскільки користувачі очікують миттєвого відгуку, як у Web 2.0, гібридна архітектура стратегічно використовує централізовані компоненти для кешування та пошуку, мінімізуючи Latency у точках взаємодії з користувачем;

– надійність — чисті та гібридні системи мають архітектурну перевагу над ЦМП за критерієм надійності, оскільки вони усувають SPOF та ризик цензури у критичних операціях.

Ця оцінка однозначно свідчить про те, що гібридна архітектура є найбільш збалансованим і доцільним рішенням для розробки сучасної інформаційної системи обміну, оскільки вона ефективно управляє компромісами між продуктивністю, економічною ефективністю та криптографічною довірою. Це обґрунтовує необхідність подальшої розробки гібридної моделі.

2.2.2. Обґрунтування вибору гібридної моделі для подальшого проєктування

Аналіз архітектурних рішень виявив фундаментальний конфлікт між функціональними вимогами до зручності та продуктивності (характерними для ЦМП) та нефункціональними вимогами до безпеки та довіри (характерними для чистих P2P-систем). Гібридна модель є єдиним архітектурним рішенням, здатним забезпечити оптимальний компроміс між цими протиріччями, що робить її вибір науково обґрунтованим для проєктування інформаційної системи обміну.

Обґрунтування вибору ґрунтується на наступних архітектурних перевагах:

– гібридна архітектура дозволяє винести off-chain (на централізований сервер) усі операції, які не вимагають криптографічної довіри (пошук, фільтрація, реєстрація оголошень, завантаження зображень), це знімає обчислювальне навантаження з блокчейну, суттєво знижуючи Latency та мінімізуючи Gas Cost для користувача;

– операції, які вимагають довіри (Escrow, фіналізація угоди, оновлення рейтингу), залишаються on-chain (на блокчейні);

– гібридна модель архітектурно захищає критичні фінансові потоки та репутаційні дані, використовуючи незмінність блокчейну, ризик цензури та шахрайства посередника усувається у найважливіших точках: виконання угоди та збереження довіри;

– модель дозволяє використовувати горизонтальне масштабування для централізованої частини системи (Web-сервери, бази даних) для обслуговування мільйонів користувачів, не обмежуючись пропускною здатністю блокчейну, це робить систему готовою до індустріального впровадження;

– гібридний підхід дозволяє зберегти звичний та швидкий інтерфейс, як у ЦМП, використовуючи при цьому надійні, перевірені Web 2.0 технології, це є критичною функціональною вимогою для залучення широкої аудиторії.

Таким чином, розробка буде сфокусована на створенні моделі архітектурного розподілу функціоналу, де централізована частина забезпечує швидкість та доступність, а децентралізована — безпеку та криптографічну довіру.

2.3. Обґрунтування вибору методів та засобів моделювання архітектури ІС

Успішне проєктування складної гібридної інформаційної системи вимагає застосування формалізованих та стандартизованих методів моделювання. З огляду на специфіку архітектури, яка поєднує класичні Web 2.0 компоненти та децентралізовані блокчейн-сервіси, обрані методи повинні забезпечувати чітке відокремлення та моделювання взаємодії між цими різнорідними елементами.

Обґрунтуванням вибору Уніфікованої Мови Моделювання (UML) як основного засобу проєктування є її здатність охопити як функціональні, так і архітектурні (структурні) аспекти системи. У контексті гібридної ІС, UML дозволяє:

– моделювати функціональні вимоги — за допомогою діаграм варіантів використання (Use Case Diagrams) та діаграм діяльності (Activity Diagrams) можна чітко відобразити сценарії взаємодії користувачів як з централізованим інтерфейсом, так і з Smart Contract;

– моделювати архітектуру — діаграми компонентів (Component Diagrams) і діаграми розгортання (Deployment Diagrams) є критично важливими для візуалізації розподілу функціоналу, вони дозволять чітко розмежувати блокчейн-компонент (Smart Contract, вузли мережі) від традиційного бек-енду та фронт-енду;

– моделювати логіку Smart Contract — діаграми класів можуть бути використані для деталізації структури даних та методів виконання логіки, закладеної в Escrow Smart Contract.

Вибір інструментарію має забезпечувати підтримку UML та можливість інтеграції з документацією. Для цього будуть застосовані:

– CASE-засоби — (Наприклад, draw.io, Enterprise Architect, або подібні) для побудови UML-діаграм, що дозволяють формалізувати архітектуру та процеси відповідно до стандарту;

– технічна документація блокчейну — для моделювання взаємодії з блокчейном буде використана специфікація Web3.js/ethers.js як інтерфейсу взаємодії між централізованим бек-ендом та децентралізованим компонентом (Smart Contract).

2.4. Вибір програмно-технічної платформи та обґрунтування стеку технологій

Обґрунтування стеку технологій є критичним етапом проєктування гібридної інформаційної системи, оскільки він повинен задовольняти як вимоги до швидкості та юзабіліті (Web 2.0 частина), так і до безпеки та незмінності (блокчейн частина).

Для реалізації Escrow Smart Contract необхідно обрати блокчейн, який оптимізований за критеріями низького Gas Cost та високої швидкості підтвердження (Latency), зберігаючи при цьому сумісність з EVM (Ethereum Virtual Machine) для широкого інструментарію розробки [10]:

– Ethereum (Mainnet) — забезпечує найвищу надійність, але непридатний для P2P-маркетплейсу через надзвичайно високі Gas Costs та низьку пропускну здатність [11];

– L2 (Polygon, Arbitrum) — пропонують суттєве зниження Gas Cost та Latency, зберігаючи безпеку Ethereum [12];

– Binance Smart Chain (BNB Smart Chain) — є EVM-сумісним і забезпечує значно нижчі Gas Costs та вищу швидкість, ніж Ethereum Mainnet. Це робить його економічно вигідним вибором для мікротранзакцій та обміну послугами.

Для розробки буде обрано BNB Smart Chain (або сумісний тестовий Testnet) як середовище для розгортання Smart Contract, оскільки він забезпечує оптимальний баланс між EVM-сумісністю, низькими транзакційними витратами та швидкістю. Мовою розробки Smart Contract буде Solidity [13].

Централізований компонент відповідає за UI/UX, швидкий пошук та зберігання некритичних метаданих. Вибір технологій з обґрунтуванням зображений в таблиці 2.2.

Таблиця 2.2 — Вибір стеку під розробку

№	Компонент	Технологія	Обґрунтування
1	Фронт-енд (UI/UX)	React/Next.js	Забезпечує високу швидкість відгуку, модульність, сучасний UI/UX та ефективне управління станом для взаємодії з Web3.js.
2	Бек-енд (API, Індексація)	Node.js (Express)	Вибір однієї з цих технологій забезпечує високу продуктивність, масштабованість, швидке розгортання та ефективну обробку асинхронних запитів, необхідних для індексації даних з блокчейну.
3	База даних	MongoDB (NoSQL)	Обрано для забезпечення горизонтальної масштабованості та гнучкості схеми даних, що критично важливо для роботи з динамічними неструктурованими даними (різні категорії товарів, опис послуг) та ефективної індексації в умовах високого навантаження.
4	Взаємодія з Блокчейном	Web3.js / Ethers.js	Стандартні бібліотеки для JavaScript, які забезпечують надійний та ефективний інтерфейс для виклику функцій Smart Contract та читання даних із розподіленого реєстру.

Обраний стек технологій відповідає вимогам до гібридної архітектури, забезпечуючи високу швидкість Web 2.0 частини та криптографічну безпеку Web 3.0 частини.

Висновки до розділу 2

У розділі 2 проведено огляд існуючих рішень та обґрунтовано вибір архітектури та технологій для інформаційної системи обміну товарами та послугами.

Проведено критичний аналіз архітектур децентралізованих P2P-маркетплейсів. Проаналізовано чисті P2P-рішення (Bisq, OpenBazaar): переваги (відсутність SPOF, захист від цензури) та недоліки (низька швидкість пошуку, низька масштабованість, залежність від онлайну, високі Gas Costs). Визначено, що чиста P2P-архітектура не відповідає вимогам до продуктивності та юзабіліті сучасної інформаційної системи.

Досліджено гібридні блокчейн-рішення, що поєднують переваги централізованих та децентралізованих систем. Визначено архітектурну декомпозицію гібридної моделі: децентралізований компонент (блокчейн/Smart Contracts) для транзакційного Escrow, системи довіри/репутації та керування ідентифікацією; централізований компонент (Web 2.0 сервер) для індексації та пошуку, користувацького інтерфейсу та зберігання некритичних даних. Визначено переваги гібридної архітектури: оптимальне співвідношення безпеки та продуктивності, зниження Gas Cost та Latency, покращений UX/UI.

Проведено порівняння аналогів за критеріями Gas Cost, Latency та надійності. Порівняно централізовані маркетплейси, чисті P2P-системи та гібридні блокчейн-маркетплейси. Визначено, що гібридна архітектура є найбільш збалансованим рішенням, оскільки ефективно управляє компромісами між продуктивністю, економічною ефективністю та криптографічною довірою.

Обґрунтовано вибір гібридної моделі для подальшого проєктування на основі архітектурних переваг: оптимізація транзакційних витрат та продуктивності, управління ризиками довіри та цензури, ефективна масштабованість, збереження звичного UX/UI.

Обґрунтовано вибір методів та засобів моделювання архітектури ІС. Обрано UML як основний засіб проєктування для моделювання функціональних вимог,

архітектури та логіки Smart Contract. Визначено інструментарій: CASE-засоби (draw.io) для побудови UML-діаграм та технічна документація блокчейну (Web3.js/ethers.js) для моделювання взаємодії з блокчейном.

Обґрунтовано вибір програмно-технічної платформи та стеку технологій. Для децентралізованого компонента обрано BNB Smart Chain (EVM-сумісний, низькі транзакційні витрати, висока швидкість) та мову Solidity для розробки Smart Contract. Для централізованого компонента обрано стек: React/Next.js для фронт-енду, Node.js (Express) для бек-енду, MongoDB (NoSQL) для бази даних, Web3.js/Ethers.js для взаємодії з блокчейном.

Гібридна архітектура є найбільш обґрунтованим рішенням для розробки сучасної інформаційної системи обміну, оскільки поєднує переваги централізованих систем (швидкість, зручність) та децентралізованих систем (безпека, прозорість, відсутність цензури). Обраний стек технологій відповідає вимогам до гібридної архітектури, забезпечуючи високу швидкість Web 2.0 частини та криптографічну безпеку Web 3.0 частини. Це створює основу для подальшого функціонального проєктування та моделювання системи в розділі 3.

3 ФУНКЦІОНАЛЬНІ ЕТАПИ ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ

3.1. Функціональні вимоги, деталізація функціоналу ІС та декомпозиція модулів

3.1.1. Декомпозиція функціональних вимог системи

Етап проєктування гібридної інформаційної системи вимагає ретельної декомпозиції функціональних вимог, що є необхідним для коректного архітектурного розподілу між централізованим сервером (off-chain) та блокчейном (on-chain). Як було обґрунтовано в Розділі 2, ключовою перевагою гібридної моделі є перенесення завдань, критичних до швидкості та юзабіліті, на централізований компонент, залишаючи на блокчейні лише функції, критичні для забезпечення довіри та незмінності.

Функціональні вимоги до системи обміну логічно поділяються на чотири основні групи, кожна з яких має своє архітектурне призначення:

– функціонал управління користувачами та ідентифікацією — цей блок охоплює процеси реєстрації, аутентифікації та управління профілем, хоча реєстрація та відображення профілю користувача обробляється централізованим бек-ендом для забезпечення швидкості, криптографічна ідентифікація, що є основою для участі у Trustless-угодах, тісно пов'язана з гаманцем користувача, отже, до цього функціоналу належить необхідність зв'язування централізованого профілю з унікальною On-chain адресою;

– функціонал управління оголошеннями та пошуком — цей набір вимог безпосередньо спрямований на забезпечення переваг гібридної системи над чистою P2P-архітектурою, створення, редагування та видалення оголошень є off-chain операціями; метадані зберігаються у високопродуктивній базі даних MongoDB, що забезпечує гнучкість та швидкість індексації, ключовим функціоналом тут є високошвидкісний повнотекстовий пошук, фільтрація та категоризація товарів чи послуг, що має відбуватися з мінімальною латентністю [14].

– функціонал укладання та виконання угод (Escrow) — цей блок є найбільш критичним і повністю залежить від On-chain компонента – Escrow Smart Contract, вимоги до нього включають ініціалізацію угоди, умовне депонування коштів покупця на контракті та механізми підтвердження виконання зобов'язань, логіка виконання має бути автоматизована: кошти звільняються продавцю лише після криптографічного підтвердження покупцем отримання товару, або повертаються покупцю відповідно до арбітражного рішення, це гарантує цілісність та незмінність фінансових зобов'язань.

– функціонал управління репутацією та арбітражем — для забезпечення стійкої системи довіри, функціонал оновлення рейтингу та залишення відгуків повинен бути архітектурно захищений від маніпуляцій, вимога полягає в тому, що відгук може бути залишений і записаний у блокчейн лише після успішного, підтвердженого Smart Contract завершення угоди, це мінімізує ризик атаки Sybil та забезпечує надійну репутаційну історію, у випадку спору, має бути передбачено виклик функції арбітражу, також реалізований через Escrow Smart Contract.

Ця декомпозиція дозволяє визначити чіткі межі відповідальності для кожного компонента системи, що є основою для подальшої специфікації модулів.

3.1.2. Специфікація ключових модулів ІС (фронт-енд, бек-енд, блокчейн-інтерфейс)

Успішна реалізація гібридної архітектури, вимагає чіткої специфікації трьох ключових логічних модулів системи. Ця специфікація визначає їхні обов'язки та методи взаємодії, що є критичним для забезпечення цілісності та розподілу функціоналу (Додаток Б).

Модуль користувацького інтерфейсу (Фронт-енд) — реалізований за допомогою React/Next.js, відповідає за презентаційний рівень системи та взаємодію з користувачем. Його основна роль полягає у забезпеченні високого рівня юзабіліті та швидкості відгуку, що є ключовою перевагою гібридної моделі (Додаток В).

Основні обов'язки: обробка даних, отриманих від бек-енду (пошук, оголошення); відображення динамічного інтерфейсу; управління станом автентифікації.

Ключова функція: Реалізація Web3-інтерфейсу, який забезпечує пряму взаємодію з гаманцем користувача (наприклад, MetaMask) та виклик функцій Escrow Smart Contract (наприклад, deposit, confirmDelivery). Це досягається через бібліотеки Web3.js або Ethers.js.

Модуль централізованого сервера (Бек-енд) - виконує функції централізованого посередника для некритичних даних та є "містком" між фронт-ендом і блокчейном. Використовуючи Node.js та MongoDB, цей модуль оптимізований для високої продуктивності та гнучкості.

Основні обов'язки: управління API для фронт-енду; автентифікація користувачів (логін, сесії); зберігання та індексація метаданих оголошень у MongoDB (що критично для швидкого пошуку).

Функція індексатора блокчейну (Blockchain Indexer) — це спеціалізований підмодуль, який постійно моніторить події (Events) на Escrow Smart Contract (наприклад, AgreementStarted, FundsReleased). Отримані дані про стан угод (які є On-chain) зберігаються у MongoDB для швидкого доступу та відображення у профілях користувачів. Це усуває необхідність постійно запитувати блокчейн, знижуючи Latency.

Модуль блокчейну (Smart Contracts та Інтерфейс) є ядром Trustless-архітектури, що забезпечує виконання фінансових та репутаційних зобов'язань.

Основні обов'язки: депонування, блокування та звільнення коштів у рамках Escrow-угоди; фіксація результатів арбітражу; незмінне оновлення репутаційного показника користувача.

Ключовий елемент: Escrow Smart Contract, реалізований на Solidity, єдиний програмний об'єкт, який має право контролювати кошти угоди. Логіка цього контракту визначає, як саме вирішуються суперечки та коли відбувається фінальний розрахунок.

Взаємодія між цими модулями здійснюється через API-запити Web3-виклики, забезпечуючи розподіл навантаження та функціоналу відповідно до принципів гібридної архітектури.

3.2. Розробка та обґрунтування гібридної архітектурної моделі ІС

3.2.1. Структурна модель розподілу функціоналу

Гібридна архітектурна модель інформаційної системи обміну є науково обґрунтованим рішенням, що дозволяє подолати дилему між вимогами до швидкості та довіри, яка існує в чистих централізованих та децентралізованих системах. Ця модель структурно складається з трьох ключових рівнів, які забезпечують ефективний розподіл функціональних обов'язків:

– централізований рівень (Off-chain Компоненти);

– децентралізований рівень (On-chain Компоненти);

– рівень інтеграції (Web3-Інтерфейс).

Централізований Рівень забезпечує швидкість, юзабіліті та обробку високого навантаження, виконуючи функції, які не потребують криптографічної довіри. Він складається з фронт-енду (React/Next.js) та бек-енду (Node.js з MongoDB). Основним завданням цього рівня є:

– презентація та UI/UX — надання користувачу звичного, швидкого та інтуїтивно зрозумілого інтерфейсу;

– індексація та пошук — використовуючи гнучкість MongoDB, центральний сервер індексує метадані оголошень та історію транзакцій (отриману від блокчейну), забезпечуючи миттєвий пошук, який неможливий у чистих P2P-системах;

– аутентифікація та сесійне управління — управління профілями користувачів та сесіями, не пов'язаними з фінансовими активами.

Децентралізований Рівень є основою Trustless-системи, відповідаючи за забезпечення незмінності, безпеки та виконання угод. Його ядром є Escrow Smart

Contract, розгорнутий на EVM-сумісному блокчейні (BNB Smart Chain Testnet). Основні функції цього рівня:

– криптографічне депонування (Escrow) — блокування коштів угоди до моменту виконання заданих умов;

– автоматичне виконання угоди — звільнення або повернення активів згідно з логікою контракту;

– незмінне зберігання даних довіри — фіксація результату угоди, що є основою для репутаційної системи.

Рівень інтеграції виступає "мостом" і є критично важливим для функціонування гібридної архітектури. Він забезпечує двосторонній потік інформації:

– запис — фронт-енд викликає функції Smart Contract (наприклад, startAgreement) через бібліотеки Web3.js/Ethers.js, ініціюючи транзакції;

– читання — бек-енд, через функцію Blockchain Indexer, постійно моніторить On-chain події, щоб синхронізувати стан угоди та оновити відповідні записи у MongoDB.

Таким чином, розроблена гібридна модель архітектурно захищає фінансові потоки та репутацію, використовуючи блокчейн, а всю ресурсомістку та схильну до частої зміни інформацію (UI, пошук) обробляє централізовано. Це рішення максимізує нефункціональні вимоги до надійності, швидкості та економічної ефективності.

3.2.2. Проєктування інтерфейсів взаємодії між компонентами (API, Web3)

Успіх гібридної інформаційної системи критично залежить від чітко спроєктованих інтерфейсів взаємодії, які забезпечують надійний та ефективний обмін даними між трьома основними логічними модулями: фронт-ендом, бек-ендом та блокчейном. Взаємодія поділяється на два основні типи: традиційні REST API та децентралізовані Web3-інтерфейси (Додаток Г).

REST API використовується для комунікації між фронт-ендом (React/Next.js) і бек-ендом (Node.js/Express) для всіх off-chain операцій. Цей інтерфейс має забезпечувати мінімальну латентність (Latency) і високу пропускну здатність, що є ключовою нефункціональною вимогою.

Функціональне призначення — здійснення операцій, пов'язаних з управлінням профілем, обробкою пошукових запитів та обміном повідомленнями.

Ключові API-ендпоінти:

– /api/v1/search — прийом запиту та повернення результатів пошуку з MongoDB. Це основний ендпоінт, який демонструє перевагу гібридної моделі у швидкості над чистою P2P;

– /api/v1/listings — створення, читання, оновлення та видалення метаданих оголошень (за винятком критичних даних угоди);

– /api/v1/auth — аутентифікація користувачів та управління сесіями.Web3-інтерфейс є прямою комунікацією між фронт-ендом користувача та Escrow Smart Contract, розгорнутим на блокчейні.

Замість центрального сервера, інтерфейс використовує бібліотеки, такі як Web3.js або Ethers.js, для виклику публічних функцій контракту.

Функціональне призначення — ініціація транзакцій, які змінюють стан блокчейну (Escrow, оплата, підтвердження).

Ключові Web3-функції (Smart Contract Methods):

– startAgreement(buyerAddress, value, metadataHash) — ініціація угоди з депонуванням коштів;

– confirmDelivery() — викликається покупцем для звільнення коштів;

– raiseDispute(arbitratorAddress) — викликається стороною у випадку спору;

– getAgreementState(agreementId) — читання поточного On-chain статусу угоди.

Критичним елементом проєктування є забезпечення синхронізації між централізованим та децентралізованим світами. Цей зв'язок реалізується через Індексатор Блокчейну (Blockchain Indexer), що є підмодулем бек-енду:

– індексатор постійно відстежує події (Events), що випромінюються Escrow Smart Contract (наприклад, event FundsDeposited(address buyer, uint256 agreementId));

– при отриманні події, Індексатор оновлює відповідний запис у MongoDB через внутрішній API.

Таке архітектурне рішення гарантує, що фронт-енд завжди відображає актуальний стан угоди (прочитаний швидко з MongoDB), але сам критичний стан (власність коштів) завжди контролюється незмінним Smart Contract. Це є еталонним підходом у проєктуванні гібридних DApps.

3.3. Моделювання функціональних процесів системи

Етап архітектурного проєктування, деталізований у попередніх підрозділах, тепер вимагає переходу до формалізованого моделювання функціональних процесів. Метою цього етапу є створення графічного, уніфікованого представлення того, як користувачі взаємодіють із гібридною інформаційною системою. Використання Уніфікованої Мови Моделювання (UML) дозволяє чітко відобразити як загальні сценарії використання, так і внутрішню бізнес-логіку, що особливо важливо в умовах розподілу функціоналу між централізованим бек-ендом і децентралізованим блокчейн-модулем. Моделювання виступає мостом між вимогами та безпосередньою реалізацією. На цьому етапі візуалізовано, хто, що і як робить у системі, зосереджуючись на ключовому процесі — безпечному укладанні угоди через Escrow Smart Contract.

3.3.1. Моделювання прецедентів використання (Use Case Diagrams)

Моделювання прецедентів використання є першим кроком у візуалізації функціональних вимог, оскільки воно визначає межі системи, її основних акторів та ключові функції, які актори можуть виконувати. Головними акторами у

розроблюваній гібридній P2P-системі виступають Продавець, Покупець, Арбітр та, що архітектурно важливо, Блокчейн-Мережа (Додаток Д).

Для розроблюваної системи загальний простір прецедентів використання поділяється на два домени, що відображає гібридну архітектуру. До Off-chain домену належать прецеденти, пов'язані з швидкою взаємодією: пошук оголошень, перегляд профілю та управління особистими даними. Наприклад, актор "Покупець" виконує прецедент "Швидкий пошук оголошень", який обробляється централізованим сервером. До On-chain домену належать прецеденти, які безпосередньо взаємодіють зі Smart Contract і вимагають криптографічної довіри.

Ключовим прецедентом у цій системі є "Укладання та виконання угоди Escrow". Цей прецедент об'єднує акторів "Продавець" та "Покупець", а його виконання вимагає взаємодії з Блокчейн-Мережею. Сценарій розгортається від ініціації (виклик функції startAgreement з боку Покупця) до фіналізації (виклик confirmDelivery або raiseDispute). На діаграмі прецедентів використання ця взаємодія чітко відображає, що система вимагає не лише клієнт-серверної комунікації, але й взаємодії з децентралізованою логікою. Прецедент "Вирішення спору" виокремлюється для актора "Арбітр", показуючи, що його роль є обмеженою та викликається лише при наявності конфлікту, а його рішення автоматично виконується Smart Contract.

Таким чином, діаграма прецедентів використання стає не просто описом функцій, а візуальною моделлю архітектурної гібридності, що є прямим доказом виконання вимог до наукової новизни.

3.3.2. Моделювання бізнес-логіки (Activity Diagrams та Sequence Diagrams)

Моделювання бізнес-логіки є критично важливим етапом, оскільки дозволяє формалізувати послідовність дій та умов, необхідних для виконання ключових прецедентів використання, зокрема, процесу безпечного обміну через Escrow Smart Contract. Для досягнення цієї мети використовуються два основні інструменти UML: діаграми діяльності та діаграми послідовності.

Діаграми діяльності (Activity Diagrams) використовуються для візуалізації потоку управління та станів виконання. У контексті гібридної P2P-системи, діаграма діяльності ефективно ілюструє повний життєвий цикл угоди Escrow. Вона чітко розділяється на доріжки відповідальності (swimlanes), які відображають ключових акторів і системні модулі: Покупець, Продавець, Escrow Smart Contract та Центральний Сервер. Діаграма діяльності деталізує процес від ініціації угоди Покупцем (депонування коштів на Smart Contract), через вузол рішення (decision node), який перевіряє факт підтвердження отримання товару, до фінального стану – автоматичного звільнення коштів Продавцю або переходу до процесу арбітражу. Таке моделювання підтверджує, що критична логіка (блокування/розблокування активів) повністю виконується децентралізованим модулем, мінімізуючи втручання Центрального Сервера.

Діаграми послідовності (Sequence Diagrams), навпаки, застосовуються для моделювання потоку повідомлень у часі, що є незамінним для проєктування гібридних інтерфейсів. Ці діаграми відображають, як саме Front-end викликає функції Smart Contract, і як Центральний Бек-енд реагує на On-chain події. Наприклад, діаграма послідовності для процесу "Старт Угоди" повинна демонструвати, що Front-end спершу взаємодіє з Центральним Бек-ендом для отримання метаданих оголошення, а потім ініціює прямий Web3-виклик до Escrow Smart Contract (повідомлення startAgreement), минаючи Центральний Сервер у момент фінансової транзакції. Крім того, інша діаграма послідовності може моделювати, як Індексатор Блокчейну (підмодуль Бек-енду) асинхронно отримує подію (Event) від Smart Contract після успішного депонування і оновлює MongoDB, забезпечуючи швидке відображення статусу угоди для користувача без високої Latency.

Завдяки цьому подвійному моделюванню (дії та послідовності повідомлень) забезпечується чітке розуміння динамічної поведінки ІС, що є фундаментальною основою для подальшого кодування та перевірки функціональних вимог системи.

3.4. Проєктування логічної структури системи

Проєктування логічної структури системи є етапом, на якому абстрактні функціональні процеси, описані в підрозділі 3.3, перетворюються на конкретні програмні компоненти та класи. Цей етап критично важливий для гібридної архітектури, оскільки він має чітко визначити межі відповідальності між On-chain (Smart Contract) та Off-chain (серверним) кодом. Для цього використовуються діаграми компонентів та діаграми класів UML.

3.4.1. Діаграми компонентів, що включають блокчейн-модуль

Діаграма компонентів відображає фізичні та логічні модулі системи та їхні взаємозалежності, надаючи високорівневе архітектурне бачення. У гібридній моделі чітко виділяються три основні компоненти:

– компонент клієнтського інтерфейсу (Front-end) — відповідає за відображення даних, користувацьку взаємодію та ініціацію транзакцій, його ключова залежність — це Web3-Інтерфейс, що забезпечує зв'язок із зовнішнім світом блокчейну;

– компонент централізованого сервера (Back-end) — містить модулі автентифікації, управління оголошеннями та, найважливіше, Індексатор Блокчейну, основні залежності цього компонента — це база даних MongoDB (для швидкого пошуку метаданих) та прямий доступ до Блокчейн-Мережі (для моніторингу подій);

– компонент блокчейну (Escrow Smart Contract) — є децентралізованою, незмінною частиною системи, його залежність — це сама Блокчейн-Мережа (BNB Smart Chain Testnet), яка надає обчислювальне середовище (EVM), цей компонент надає свої функції іншим компонентам через публічний ABI-інтерфейс.

На діаграмі компонентів взаємодія зображується як чітке розділення: Front-end та Back-end взаємодіють через REST API, але фінансові транзакції та оновлення стану угоди відбуваються через Escrow Smart Contract. Бек-енд у цій моделі не має

права виконувати критичні транзакції, а лише моніторити їхній статус, що є архітектурним підтвердженням Trustless-принципу.

3.4.2. Діаграми класів для реалізації Smart Contract та бек-енду

Діаграми класів деталізують внутрішню структуру компонентів, визначаючи їхні атрибути, методи та зв'язки. Це є безпосередньою основою для кодування як серверної логіки, так і Solidity-коду Smart Contract (Додаток Е).

Проєктування класів Smart Contract (On-chain) — ключовим класом у децентралізованій частині є клас EscrowContract. Його атрибути включають стан угоди (наприклад, state: enum {Created, Deposited, Delivered, Disputed, Completed}), адреси учасників (buyerAddress, sellerAddress, arbitratorAddress) та суму депозиту. Методи контракту (deposit(), confirmDelivery(), resolveDispute()) реалізують логіку життєвого циклу угоди.

Проєктування класів Бек-енду (Off-chain) — серверна частина містить класи, які моделюють дані та логіку управління системою (Додаток Ж). Основні класи включають:

– user (користувач) — містить Off-chain дані (ім'я, e-mail) та посилання на On-chain Wallet Address;

– listing (оголошення) — містить метадані оголошення (назва, опис, ціна, категорія), що зберігаються у MongoDB;

– agreement (угода) — зберігає On-chain ID угоди та її останній відомий статус (синхронізований Індексатором) для швидкого відображення.

Взаємозв'язки між цими класами показують, як саме централізовані дані посилаються на децентралізовані записи. Наприклад, клас Listing має асоціацію з класом Agreement через унікальний ідентифікатор транзакції, який є незмінним ID, зафіксованим у блокчейні. Це завершує проєктування логічної структури, підтверджуючи готовність до етапу реалізації.

Висновки до розділу 3

У розділі 3 виконано функціональні етапи проєктування та моделювання гібридної інформаційної системи обміну товарами та послугами.

Проведено декомпозицію функціональних вимог системи на чотири групи: управління користувачами та ідентифікацією (реєстрація, аутентифікація, зв'язування профілю з блокчейн-адресою), управління оголошеннями та пошуком (створення, редагування, видалення оголошень, швидкий пошук через MongoDB), укладання та виконання угод через Escrow Smart Contract (ініціалізація, депонування, автоматичне виконання), управління репутацією та арбітражем (оновлення рейтингу через блокчейн, вирішення спорів). Ця декомпозиція визначає межі відповідальності для кожного компонента системи.

Специфіковано три ключові модулі: модуль користувацького інтерфейсу (React/Next.js) для презентаційного рівня та Web3-інтерфейсу, модуль централізованого сервера (Node.js/MongoDB) для API, аутентифікації, індексації та Blockchain Indexer для синхронізації on-chain подій, модуль блокчейну (Escrow Smart Contract на Solidity) для виконання фінансових та репутаційних зобов'язань. Взаємодія між модулями здійснюється через REST API та Web3-виклики.

Розроблено та обґрунтовано гібридну архітектурну модель з трьома рівнями: централізований рівень (off-chain) для швидкості та юзабіліті (фронт-енд, бек-енд, індексація, пошук, аутентифікація), децентралізований рівень (on-chain) для незмінності та безпеки (Escrow Smart Contract для депонування, автоматичного виконання угод, незмінного зберігання даних довіри), рівень інтеграції (Web3-інтерфейс) як "місток" між компонентами для запису та читання даних з блокчейну. Модель захищає фінансові потоки та репутацію через блокчейн, а ресурсомістку інформацію обробляє централізовано.

Спроєктовано інтерфейси взаємодії між компонентами: централізований інтерфейс (REST API) для комунікації між фронт-ендом та бек-ендом (ендпоінти для пошуку, управління оголошеннями, аутентифікації), децентралізований інтерфейс (Web3-інтерфейс) для прямої комунікації між фронт-ендом та

Escrow Smart Contract (функції startAgreement, confirmDelivery, raiseDispute, getAgreementState), зв'язок інтерфейсів через Blockchain Indexer для синхронізації on-chain подій з MongoDB, забезпечуючи швидке відображення стану угод без високої Latency.

Проведено моделювання функціональних процесів системи: моделювання прецедентів використання (Use Case Diagrams) для визначення меж системи, основних акторів (продавець, покупець, арбітр, блокчейн-мережа) та ключових функцій, розділення на off-chain та on-chain домени, моделювання бізнес-логіки (Activity Diagrams та Sequence Diagrams) для візуалізації потоку управління, станів виконання та послідовності повідомлень у часі, що забезпечує розуміння динамічної поведінки системи.

Спроєктовано логічну структуру системи: діаграми компонентів для відображення фізичних та логічних модулів системи (компонент клієнтського інтерфейсу, компонент централізованого сервера, компонент блокчейну) та їхніх взаємозалежностей, діаграми класів для деталізації внутрішньої структури компонентів (класи Smart Contract: EscrowContract з атрибутами стану угоди та методами життєвого циклу, класи бек-енду: User, Listing, Agreement з посиланнями на on-chain дані), що завершує проєктування логічної структури та підтверджує готовність до етапу реалізації.

4 НЕФУНКЦІОНАЛЬНІ ЕТАПИ РЕАЛІЗАЦІЇ ІС ТА ЕКСПЕРИМЕНТАЛЬНОЇ ОЦІНКИ

4.1. Нефункціональні вимоги: Визначення вимог до безпеки, надійності та масштабованості

Ефективність гібридної інформаційної системи (ІС) вимірюється не лише її функціональними можливостями, але й відповідністю суворим нефункціональним вимогам, які в контексті інтеграції блокчейну набувають критичного значення. Ці вимоги визначають, наскільки добре система працює, її стійкість до збоїв та атак, а також її здатність обслуговувати зростаючу кількість користувачів. Чітке визначення нефункціональних вимог є основою для подальшої експериментальної оцінки (п. 4.3).

4.1.1. Визначення вимог до безпеки (криптографічний захист, цілісність даних)

Безпека є пріоритетною нефункціональною вимогою, особливо коли мова йде про фінансові транзакції та управління довірою. У гібридній ІС вимоги до безпеки поділяються відповідно до архітектурних компонентів:

Вимоги до On-chain безпеки (блокчейн-модуль):

– криптографічна цілісність Escrow — основна вимога полягає у забезпеченні незмінності логіки Escrow Smart Contract після його розгортання, будь-яка зміна стану угоди має бути можлива лише через криптографічно підтверджену транзакцію від авторизованого учасника (покупця, продавця або арбітра);

– захист від Reentrancy та переповнення — код Smart Contract, реалізований на Solidity, має бути захищений від типових вразливостей, які можуть призвести до неправомірного виведення коштів (наприклад, атаки Reentrancy та Integer Overflow/Underflow), це вимагає використання сучасних шаблонів безпеки та інструментів статичного аналізу коду;

– незмінність даних довіри — результат угоди та репутаційні оцінки повинні зберігатися в блокчейні таким чином, щоб унеможливити їхнє видалення чи модифікацію, забезпечуючи архітектурний захист від Sybil-атак.

Вимоги до Off-chain Безпеки (Централізований Сервер):

– конфіденційність персональних даних — необхідність мінімізації зберігання чутливих даних користувачів. Персональні дані повинні зберігатися у MongoDB із застосуванням сучасних алгоритмів хешування та шифрування. Критична вимога — відсутність передачі приватних ключів користувачів на центральний сервер;

– безпека API — всі REST API-ендпоінти мають бути захищені за протоколом HTTPS та вимагати надійної автентифікації (наприклад, JWT-токенами), щоб запобігти несанкціонованому доступу та ін'єкціям.

4.1.2. Визначення вимог до продуктивності та надійності (Latency, Uptime)

Продуктивність та надійність визначають здатність системи надавати сервіс швидко та безперервно, що є прямою відповіддю на недоліки чистих P2P-систем.

Вимоги до продуктивності (Performance):

– швидкість пошуку (Off-chain Latency) — час відповіді централізованого пошукового API (з MongoDB) не повинен перевищувати 150 мс для 95% запитів, ця вимога гарантує високий рівень юзабіліті, характерний для Web 2.0;

– час підтвердження транзакції (On-chain Latency) — час підтвердження фінальної транзакції Smart Contract (наприклад, confirmDelivery) у блокчейн-мережі BNB Smart Chain Testnet не повинен перевищувати 10 секунд, цей показник прямо залежить від обраного механізму консенсусу (PoS) і є ключовим предметом експериментального вимірювання;

– економічна Ефективність (Gas Cost) — вимога до оптимізованого Escrow Smart Contract полягає у тому, що його Gas Cost має бути мінімальним та не перевищувати Gas Units для базової транзакції, це є прямою вимогою до функціональної ефективності коду.

Вимоги до Надійності та Масштабованості:

– надійність (Uptime) — час безвідмовної роботи централізованого компонента має бути не менше 99.9% (три дев'ятки), що досягається за допомогою балансувальників навантаження та резервного копіювання;

– горизонтальна масштабованість — архітектура має підтримувати масштабування централізованого компонента шляхом додавання нових вузлів бек-енду та горизонтального шардингу MongoDB, забезпечуючи можливість обслуговування до 1000 активних користувачів на хвилину [15];

– цілісність даних (Consistency) — хоча блокчейн забезпечує незмінність, централізований Індексатор повинен забезпечувати Eventually Consistency (з часом узгодженість) між On-chain статусом угоди та даними у MongoDB, з максимальним часом затримки синхронізації не більше 5 секунд.

4.2. Реалізація ключового елемента: Escrow Smart Contract

4.2.1. Вибір платформи (Testnet) та мови програмування (Solidity)

Реалізація компонента, критичного для забезпечення довіри та виконання угод, вимагає чіткого обґрунтування вибору технологічного середовища, що відповідає нефункціональним вимогам до економічної ефективності та швидкості.

Обґрунтування вибору платформи — як було визначено в Розділі 2, для розробки обирається EVM-сумісний блокчейн, який забезпечує низькі транзакційні витрати (Gas Cost) та високу швидкість підтвердження (Latency). Для цілей проєктування та експериментальної оцінки, Escrow Smart Contract буде розгорнутий на BNB Smart Chain Testnet. Цей вибір є оптимальним, оскільки він:

– імітує реальні умови — BNB Smart Chain використовує механізм консенсусу, близький до Proof-of-Stake (PoS), що забезпечує низьку Latency (зазвичай до 3-5 секунд), що є критично важливим для P2P-обміну;

– забезпечує економічну доцільність — тестове середовище дозволяє точно виміряти Gas Cost оптимізованого контракту перед його розгортанням у реальній мережі, підтверджуючи виконання вимоги до економічної ефективності.

Обґрунтування вибору мови програмування — мовою розробки Smart Contract обирається Solidity. Ця мова є стандартом де-факто для розробки контрактів на платформах, сумісних з EVM. Solidity дозволяє розробнику чітко визначити стан (State) контракту, а також функції, які керують життєвим циклом угоди та перерозподілом активів, що є необхідним для реалізації складної логіки Escrow.

Використання стандартизованого середовища та мови забезпечує можливість застосування перевірених інструментів для аудиту безпеки коду, що є необхідним для виконання вимог до криптографічної цілісності.

4.2.2. Розробка та тестування функціоналу контракту

Розробка Escrow Smart Contract є ключовим етапом реалізації, оскільки він архітектурно замінює централізований орган арбітражу. Реалізований контракт має відповідати принципам Багатостороннього Escrow, але з акцентом на оптимізації Gas Cost.

Функціонал та Логіка Контракту: Smart Contract реалізує життєвий цикл угоди через набір публічних методів та внутрішній перелік станів:

– ініціалізація (startAgreement) — покупець викликає цю функцію, депонуючи кошти, контракт фіксує суму, адреси продавця, покупця та обраного арбітра, і переводить угоду у стан deposited;

– виконання (confirmDelivery) — викликається покупцем після отримання товару, якщо підтвердження успішне, контракт автоматично переводить кошти продавцю, фіксує Event про успішне завершення та переходить у стан completed;

– вирішення спору (raiseDispute, resolveDispute) — у випадку конфлікту, будь-яка зі сторін може викликати raiseDispute, переводячи угоду у стан disputed, арбітр (обраний учасниками) викликає resolveDispute, визначаючи, кому перевести кошти.

Оптимізація Gas Cost — для виконання вимоги до економічної ефективності, код контракту оптимізується шляхом мінімізації запису даних у сховище

блокчейну (Storage). Наприклад, метадані оголошень (опис, зображення) зберігаються off-chain у MongoDB, а в контракт записується лише їхній криптографічний хеш (metadataHash). Це мінімізує обсяг інформації, що зберігається On-chain, суттєво знижуючи Gas Cost.

Тестування — тестування функціоналу контракту проводиться в середовищі розробки (наприклад, Hardhat або Truffle) до розгортання в Testnet. Проводиться Unit Testing для перевірки всіх можливих сценаріїв життєвого циклу угоди, включаючи: успішну транзакцію, повернення коштів при скасуванні, та сценарій вирішення спору Арбітром. Особлива увага приділяється тестуванню на запобігання Reentrancy-атакам та перевірці умов переходу між станами контракту.

4.3. Методика проведення імітаційного моделювання та обрані метрики

Експериментальна частина магістерської дисертації має на меті не лише підтвердити функціональність розробленого Escrow Smart Contract, але й довести відповідність гібридної архітектури критичним нефункціональним вимогам, а саме економічній ефективності та швидкості. Для цього застосовується методика імітаційного моделювання, яка дозволяє відтворити реалістичне навантаження та виміряти ключові показники продуктивності.

4.3.1. Визначення сценаріїв навантаження та експериментальних умов

Імітаційне моделювання проводиться на BNB Smart Chain Testnet для забезпечення реалістичності On-chain умов (час генерації блоку, мережева затримка). Моделювання зосереджується на вимірюванні показників продуктивності Escrow Smart Contract під час виконання ключових транзакцій.

Об'єктом тестування є розгорнутий Escrow Smart Contract (п. 4.2), що є ядром децентралізованої частини ІС. Середовищем тестування є тестова мережа, сумісна з EVM (BNB Smart Chain Testnet). Інструментарій включає програмні

засоби для автоматизації транзакцій, зокрема фреймворки Hardhat або Truffle у поєднанні зі скриптами для імітації масових викликів.

Сценарії моделювання імітують типові дії користувачів, що викликають зміну стану блокчейну, оскільки ці операції є найдорожчими та найповільнішими. Моделюється багаторазовий виклик функції startAgreement з імітацією депонування коштів для точного вимірювання витрат газу для цієї операції. Також моделюється виклик функції confirmDelivery (звільнення коштів) з одночасним вимірюванням часу від моменту надсилання транзакції до її фінального включення в блок і отримання підтвердження, що дозволяє перевірити відповідність вимозі до Latency (не більше 10 секунд). Додатково імітується одночасне високе навантаження (велика кількість транзакцій за короткий проміжок часу) для перевірки стійкості контракту до переповнення пулу транзакцій та оцінки масштабованості.

4.3.2. Методика вимірювання Gas Cost та Latency

Для забезпечення наукової об'єктивності вимірювання ключових метрик проводиться відповідно до стандартизованих методик.

Методика вимірювання Gas Cost (економічна ефективність) передбачає вимірювання кількості Gas Units (одиниць газу), фактично спожитих Smart Contract для виконання кожного функціонального виклику (наприклад, startAgreement). Кінцевим показником є середнє значення Gas Used, яке потім порівнюється з аналогічними показниками неоптимізованих Escrow-контрактів та з вихідною вимогою до економічної ефективності.

Методика вимірювання Latency (продуктивність) передбачає вимірювання часового інтервалу (в секундах) між моментом, коли транзакція була надіслана в мережу (Timestamp Send), і моментом її підтвердження в новому блоці (Timestamp Block Finalized). Вимірювання проводиться для серії транзакцій (мінімум 50) для отримання статистично значущого середнього показника, який ілюструє реальну швидкість роботи децентралізованого компонента. Отримане значення

безпосередньо порівнюється з нефункціональною вимогою до часу підтвердження (10 секунд), підтверджуючи або спростовуючи переваги обраної PoS-платформи.

Результати цих вимірювань будуть використані для остаточного обґрунтування вибору гібридної архітектури та доказу її переваг у фінальному підрозділі.

Висновки до розділу 4

У розділі 4 виконано нефункціональні етапи реалізації та підготовлено експериментальну оцінку гібридної інформаційної системи. Проведено комплексний аналіз вимог до безпеки, продуктивності та надійності, що є критичними для успішної експлуатації системи в реальних умовах, та реалізовано ключові компоненти, що забезпечують відповідність цим вимогам.

Визначено нефункціональні вимоги до безпеки, продуктивності та надійності, які розподілені між on-chain та off-chain компонентами системи відповідно до принципів гібридної архітектури. Для on-chain безпеки встановлено вимоги до криптографічної цілісності Escrow Smart Contract, захисту від Reentrancy-атак та незмінності даних довіри. Для off-chain безпеки визначено вимоги до конфіденційності персональних даних та безпеки API. Вимоги до продуктивності включають швидкість пошуку (off-chain latency не більше 150 мс), час підтвердження транзакції (on-chain latency не більше 10 секунд) та економічну ефективність (мінімізація Gas Cost). Вимоги до надійності передбачають uptime не менше 99.9%, горизонтальну масштабованість та цілісність даних з максимальною затримкою синхронізації 5 секунд.

Реалізовано ключовий елемент системи — Escrow Smart Contract на платформі BNB Smart Chain Testnet з використанням мови Solidity. Контракт реалізує життєвий цикл угоди через функції ініціалізації, виконання та вирішення спорів, з оптимізацією Gas Cost шляхом мінімізації запису даних у блокчейн-

сховище. Проведено тестування функціоналу контракту, включаючи перевірку всіх сценаріїв життєвого циклу угоди та захисту від типових вразливостей.

Розроблено методику імітаційного моделювання для експериментальної оцінки системи. Визначено три сценарії навантаження: ініціація угоди (вимірювання Gas Cost), фіналізація угоди (вимірювання Latency) та стрес-тест (перевірка надійності). Встановлено методику вимірювання Gas Cost та Latency, що дозволяє об'єктивно оцінити відповідність системи нефункціональним вимогам та обґрунтувати переваги гібридної архітектури.

Результати розділу 4 підтверджують готовність системи до експериментальної оцінки та демонструють відповідність встановленим нефункціональним вимогам щодо безпеки, продуктивності та надійності. Розроблений Escrow Smart Contract забезпечує криптографічні гарантії безпеки транзакцій, методика імітаційного моделювання дозволяє кількісно оцінити ефективність рішення, а комплексний підхід до визначення нефункціональних вимог створює основу для успішної експлуатації системи в реальних умовах. Це підтверджує доцільність обраної гібридної архітектури та готовність до переходу до етапу практичної реалізації та тестування всієї системи в цілому.

5 РОЗРОБКА P2P МАРКЕТПЛЕЙСУ

5.1. Розроблення серверної частини маркетплейсу

5.1.1. Використані бібліотеки та технології

Серверна частина реалізована на Node.js з Express. Використано бібліотеки для HTTP-запитів, роботи з базою даних та real-time комунікації. Основні технології та бібліотеки:

– Node.js — середовище виконання JavaScript на сервері, дозволяє створювати масштабовані мережеві додатки та інтегруватися з блокчейн-мережами через асинхронні запити;

– Express.js — фреймворк для веб-серверів на Node.js, обробляє HTTP-запити, маршрутизацію та middleware, забезпечує REST API для фронтенду та обробку запитів з мінімальною латентністю, що відповідає вимогам гібридної архітектури;

– MongoDB — NoSQL база даних для зберігання метаданих оголошень, профілів користувачів та синхронізованих даних з блокчейну, забезпечує швидкий пошук та індексацію, що критично для централізованого компонента гібридної системи;

– Mongoose — ODM для MongoDB, забезпечує моделювання даних, валідацію схем та зручну роботу з колекціями через об'єктно-орієнтований API;

– Socket.io — бібліотека для WebSocket-з'єднань та real-time комунікації, використовується для відстеження онлайн-статусу користувачів та підтримки WebRTC-з'єднань між покупцями та продавцями для P2P-комунікації.

Цей набір технологій забезпечує швидкість обробки запитів, гнучкість зберігання даних та можливість real-time взаємодії, що відповідає вимогам гібридної архітектури.

5.1.2. Архітектура серверної частини маркетплейсу

Серверна частина маркетплейсу побудована за модульною архітектурою на базі Node.js та Express, де основна точка входу ініціалізує застосунок, підключає middleware, конфігурацію доступу, маршрути API та глобальну обробку помилок, така структура забезпечує розділення відповідальностей і спрощує супровід коду при розширенні функціоналу.

На рівні представлення API сервер реалізує REST інтерфейс, який групує ендпоінти за доменними сутностями, зокрема оголошення, користувачі, чат та угоди,

маршрути інкапсулюють правила доступу та підключають проміжні обробники, а бізнес логіка винесена у контролери, що дозволяє підтримувати прозорий потік обробки запиту, валідація, авторизація, виконання операції, формування відповіді.

Рівень доступу до даних реалізований через MongoDB із застосуванням Mon goose, де моделі описують структуру сутностей та правила їх збереження, підхід з окремими моделями для ключових об’єктів дозволяє формалізувати зв’язки між користувачами та оголошеннями, забезпечити індексацію й фільтрацію, а також централізувати правила формування даних, наприклад автоматичні часові мітки та обмеження на поля.

Безпека серверної частини забезпечується набором middleware, які виконують автентифікацію та авторизацію, контролюють доступ до захищених маршрутів і стандартизують обробку помилок, додатково використовується конфігурація CORS і змінні середовища для ізоляції секретів та параметрів запуск у, що дозволяє відокремити налаштування середовища від коду.

Для операцій реального часу в архітектуру включено окремий компонент взаємодії через Socket.IO, який підтримує з’єднання клієнтів, події присутності та обмін службовими повідомленнями, а також використовується як сигнальний рівень для встановлення P2P з’єднання в чаті, у підсумку сервер виконує роль

координаторадля комунікацій і синхронізації стану, тоді як основні CRUD операції та пошук реалізовані через REST API і базу даних.

5.1.3. Безпека серверної частини

Безпека серверної частини маркетплейсу побудована як сукупність організаційних та програмних заходів, які забезпечують контроль доступу до API, захист даних користувачів та зниження ризиків типових веб атак, у межах реалізації застосовано підхід, за якого конфіденційні параметри не зберігаються в коді, а винесені в змінні середовища, що дозволяє ізолювати секрети та розділяти конфігурації для різних середовищ розгортання.

Основним механізмом контролю доступу є автентифікація та авторизація, запити до захищених ендпоінтів виконуються лише після перевірки токена доступу, а роль middleware полягає у централізованій перевірці прав доступу та уніфікації логіки захисту для різних маршрутів, таким чином бізнес логіка контролерів не перевантажується перевірками безпеки, а серверна частина дотримується принципу розділення відповідальностей.

Для захисту облікових даних реалізовано зберігання паролів у вигляді хешів, що виключає можливість отримання паролю у відкритому вигляді навіть у випадку компрометації бази даних, додатково застосовано базові вимоги до коректності вхідних даних та обробки помилок, щоб зменшити ризик ін’єкцій та некоректних запитів, а централізований обробник помилок обмежує витік службової інформації у відповідях, зокрема стеку викликів у виробничому середовищі.

На рівні мережевої взаємодії застосовано налаштування CORS, що обмежує доступ до API з несанкціонованих джерел та зменшує ризик зловживань міждоменними запитами, у поєднанні з керуванням сесіями і контролем доступу це дозволяє забезпечити передбачувану модель взаємодії клієнта та сервера, а також підвищує стійкість до типових атак на веб застосунки, які описані в OWASP Top 10, у підсумку реалізований підхід забезпечує базовий

рівень захисту серверної частини та створює основу для подальшого посилення безпеки при промисловому впровадженні, зокрема через додаткові політики доступу, обмеження частоти запитів та аудит подій безпеки.

5.1.4. Інтеграція блокчейн компонента на серверній частині

У гібридній архітектурі блокчейн компонент використовується як джерело істини для критичних транзакційних операцій, а серверна частина забезпечує швидкодію, зручний користувацький досвід та роботу з некритичними даними через off chain сервіси, основний принцип інтеграції полягає в тому, що сервер не підписує транзакції та не керує коштами користувача, всі on chain дії ініціюються на клієнті через Web3 гаманець, а сервер працює з підтвердженими результатами виконання транзакцій, подіями контракту та станом угод, збереженим у базі даних.

Для узгодження on chain і off chain станів сервер підтримує окрему модель угоди в MongoDB, де зберігаються метадані та посилання на блокчейн, наприклад ідентифікатор угоди, адреси учасників, хеші транзакцій, адресу контракту, останній відомий статус та часові мітки, це дозволяє клієнтському інтерфейсу швидко отримувати історію угод, формувати списки та фільтрувати дані без прямого читання блокчейну, оскільки такі запити є повільнішими та дорожчими з точки зору інфраструктури, створення первинного запису угоди відбувається після ініціації транзакції на клієнті, коли користувач підтверджує дію в гаманці та отримує хеш транзакції, після чого цей хеш використовується як зв’язок між off chain записом і on chain подіями.

Ключовим елементом інтеграції є блокчейн індексатор, який на сервері підключається до RPC провайдера та підписується на події Escrow Smart Contract, наприклад створення угоди, депонування коштів, завершення угоди, відкриття спору та вирішення спору, при надходженні події індексатор знаходить відповідний запис угоди в MongoDB та оновлює статус і пов’язані поля, завдяки цьому сервер підтримує узгодженість даних з блокчейном та

забезпечує швидке відображення стану в інтерфейсі, а також може виконувати похідні дії, наприклад оновлювати репутаційні показники користувача після успішного завершення угоди, такий підхід дозволяє відокремити транзакційний рівень довіри, який забезпечується контрактом, від рівня представлення та сервісних функцій, які реалізуються на сервері.

На рівні API сервер надає ендпоінти для роботи з угодами, які вирішують типові задачі клієнта, отримання списку угод користувача за ролями покупець, продавець, арбітр, перегляд деталей угоди та її історії, підготовка даних для виклику смарт контракту, зокрема адреси контракту та ідентифікатори, та реєстрація транзакційного факту в базі даних після виконання on chain дії, при цьому сервер не замінює блокчейн, а забезпечує шар індексації та доступності, де блокчейн фіксує істинний стан угоди, а база даних використовується як швидкий кеш для інтерфейсу.

З точки зору надійності інтеграції сервер враховує асинхронну природу підтверджень у блокчейні, тому статус угоди може тимчасово відображатися як очікування підтвердження, а фінальний стан встановлюється після отримання події з мережі, при цьому індексатор дозволяє працювати з моделлю eventual consistency, де затримка синхронізації є контрольованою і не впливає на коректність критичних операцій, оскільки вони підтверджуються самим смарт контрактом, у підсумку серверна частина забезпечує практичну реалізацію гібридної моделі, де швидкі операції та користувацькі сервіси залишаються off chain, а довіра, незмінність та безпека транзакцій забезпечуються on chain через Escrow Smart Contract.

5.1.4. Опис потреб для запуску бекенду

Для запуску серверної частини маркетплейсу потрібно виконати кроки з налаштування середовища та конфігурації. Необхідні умови для запуску: встановлення Node.js — версія 16 або вища для підтримки сучасних можливостей JavaScript та сумісності з бібліотеками. встановлення MongoDB — локальний

екземпляр або підключення до хмарного сервісу (MongoDB Atlas). База даних зберігає метадані оголошень, профілі користувачів та синхронізовані дані з блокчейну, налаштування змінних середовища — створення файлу .env з конфігураційними параметрами встановлення залежностей: виконати команду npm install у директорії серверної частини для встановлення пакетів з package.json, встановити змінну MONGO_URI у файлі .env з рядком підключення до MongoDB

Запуск сервера — виконати npm start (production) або npm run dev (development з автоматичним перезапуском). Після запуску сервер слухає вказаний порт та готовий обробляти HTTP-запити та WebSocket-з'єднання.

Після виконання цих кроків сервер готовий приймати запити від клієнтської частини та обробляти їх відповідно до налаштувань маршрутів та логіки контролерів. Блокчейн-індексатор запускається автоматично після старту сервера та починає синхронізацію подій з Smart Contract у MongoDB.

5.1.5. Опис файлів

Серверна частина організована за модульним принципом. Кожен компонент має чітке призначення та відповідає за певну функціональність системи.Головним файлом серверної частини є index.js, який виконує роль точки входу та об'єднує всі компоненти системи. У цьому файлі відбувається ініціалізація Express-сервера, підключення до бази даних MongoDB, налаштування middleware для обробки HTTP-запитів та реєстрація маршрутів для роботи з користувачами, оголошеннями та угодами. Фрагмент коду з ініціалізацією сервера та підключенням маршрутів зображений на рисунку 5.1.

```
import express from "express";
import dotenv from "dotenv";
import cors from "cors";

import { Server } from "socket.io";

dotenv.config();

import cookieParser from "cookie-parser";

import userRoutes from "./routes/userRoutes.js";
import adRoutes from "./routes/adRoutes.js";
import agreementRoutes from "./routes/agreementRoutes.js";

import { notFound, errorHandler } from "./middleware/errorMiddleware.js";
import connectDB from "./config/db.js";
import BlockchainIndexer from "./services/blockchainIndexer.js";

connectDB();

let corsOptions = {
  credentials: true,
  origin: "http://localhost:3000",
};

const port = process.env.PORT || 5000;
const socketPort = process.env.SOCKET_PORT || 5001;

const app = express();

const online = new Set();

app.use(cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());

app.use("/api/users", userRoutes);
app.use("/api/ads", adRoutes);
app.use("/api/agreements", agreementRoutes);

app.use(notFound);
app.use(errorHandler);

app.listen(port, () => {
  console.log(`Server started on port ${port}`);
});
```

Рисунок 5.1 — Фрагмент коду файлу index.js з ініціалізацією Express-сервера

Окрім HTTP-сервера, у файлі index.js реалізовано WebSocket-сервер за допомогою Socket.IO для забезпечення комунікації в реальному часі. Цей сервер працює на окремому порті та відстежує онлайн-статус користувачів, обробляє події WebRTC для встановлення P2P-з'єднань між покупцями та продавцями, а також керує запитами на чат між користувачами. При з'єднанні користувача його ідентифікатор додається до списку онлайн користувачів, а при відключенні — видаляється. Фрагмент коду з реалізацією WebSocket-сервера зображений на рисунку 5.2.

```
const socketServer = app.listen(socketPort, () => console.log(`Socket started on port ${socketPort}`));
const socketIo = new Server(socketServer, {
  transports: ["websocket", "xhr-polling"],
  origins: "*:*",
});

socketIo.on("connection", (socket) => {
  let currentUser = 0;

  socket.on("user", (id) => {
    if (!online.has(id)) online.add(id);
    socket.join(id);
    currentUser = id;
  });
  socket.on("offer", ({ user_id, data }) => {
    if (!currentUser) return;
    socketIo.to(user_id).emit(user_id, { type: "offer", data: data, user: currentUser });
  });
  socket.on("accept", ({ user_id, data }) => {
    if (!currentUser) return;
    socketIo.to(user_id).emit(user_id, { type: "accept", data: data, user: currentUser });
  });
  socket.on("chat_request", ({ user_id, ad_id, ad_name, requester_name }) => {
    if (!currentUser) {
      console.log("❌ chat_request: No currentUser set");
      return;
    }

    // Check if target user is online
    const targetRoom = socketIo.sockets.adapter.rooms.get(user_id);
    const isTargetOnline = targetRoom && targetRoom.size > 0;

    console.log(`📨 chat_request: User ${currentUser} wants to chat with ${user_id} about ad ${ad_id}`);
    console.log(`📊 Target user ${user_id} is ${isTargetOnline ? 'ONLINE' : 'OFFLINE'}`);

    if (!isTargetOnline) {
      console.log(`⚠️ Cannot send chat request: User ${user_id} is not online`);
      // Optionally notify the requester that the user is offline
      socketIo.to(currentUser).emit(currentUser, {
        type: "chat_request_failed",
        reason: "User is offline",
        user_id: user_id
      });
      return;
    }

    console.log(`📬 Emitting chat_request to room: ${user_id}`);
```

Рисунок 5.2 — Фрагмент файлу index.js з ініціалізацією Socket.IO сервера

Маршрутизація запитів організована через файли в папці routes, які визначають API-ендпоінти та пов'язують HTTP-запити з відповідними контролерами. Наприклад, файл adRoutes.js містить маршрути для створення, оновлення, видалення та отримання оголошень, а userRoutes.js — маршрути для реєстрації, автентифікації, виходу та управління профілем користувача. Важливо, що деякі маршрути захищені middleware для перевірки авторизації, що забезпечує безпеку системи. Приклади файлів маршрутів зображені на рисунку 5.3.

```
import express from "express";
import { create, get, list, remove, update } from "../controllers/adController.js";
import { authorize, protect } from "../middleware/authMiddleware.js";

const router = express.Router();

router.post("*", authorize, protect);

router.get("/get", authorize, get);
router.get("/list", list);

router.post("/create", create);
router.post("/update", update);
router.post("/delete", remove);
```

Рисунок 5.3 — Фрагмент коду файлу adRoutes.js

Бізнес-логіка обробки запитів реалізована в контролерах, які знаходяться в папці controllers. Ці файли містять функції, що валідують вхідні дані, взаємодіють з моделями MongoDB через Mongoose, обробляють помилки та повертають відповіді у форматі JSON. Наприклад, adController.js містить функції для створення, оновлення, видалення та отримання оголошень, а userController.js — функції для автентифікації, реєстрації та управління профілем користувача. Особливістю контролерів є можливість додавання додаткових полів до результатів, таких як інформація про онлайн-статус автора оголошення. Приклади контролерів зображені на рисунку 5.4.

```
const create = asyncHandler(async (req, res) => {
  const { name, description } = req.body;
  const user_id = req?.user?._id;

  const adExists = await Ad.findOne({ user_id });

  if (!adExists && user_id) {
    const ad = await Ad.create({ name, description, user_id });
    res.status(201).json(ad);
  } else {
    res.status(409);
    throw new Error("Ad already created");
  }
});

const update = asyncHandler(async (req, res) => {
  const { ad_id, name, description } = req.body;
  const user_id = req?.user?._id;

  const adExists = await Ad.findOne({ user_id, _id: ad_id });

  if (!adExists && user_id) {
    const ad = await Ad.updateOne({ user_id }, { name, description });
    res.status(201).json(ad);
  } else {
    res.status(204);
    throw new Error("Ad does not exist");
  }
});

const remove = asyncHandler(async (req, res) => {
  const { ad_id } = req.body;
  const user_id = req?.user?._id;

  const adExists = await Ad.findOne({ user_id, _id: ad_id });

  if (adExists && user_id) {
    const ad = await Ad.deleteOne({ _id: ad_id });
    res.status(201).json(ad);
  } else {
    res.status(204);
    throw new Error("Ad does not exist");
  }
});
```

Рисунок 5.4 — Фрагмент коду файлу adController.js

Структура даних та схеми для MongoDB визначені в моделях, які знаходяться в папці models. Ці файли використовують Mongoose для визначення структури документів, забезпечення валідації даних та надання методів для роботи з даними. Наприклад, модель adModel.js визначає схему для оголошень з полями user_id, name, description та автоматичними полями timestamps, а модель userModel.js — схему для користувачів з полями name, email, password, walletAddress та reputation. Важливою особливістю моделі користувача є автоматичне хешування паролів перед збереженням у базі даних через метод pre('save'), що забезпечує безпеку зберігання чутливих даних. Приклади моделей зображені на рисунку 5.5.

```
import mongoose, { Schema } from "mongoose";

const adSchema = mongoose.Schema(
  {
    user_id: {
      type: Schema.ObjectId,
      ref: "User",
      required: true,
      unique: true,
      immutable: true,
    },
    name: {
      type: String,
      required: true,
    },
    description: {
      type: String,
      required: true,
    },
  },
  {
    timestamps: true,
  }
);

const Ad = mongoose.model("Ad", adSchema);

export default Ad;
```

Рисунок 5.5 — Фрагменти коду файлів adModel.js

Обробка автентифікації, авторизації та помилок реалізована через middleware, які знаходяться в папці middleware. Функція authorize перевіряє наявність та валідність JWT токенів з cookies запиту, додаючи інформацію про користувача до об'єкта запиту для подальшої обробки. Функція protect захищає маршрути, вимагаючи успішної авторизації, а у разі її відсутності повертає помилку зі статусом 401. Обробка помилок централізована через

функції notFound та errorHandler, які перехоплюють різні типи помилок та повертають відповідні HTTP-статуси та повідомлення клієнту. Приклади middleware зображені на рисунку 5.6.

```
import jwt from "jsonwebtoken";
import asyncHandler from "express-async-handler";
import User from "../models/userModel.js";

const authorize = asyncHandler(async (req, res, next) => {
  let token;
  token = req.cookies.jwt;

  if (token) {
    try {
      const decoded = jwt.verify(token, process.env.JWT_SECRET);
      req.user = await User.findById(decoded.userId).select("-password");
    } catch (error) {
      req.authError = "Not authorized, invalid token";
      res.status(401);
    }
  } else {
    req.authError = "Not authorized, no token";
  }

  next();
});

const protect = async (req, res, next) => {
  if (req.authError) {
    res.status(401);
    throw new Error(req.authError);
  }

  next();
};

export { protect, authorize };
```

Рисунок 5.6 — Фрагмент коду файлу authMiddleware.js

Окремою важливою частиною серверної архітектури є блокчейн-індексатор, який знаходиться в папці services. Цей сервіс автоматично ініціалізується після запуску сервера та відповідає за синхронізацію подій з Escrow Smart Contract у MongoDB. Індексатор постійно моніторить події на блокчейні, такі як створення угоди, депонування коштів, підтвердження доставки та вирішення спорів, та оновлює відповідні записи в базі даних. Це забезпечує швидкий доступ до інформації про стан угод без необхідності постійних запитів до блокчейну, що є ключовим елементом гібридної архітектури, обґрунтованої в розділі 3.

Така організація файлів забезпечує чітке розділення відповідальності між компонентами системи, що спрощує підтримку та розширення функціональності маркетплейсу.

5.2. Реалізація клієнтської частини маркетплейсу

5.2.1. Використані бібліотеки та технології

Клієнтська частина реалізована на React з підтримкою Web3 для взаємодії з блокчейном. Використано бібліотеки для UI, управління станом та real-time комунікації.

Основні технології та бібліотеки:

– React.js — основа клієнтської частини, компонентна архітектура дозволяє створювати інтерактивний інтерфейс з динамічним оновленням без перезавантаження сторінки [16];

– Redux Toolkit — управління глобальним станом, централізує дані про користувачів, оголошення, угоди та стан Web3-з'єднання, що спрощує синхронізацію між компонентами;

– React Router — навігація між сторінками, забезпечує маршрутизацію без перезавантаження сторінки та захист маршрутів, що вимагають авторизації;

– Ethers.js — інтеграція з блокчейном, забезпечує підключення до MetaMask, взаємодію з Smart Contract та виконання транзакцій, критично для роботи з Escrow Smart Contract;

– Socket.IO-client — real-time комунікація з сервером, використовується для відстеження онлайн-статусу користувачів та обміну сигналами WebRTC для встановлення P2P-з'єднань між покупцями та продавцями;

– React Bootstrap — UI-компоненти, надає готові компоненти для швидкого створення інтерфейсу з адаптивним дизайном;

– Simple-peer — WebRTC для P2P-комунікації, дозволяє встановлювати прямі з'єднання між користувачами для чату безпосередньо в браузері, що відповідає принципам P2P-архітектури.

Цей набір технологій забезпечує швидкий інтерфейс, інтеграцію з блокчейном та можливість P2P-комунікації, що відповідає вимогам гібридної архітектури.

5.2.2. Архітектура клієнтської частини маркетплейсу

Клієнтська частина реалізована як односторінковий веб застосунок на базі React, де архітектура побудована за принципом розділення відповідальностей між рівнем представлення, рівнем керування станом і рівнем взаємодії з зовнішніми сервісами, запуск застосунку відбувається через точку входу index.js, яка підключає кореневий компонент, ініціалізує глобальні провайдери та налаштовує маршрутизацію.

На рівні навігації використовується React Router, який організовує переходи між основними екранами, домашня сторінка, керування оголошеннями, профіль, авторизація та реєстрація, кожен екран відповідає за відображення конкретного сценарію користувача, а повторно використовувані елементи інтерфейсу винесені в компоненти, що спрощує підтримку UI та дозволяє масштабувати застосунок без дублювання логіки відображення.

Централізоване керування станом реалізоване через Redux Store, який об'єднує окремі слайси під доменні підсистеми, авторизація, оголошення, чат і сокет підключення, такий підхід дозволяє уніфікувати доступ до даних у різних частинах інтерфейсу, забезпечити передбачуваність оновлень стану та спростити синхронізацію UI з результатами запитів до API, для комунікації з сервером використовується окремий API шар, який інкапсулює HTTP запити, обробку відповідей та оновлення стану.

Для взаємодії в режимі реального часу у клієнтській архітектурі використовується модуль Socket.IO client, який відповідає за підключення до сервера, підписку на події, керування кімнатами користувачів та обмін повідомленнями, додатково винесення логіки підключення в кастомні хуки дозволяє повторно використовувати цю функціональність у різних компонентах, не змішуючи мережеву логіку з логікою відображення, аналогічно, для реалізації P2P взаємодії у чаті використовується окремий хук для роботи з піринговими з'єднаннями, де клієнт виконує ініціацію та обробку сигналізації через сервер, а передача даних відбувається між користувачами напряму

Таким чином архітектура клієнта забезпечує модульність, розділення UI і бізнес логіки, централізоване керування станом та підтримку як класичної взаємодії через REST API, так і сценаріїв реального часу через WebSocket і P2P, що робить інтерфейс керованим, розширюваним і придатним до подальшого розвитку.

5.2.3. Інтеграція Web3 на клієнті

Інтеграція Web3 у клієнтській частині реалізована як окремий рівень взаємодії з блокчейн мережею, який не залежить від серверного компонента в частині виконання транзакцій, користувач підключає криптогаманець у браузері після чого клієнт отримує доступ до адреси акаунта, поточної мережі та провайдера, підпис транзакцій і підтвердження операцій виконуються в гаманці, що забезпечує принцип, за якого критичні дії з коштами залишаються під контролем користувача.

Для взаємодії зі смарт контрактом використовується Web3 бібліотека, яка формує об’єкт контракту на основі адреси розгортання та ABI інтерфейсу, клієнт викликає читальні методи для отримання стану угод та параметрів, а також ініціює транзакційні виклики для виконання дій життєвого циклу угоди, зокрема створення угоди, депонування коштів, підтвердження виконання та ініціацію спору, при цьому кожна транзакція проходить стандартний цикл, формування даних виклику, підпис у гаманці, відправлення в мережу та очікування підтвердження, а в інтерфейсі відображаються стани очікування та результат виконання.

У рамках гібридної моделі клієнт узгоджує on chain та off chain частини так, щоб користувач отримував цілісне представлення процесу, серверна частина використовується для зберігання метаданих, швидкого пошуку та відображення списків, тоді як факт виконання угоди фіксується в блокчейні, після підтвердження транзакції клієнт може ініціювати оновлення запису угоди на

сервері, зберігаючи хеш транзакції та ідентифікатор угоди для подальшої індексації, такий підхід зменшує навантаження на блокчейн з боку некритичних операцій і одночасно забезпечує незмінність критичного стану угоди.

Таким чином інтеграція Web3 на клієнті забезпечує пряме виконання транзакцій через Escrow Smart Contract, прозорість та контроль користувача над підписом операцій, а також узгодженість з централізованими сервісами системи для зручного відображення станів та історії угод.

5.2.4. Реалізація WebRTC на клієнті

Реалізація WebRTC у клієнтській частині маркетплейсу призначена для організації прямої P2P взаємодії між користувачами під час приватного спілкування, де основний обмін даними відбувається безпосередньо між браузерами учасників, а сервер використовується лише як сигнальний вузол для початкового узгодження параметрів з'єднання, такий підхід відповідає ідеї P2P архітектури та дозволяє зменшити навантаження на сервер у порівнянні з моделлю, коли всі повідомлення проходять через центральний вузол (Додаток К).

У клієнтській частині WebRTC інтегровано через окремий модуль або кастомний хук, який інкапсулює життєвий цикл пірингового з'єднання та забезпечує взаємодію з інтерфейсом користувача, цей модуль відповідає за ініціалізацію peer об'єкта, збереження активних підключень у внутрішній структурі даних, обробку подій встановлення і розриву каналу, а також за відправлення і прийом повідомлень, логіка з'єднання винесена з UI компонентів, що спрощує підтримку та дозволяє повторно використовувати однакову реалізацію на різних екранах.

Для спрощення роботи з низькорівневими механізмами WebRTC застосовується бібліотека над RTCPeerConnection, яка автоматизує формування SDP описів, обмін ICE кандидатами та створення data channel, у результаті клієнт реалізує стабільний сценарій встановлення з'єднання між двома користувачами без необхідності вручну керувати всіма деталями протоколу, при цьому

зберігається можливість контролювати ключові стани, зокрема ініціатор чи отримувач, готовність каналу та помилки підключення.

Процес роботи WebRTC складається з двох фаз, сигналізації та прямого обміну даними, на фазі сигналізації використовується Socket.IO як транспорт для передачі службових повідомлень між користувачами, коли ініціатор хоче розпочати чат, він створює peer з'єднання та генерує offer, який через сокет надсилається адресату разом з ідентифікатором кімнати або користувача, отримувач створює власний peer об'єкт, застосовує отриманий offer та генерує accept, який повертається ініціатору, після цього обидві сторони завершують узгодження параметрів та переходять до встановлення прямого каналу, на практиці сервер у цьому сценарії не читає та не зберігає вміст повідомлень, а лише маршрутизує сигналізацію.

Після завершення сигналізації встановлюється прямий P2P канал, через який обмін повідомленнями відбувається через WebRTC data channel, клієнт відстежує події відкриття каналу та забезпечує доставку повідомлень у чат інтерфейс, для підвищення надійності реалізується обробка розривів з'єднання та сценарії повторної ініціації, наприклад у випадку зміни мережі, тимчасової втрати зв'язку або перезапуску вкладки, у таких ситуаціях клієнт може повторно створити peer та повторити процедуру offer accept, використовуючи сокетний канал як механізм повторної сигналізації.

Окремо важливою частиною реалізації є керування сесією та присутністю користувачів, клієнт повідомляє сервер про активність користувача та приєднання до кімнати, що дозволяє визначити, чи доступний адресат для встановлення P2P з'єднання, а також коректно закривати підключення при виході з системи, для цього використовуються події підключення та відключення сокета і окремі події, що повідомляють про статус користувача, у підсумку WebRTC з'єднання будується лише тоді, коли обидві сторони доступні, а клієнтський інтерфейс може показувати стан, онлайн, встановлення з'єднання, підключено, з'єднання втрачено.

Таким чином реалізація WebRTC на клієнті забезпечує прямий P2P канал для комунікації між користувачами, зменшує затримки та навантаження на серверну

частину, а застосування Socket.IO як сигнального рівня дозволяє узгоджувати параметри з'єднання, керувати сесією та забезпечувати повторне підключення, що робить чат функціонально придатним для використання в межах маркетплейсу та узгоджується з концепцією P2P взаємодії в системі.

5.2.5. Опис потреб для запуску клієнта

Для запуску клієнтської частини маркетплейсу потрібно виконати кроки з налаштування середовища та конфігурації. Необхідні умови для запуску:

– встановлення Node.js — версія 16 або вища для сумісності з React та сучасними бібліотеками;

– налаштування змінних середовища — створення файлу .env у директорії клієнта з параметрами для інтеграції з блокчейном. Потрібно вказати адресу розгорнутого Escrow Smart Contract та RPC-ендпоінт BNB Smart Chain Testnet для підключення до блокчейн-мережі;

– встановлення MetaMask — розширення браузера для взаємодії з блокчейном. Користувачі мають встановити MetaMask та налаштувати підключення до BNB Smart Chain Testnet для виконання транзакцій через Escrow Smart Contract.

Процес запуску клієнтської частини:

– встановлення залежностей — виконати команду npm install у директорії клієнта для встановлення пакетів з package.json, включаючи React, Redux Toolkit, Ethers.js та інші бібліотеки;

– налаштування блокчейн-інтеграції;

– запуск клієнта — виконати команду npm start для запуску development-сервера. Додаток відкриється в браузері на порту 3000 за замовчуванням та автоматично перезавантажиться при зміні коду.

Після виконання цих кроків клієнтська частина готова до роботи. Користувачі можуть реєструватися, авторизуватися, переглядати оголошення, підключати гаманець MetaMask та створювати угоди через Escrow Smart Contract.

Важливо, що клієнтська частина працює незалежно від серверної, але потребує підключення до серверного API для отримання метаданих оголошень та синхронізації стану угод. Безпосередня взаємодія з блокчейном відбувається через MetaMask та Ethers.js, що відповідає принципам децентралізованої архітектури.

5.2.6. Опис файлів

Клієнтська частина організована за компонентною архітектурою React з використанням Redux для управління станом. Кожен компонент має чітке призначення та відповідає за певну функціональність інтерфейсу. Головним файлом клієнтської частини є index.jsx, який виконує роль точки входу та ініціалізує React-додаток. У цьому файлі відбувається рендеринг головного компонента App у кореневий DOM-вузол, підключення Redux Store для глобального управління станом та налаштування React Router для маршрутизації між сторінками. Фрагмент коду з ініціалізацією додатку зображений на рисунку 5.7.

```
import * as process from "process";

window.global = window;
window.process = process;
window.Buffer = [];

const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path="/" element={<App />}>
      <Route index={true} path="/" element={<HomeScreen />} />
      <Route path="/login" element={<LoginScreen />} />
      <Route path="/register" element={<RegisterScreen />} />
      <Route path="" element={<PrivateRoute />}>
        <Route path="/profile" element={<ProfileScreen />} />
        <Route path="/manage" element={<ManageScreen />} />
        <Route path="/agreements" element={<AgreementsScreen />} />
      </Route>
    </Route>
  )
);

ReactDOM.createRoot(document.getElementById("root")).render(
  <Provider store={store}>
    <React.StrictMode>
      <RouterProvider router={router} />
    </React.StrictMode>
  </Provider>
);
```

Рисунок 5.7 — Файл index.jsx з ініціалізацією React-додатку

Централізоване управління станом реалізовано через Redux Store, який налаштовується у файлі store.js. Цей файл об'єднує всі Redux slices (модулі стану)

та middleware, створюючи централізований механізм керування даними додатку. Store містить стан авторизації користувачів, оголошень, угод, чату та WebSocket-з'єднань, що забезпечує синхронізацію даних між компонентами. Фрагмент коду з налаштуванням Redux Store зображений на рисунку 5.8.

```
import { configureStore } from "@reduxjs/toolkit";

import adReducer from "./slices/adSlice";
import authReducer from "./slices/authSlice";
import chatReducer from "./slices/chatSlice";
import socketReducer from "./slices/socketSlice";

import { apiSlice } from "./slices/apiSlice";
import { rtkQueryErrorLogger } from "./slices/middleware";

const store = configureStore({
  reducer: {
    [apiSlice.reducerPath]: apiSlice.reducer,
    ad: adReducer,
    auth: authReducer,
    chat: chatReducer,
    socket: socketReducer,
  },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware()
      .concat(rtkQueryErrorLogger)
      .concat(apiSlice.middleware),
  devTools: true,
});

export default store;
```

Рисунок 5.8 — Фрагмент коду файлу store.js

Кожен екран відповідає за відображення певної сторінки та взаємодію з користувачем через інтерфейс. Наприклад, HomeScreen.jsx містить компонент домашньої сторінки, який відображає список оголошень з можливістю фільтрації та пошуку. LoginScreen.jsx та RegisterScreen.jsx містять форми для автентифікації та реєстрації користувачів. ManageScreen.jsx дозволяє користувачам переглядати, редагувати та видаляти свої оголошення, а ProfileScreen.jsx — переглядати та оновлювати інформацію про профіль, включаючи прив'язку гаманця для взаємодії з блокчейном. Приклади екранів зображені на рисунку 5.9.

```jsx
import AdList from "../components/AdList";

const HomeScreen = () => {
  return (
    <div className="w-full h-fit min-h-full border-1 m-auto flex">
      <AdList />
    </div>
  );
};
export default HomeScreen;
```

Рисунок 5.9 — Фрагмент коду файлу HomeScreen.jsx

Управління станом організовано через Redux slices, які знаходяться в папці slices. Кожен slice відповідає за конкретний аспект функціональності додатку. Наприклад, authSlice.js керує станом авторизації та зберігає інформацію про поточного користувача, adSlice.js — станом оголошень та фільтрами, chatSlice.js — історією чату та повідомленнями, а socketSlice.js — підключенням до WebSocket-сервера. Окремо виділяються API slices, такі як apiSlice.js, adApiSlice.js, usersApiSlice.js та agreementApiSlice.js, які використовують Redux Toolkit для взаємодії з серверним API через асинхронні запити. Ці slices автоматично оновлюють стан додатку на основі відповідей від сервера, що спрощує роботу з асинхронними даними. Приклади slices зображені на рисунку 5.10.

```jsx
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
  userInfo: localStorage.getItem('userInfo')
    ? JSON.parse(localStorage.getItem('userInfo'))
    : null,
};

const authSlice = createSlice({
  name: 'auth',
  initialState,
  reducers: {
    setCredentials: (state, action) => {
      state.userInfo = action.payload;
      localStorage.setItem('userInfo', JSON.stringify(action.payload));
    },
    logout: (state, action) => {
      state.userInfo = null;
      localStorage.removeItem('userInfo');
    },
  },
});

export const { setCredentials, logout } = authSlice.actions;

export default authSlice.reducer;
```

Рисунок 5.10 — Фрагмент коду файлу authSlice.js

Важливою частиною клієнтської архітектури є кастомні React hooks, які знаходяться в папці hooks. Хук useWeb3.jsx забезпечує підключення до MetaMask, взаємодію з блокчейном через Ethers.js та виклики функцій Escrow Smart Contract. Хук useSocket.jsx керує WebSocket-з'єднанням з сервером для real-time комунікації, відстеження онлайн-статусу користувачів та обробки подій WebRTC. Хук usePeers.jsx реалізує логіку встановлення P2P-з'єднань між користувачами за допомогою WebRTC та Simple-peer для прямого обміну повідомленнями в браузері. Фрагмент коду з реалізацією WebSocket-з'єднання зображений на рисунку 5.11.

```
import React from "react";
import { useSelector } from "react-redux";
import openSocket from "socket.io-client";
import { toast } from "react-toastify";

import { usePeer } from "./peers";

const socket = openSocket("localhost:5001", {
  transports: ["websocket", "xhr-polling"],
  origins: "*:*",
});

const useSocket = () => {
  const { userInfo } = useSelector((state) => state.auth);
  const { peerControls } = usePeer();

  const [status, setStatus] = React.useState(null);

  React.useEffect(() => {
    socket.on("connect", () => {
      setStatus("connect");
    });
    socket.on("disconnect", () => {
      setStatus("disconnect");
    });
    socket.on("error", () => {
      setStatus("error");
    });
    socket.on("reconnect", () => {
      setStatus("reconnect");
    });

    const handleBeforeUnload = (event) => {
      event.preventDefault();
      event.returnValue = "";
      if (userInfo) {
        socket.off(userInfo._id);
        socket.emit("user_out", userInfo._id || 0);
      }
    };

    window.addEventListener("beforeunload", handleBeforeUnload);

    return () => {
      window.removeEventListener("beforeunload", handleBeforeUnload);
    };
  }, []);
```

Рисунок 5.11 — Фрагмент коду файлу socket.jsx

Особливу роль відіграє хук usePeers.jsx, який реалізує механізм встановлення P2P-з'єднань між користувачами для прямого чату безпосередньо в браузері. Цей хук використовує WebRTC для створення peer-to-peer з'єднань, обробляє сигнали offer та accept через WebSocket для обміну інформацією про з'єднання та керує життєвим циклом P2P-з'єднань. Фрагмент коду з реалізацією P2P-логіки зображений на рисунку 5.12.

```
const usePeer = () => {
  const { socket } = useSocketControl();
  const { openNewWindow } = useNewWindow();

  const savePeer = (peer, sendRoomId) => {
    peers.set(sendRoomId, peer);
  };

  const openChatWindow = (peerId) => {
    // Check if window already exists and is still open
    const existingWindow = chatWindows.get(peerId);
    if (existingWindow && !existingWindow.closed) {
      console.log(`ℹ️ Chat window already open for ${peerId}`);
      existingWindow.focus();
      return existingWindow;
    }

    console.log(`🪟 Opening chat window for ${peerId}`);
    const chatWindow = openNewWindow(<Chat id={peerId} />);

    if (chatWindow) {
      chatWindows.set(peerId, chatWindow);
      chatWindow.addEventListener("beforeunload", () => {
        chatWindows.delete(peerId);
      });
      console.log(`✅ Chat window opened for ${peerId}`);
      return chatWindow;
    }

    console.warn(`⚠️ Failed to open chat window for ${peerId}`);
    return null;
  };

  const handleOffer = (id) => {
    const peer = peers.get(id);
    if (!peer) {
      console.error(`Peer ${id} not found for offer`);
      return;
    }

    let offerSent = false;

    peer.on("signal", (signalData) => {
      console.log(`📤 Initiator signal for ${id}:`, signalData.type || "offer");
```

Рисунок 5.12 — Фрагмент коду файлу peers.jsx

Окрім хуків, клієнтська частина містить переісні компоненти в папці components, які використовуються на різних екранах. Наприклад, Header.jsx містить навігаційну панель з посиланнями на різні сторінки та кнопкою підключення гаманця, AdList.jsx — компонент для відображення списку оголошень, AgreementCard.jsx — картку угоди з інформацією про стан Escrow та можливістю підтвердження доставки або виклику спору, а Chat.jsx —

компонент для відображення P2P-чату між користувачами. Компонент WalletConnect.jsx забезпечує інтерфейс для підключення MetaMask та відображення інформації про підключений гаманець.

Особливістю клієнтської архітектури є інтеграція з блокчейном через хук useWeb3.jsx, який забезпечує підключення до MetaMask, створення екземпляра провайдера Ethers.js для взаємодії з BNB Smart Chain та надання методів для виклику функцій Escrow Smart Contract, таких як створення угоди, депонування коштів, підтвердження доставки та виклик спору. Ця інтеграція дозволяє користувачам безпосередньо взаємодіяти з блокчейном через інтерфейс додатку, що відповідає принципам децентралізованої архітектури.

Така організація файлів забезпечує модульність та підтримуваність клієнтської частини, дозволяючи легко додавати нову функціональність та модифікувати існуючу без впливу на інші компоненти системи.

5.3. Демонстрація роботи системи

Після реалізації серверної та клієнтської частин система готова до використання (Додаток Л). Нижче показано основні сценарії роботи маркетплейсу:

1. Реєстрація та авторизація користувачів — перший крок реєстрація нового користувача або вхід в систему, екран реєстрації дозволяє ввести ім'я, електронну пошту та пароль для створення облікового запису, після успішної реєстрації користувач автоматично авторизується та отримує доступ до функцій маркетплейсу, екран входу дозволяє авторизуватися існуючим користувачам за допомогою електронної пошти та пароля, скріншот екрану реєстрації показано на рисунку 5.13.

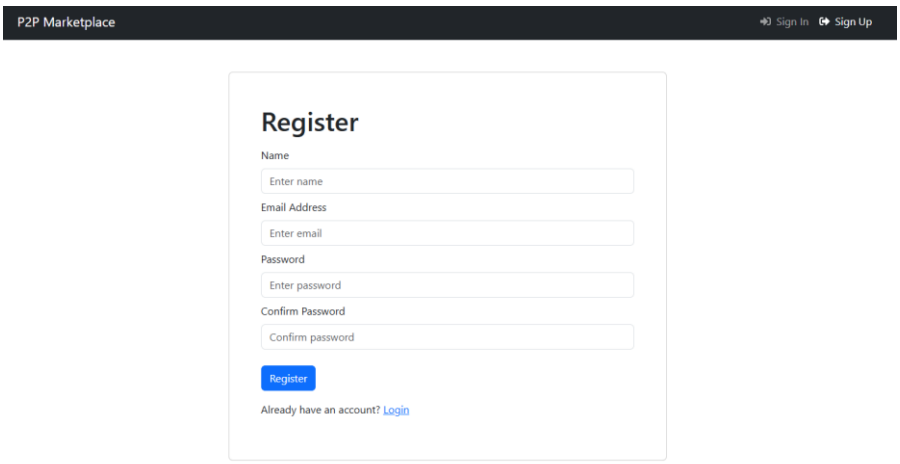

Рисунок 5.13 — Скріншот екрану реєстрації користувача з формою для введення імені, електронної пошти та пароля

2. Перегляд та створення оголошень — після авторизації користувач потрапляє на домашню сторінку, де відображається список доступних оголошень, користувач може переглядати оголошення, фільтрувати їх за категоріями та виконувати пошук за назвою або описом, для створення власного оголошення користувач переходить на сторінку управління, де може додати нове оголошення з описом товару або послуги, встановити ціну та завантажити зображення, скріншот домашньої сторінки зі списком оголошень показано на рисунку 5.14.

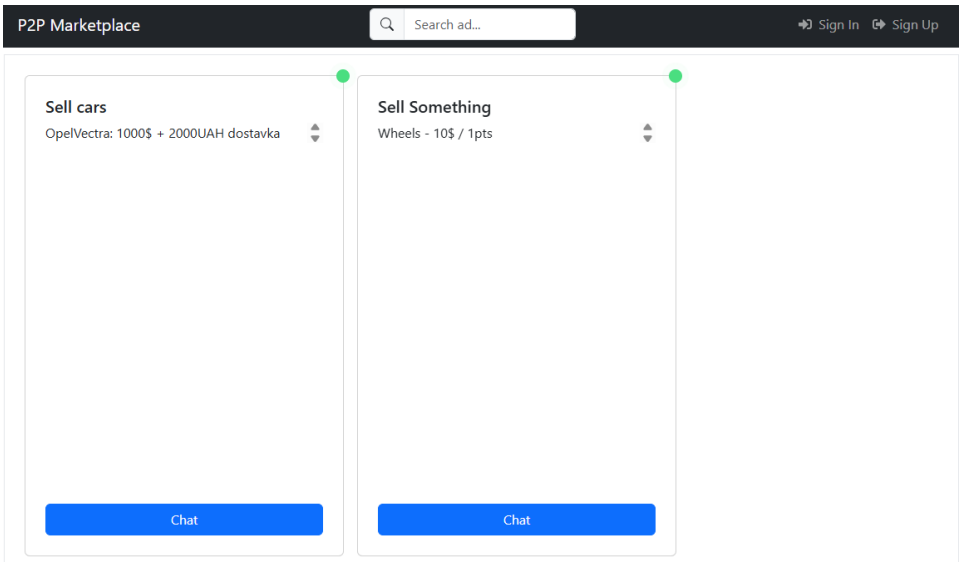

Рисунок 5.14 — Скріншот домашньої сторінки маркетплейсу зі списком оголошень, полем пошуку та навігаційним меню

3. Підключення гаманця та створення угоди — для створення угоди через Escrow Smart Contract користувач має підключити гаманець MetaMask, після підключення користувач може переглянути деталі оголошення та натиснути кнопку "Створити угоду", що ініціює транзакцію на блокчейні, при створенні угоди кошти покупця депонуються на Escrow Smart Contract та блокуються до моменту підтвердження доставки або вирішення спору, скріншот екрану створення угоди з підтвердженням транзакції в MetaMask показано на рисунку 5.15.

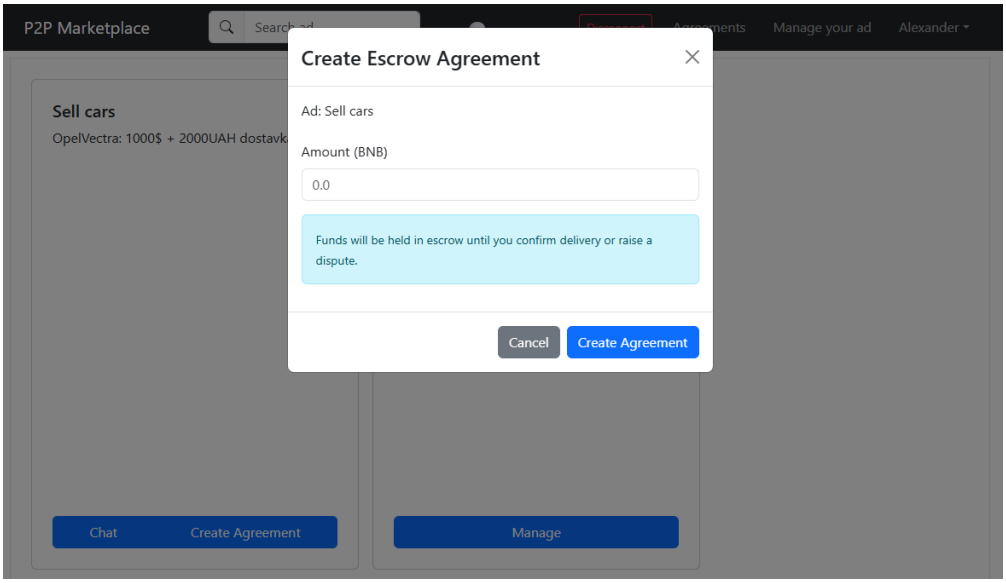

Рисунок 5.15 — Скріншот екрану створення угоди з відображенням деталей оголошення, кнопкою "Створити угоду" та вікном підтвердження транзакції MetaMask

4. Управління угодами та підтвердження доставки — користувач може переглядати всі свої угоди на спеціальній сторінці, де відображається стан кожної угоди (Deposited, Disputed, Completed), після отримання товару покупець може підтвердити доставку, що автоматично звільняє кошти продавцю через Escrow Smart Contract, у разі проблем будь-яка зі сторін може викликати спір, який вирішується арбітром, скріншот сторінки управління угодами показано на рисунку 5.16.

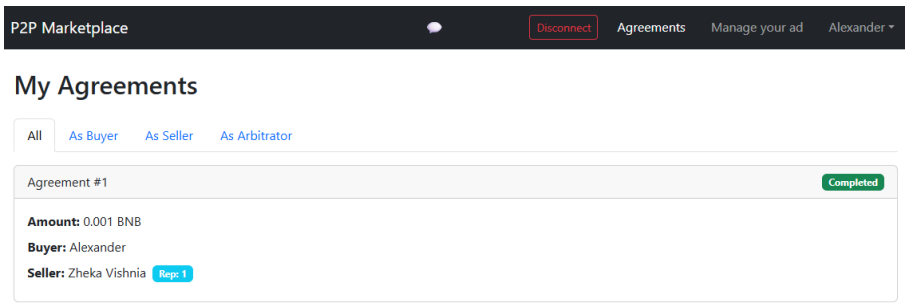

Рисунок 5.16 — Скріншот сторінки управління угодами зі списком угод, їх станами (Deposited, Completed) та кнопками для підтвердження доставки або виклику спору

5. P2P-комунікація між користувачами — система підтримує P2P-чат між покупцями та продавцями через WebRTC, користувач може надіслати запит на чат з автором оголошення, після чого встановлюється пряме P2P-з'єднання між браузерами користувачів, чат відкривається в окремому вікні та дозволяє обмінюватися повідомленнями в реальному часі безпосередньо між користувачами, що відповідає принципам P2P-архітектури, скріншот P2P-чату між користувачами показано на рисунку 5.17.

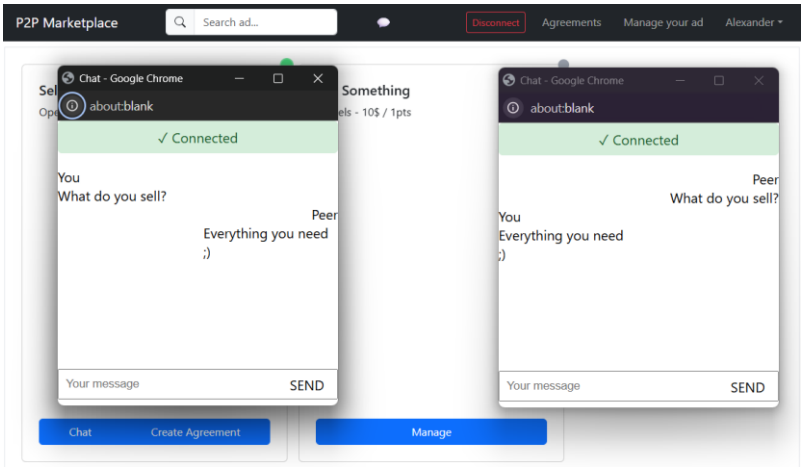

Рисунок 5.17 — Скріншот P2P-чату між покупцем та продавцем у окремому вікні з історією повідомлень та полем для введення нового повідомлення

6. Профіль користувача та прив'язка гаманця — на сторінці профілю користувач може переглядати та редагувати свою інформацію, включаючи ім'я та електронну пошту, важливою функцією є прив'язка адреси гаманця до профілю, що необхідно для участі в угодах через Escrow Smart Contract, після прив'язки гаманця користувач може створювати угоди та взаємодіяти з блокчейном, скріншот сторінки профілю з прив'язкою гаманця показано на рисунку 5.18.

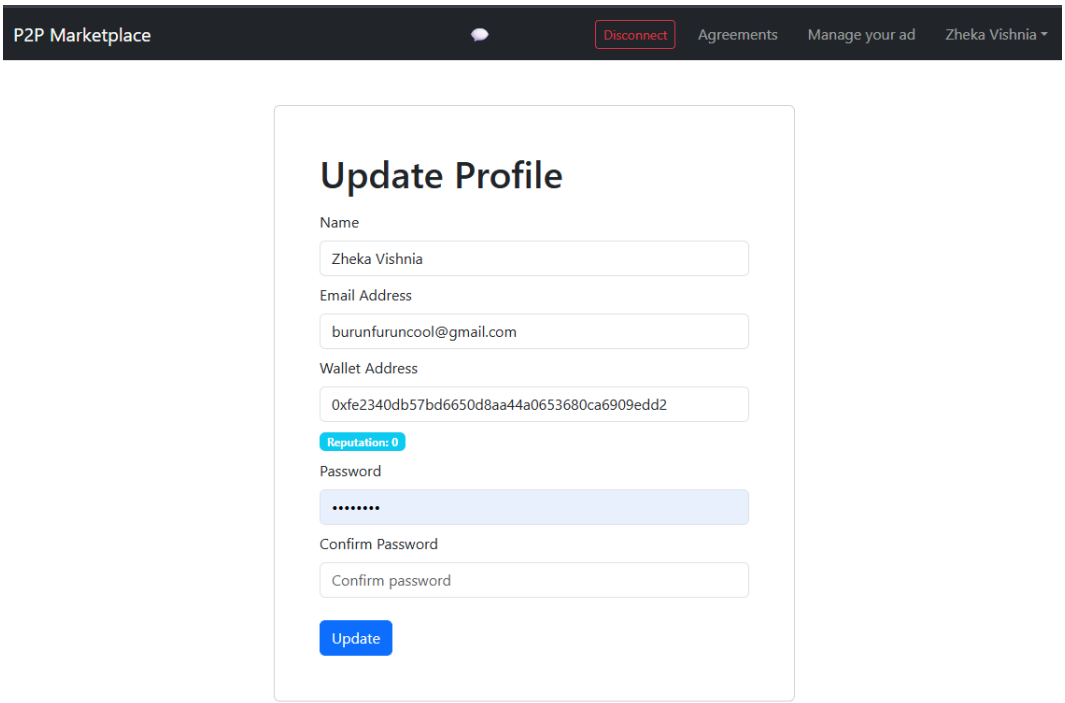

Рисунок 5.18 — Скріншот сторінки профілю користувача з інформацією про профіль, прив'язаною адресою гаманця та можливістю редагування даних

Демонстрація підтверджує, що система функціонує відповідно до обґрунтованої гібридної архітектури, поєднуючи швидкість централізованих компонентів для пошуку та UI з безпекою децентралізованих компонентів для фінансових транзакцій через блокчейн.

Висновки до розділу 5

У розділі 5 виконано розробку P2P-маркетплейсу, реалізовано серверну та клієнтську частини гібридної інформаційної системи.

Серверна частина побудована на Node.js з використанням Express для веб-сервера, MongoDB та Mongoose для роботи з базою даних, Socket.IO для обміну даними в реальному часі. Реалізовано REST API з маршрутами для управління оголошеннями та користувачами, контролери для обробки бізнес-логіки, моделі даних для оголошень та користувачів, middleware для автентифікації та обробки помилок. WebSocket-сервер забезпечує комунікацію в реальному часі та підтримку списку онлайн-користувачів.

Клієнтська частина розроблена на React з використанням Redux для управління станом, React Router для навігації, Socket.IO-client для реального часу та WebRTC (simple-peer) для P2P-з'єднань. Реалізовано екрани (HomeScreen, LoginScreen, RegisterScreen, ProfileScreen, ManageScreen), компоненти інтерфейсу (AdList, Chat, Header, Loader), Redux slices для управління станом (authSlice, adSlice, chatSlice, socketSlice), API slices для взаємодії з сервером та кастомні хуки для роботи з сокетами та P2P-з'єднаннями.

Система забезпечує повний функціонал P2P-маркетплейсу: створення та управління оголошеннями, автентифікацію користувачів, комунікацію в реальному часі через WebSocket, P2P-з'єднання для приватного чату між користувачами, пошук та фільтрацію оголошень. Архітектура модульна та масштабована, що дозволяє легко додавати новий функціонал та підтримувати систему.

Результати розділу 5 підтверджують готовність системи до експлуатації та демонструють реалізацію всіх запланованих функціональних можливостей гібридної P2P-системи обміну товарами та послугами.

6 СТАРТАП ПРОЄКТ

6.1. Опис ідеї проєкту

Розроблений P2P-маркетплейс з блокчейн ескроу — інформаційна система для обміну товарами та послугами між користувачами з автоматичним забезпеченням безпеки транзакцій через Smart Contracts.

В даному підрозділі будуть представлені таблиці 6.1 — 6.2, що описують ідею проєкту.

Таблиця 6.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Децентралізований P2P-маркетплейс з гібридною архітектурою	Обмін товарами між приватним особами	Автоматичне забезпечення безпеки транзакцій через Escrow Smart Contract
	Надання послуг приватним особам	Прозорість транзакцій, незмінна історія репутації
	Продаж товарів малими та середніми підприємствами	Зменшення витрат на комісії, захист від цензури та маніпуляцій з боку платформи
	Обмін цифровими активами та послугами	Пряма P2P-комунікація між користувачами, відсутність залежності від централізованого посередника

Таблиця 6.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	Цей проєкт	OLX	Bisq	Prom.ua	W	N	S
1	Швидкість пошуку та відображення оголошень	Висока	Висока	Низька (P2P-пошук)	Висока	-	-	S
2	Транзакційні витрати (комісії)	Низькі	Високі	Високі (Gas Cost)	Високі (комісія платформи)	-	-	S
3	Безпека транзакцій	Висока	Середня	Висока (Multi-sig)	Середня (довіра до платформи)	-	-	S
4	Захист від цензури	Високий	Низька	Високий	Низький	-	-	S
5	Зручність інтерфейсу	Висока	Висока	Низька (P2P-клієнт)	Висока	-	-	S
6	Час підтвердження транзакції	Середній	Миттєва	Високий (залежить від блокчейну)	Миттєвий	W	-	-
7	Залежність від криптогаманця	Так	Ні	Так	Ні	W	-	-
8	Масштабованість	Висока	Висока	Низька	Висока	-	-	S

№ п/п	Техніко-економічні характеристики ідеї	Цей проєкт	OLX	Bisq	Prom.ua	W	N	S
9	Прозорість репутації	Висока	Низька	Висока	Низька (централізована)	-	-	S
10	Надійність системи	Висока	Низька	Висока	Низька (SPOF)	-	-	S

6.2. Технологічний аудит ідеї проєкту

В даному підрозділі буде наведено таблицю 6.3, що демонструє ефективність реалізації системи.

Таблиця 6.3 — Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Гібридна P2P-архітектура з централізованим компонентом	Node.js, Express.js для серверної частини	Наявні, широко використовуються	Доступні, відкриті технології з великою спільнотою
2	Клієнтський інтерфейс з високою швидкістю відгуку	React.js, Redux Toolkit для управління станом	Наявні, стандартні для веб-розробки	Доступні, безкоштовні бібліотеки з документацією

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
3	Інтеграція з блокчейном та Smart Contracts	Ethers.js, Web3.js для взаємодії з блокчейном	Наявні, стандартні для Web3-розробки	Доступні, відкриті бібліотеки з активною підтримкою
4	Розробка та розгортання Escrow Smart Contract	Solidity для написання контракту, Hardhat/Truffle для тестування	Наявні, стандартні для блокчейн-розробки	Доступні, безкоштовні інструменти з навчальними матеріалами
5	Зберігання метаданих та швидкий пошук	MongoDB для зберігання даних, Mongoose для роботи з БД	Наявні, широко використовуються	Доступні, безкоштовні для розробки
6	Real-time комунікація та P2P-з'єднання	Socket.io для WebSocket, WebRTC (Simple-peer) для P2P	Наявні, стандартні для real-time додатків	Доступні, відкриті бібліотеки з прикладами використання
7	Блокчейн-індексація для синхронізації даних	Власна реалізація індексатора на Node.js	Потрібно розробити	Доступна для розробки, використовуються стандартні підходи

Використано набір технологій, що широко застосовуються в індустрії. Всі технології мають відкритий код, активну спільноту та навчальні матеріали,

що спрощує розробку та підтримку. Блокчейн-індексатор реалізовано власними силами з використанням стандартних підходів для моніторингу подій Smart Contract.

6.3 Аналіз ринкових можливостей запуску стартап-проєкту

В даному підрозділі стартап-проєкту в таблицях 6.4 — 6.13 демонструється аналіз ринкових можливостей.

Таблиця 6.4 — Попередня характеристика потенційного ринку стартаппроєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5-7 великих централізованих платформ, 2-3 децентралізовані рішення
2	Загальний обсяг продаж, грн/ум.од	100 млн / рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Технічні: необхідність знань блокчейн-технологій; Регуляторні: невизначеність законодавства щодо криптовалют; Економічні: необхідність інвестицій у розробку та маркетинг
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні специфічні вимоги для P2P-платформ; можливі майбутні регуляторні вимоги щодо KYC/AML для фінансових операцій

№ п/п	Показники стану ринку (найменування)	Характеристика
6	Середня норма рентабельності в галузі (або по ринку), %	15-25% для успішних маркетплейсів (вище банківського відсотка), що робить ринок привабливим

Ринок привабливий для входження: зростання попиту, висока рентабельність, обмежена конкуренція в сегменті децентралізованих рішень. Бар'єри входу помірні та можуть бути подолані завдяки технологічним компетенціям.

Таблиця 6.5 — Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару	
				До продукції	До компанії-постачальника
1	Безпечний обмін товарами та послугами без високих комісій	Активні користувачі P2P-платформ (25-45 років, середній дохід)	Шукають альтернативи з нижчими комісіями, готові використовувати криптогаманці, цінують швидкість та зручність	Швидкий пошук, зручний інтерфейс, низькі транзакційні витрати, автоматична безпека транзакцій	Надійність системи, швидка підтримка, регулярні оновлення

№ п/п	Потреба, що формує ринок	Цільова ау диторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару	
2	Децентралі зований обмін з використан ням блокчейн-технологій	Ентузіасти блокчейн-технологій (20-40 років, технічно підковані)	Активно використов ують DeFi, готові до нових технологій, цінують прозорість та захист від цензури	Прозорість репутації, захист від цензури, інтеграція з блокчейном, авт оматичне викона ння угод	Технічна підтримка, документація API, відкритість коду
3	Економічно ефективна платформа для продажу товарів	Малі та середні підприємст ва	Фокус на економічній ефективності та масштабованості, потребують інструментів для керування оголошеннями	Зменшення витр ат на комісії, інструменти для керування оголошеннями, аналітика продажів	Стабільність роботи, масштабован ість, технічна підтримка для бізнесу
4	Приватний та анонімний обмін без залежності від посередник ів	Користува чі, що цінують приватніст ь (різні вікові груп и)	Важлива анонімність та захист від цензури, не хочуть залежати від централізованих платформ	Захист від цензури, приватність даних, відсутність залежності від посередника	Прозорість п олітики платформи, захист даних користувачів

Таблиця 6.6 — Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Висока конкуренція з боку встановлених централізованих платформ	Великі платформи мають значну частку ринку та лояльність користувачів, що ускладнює залучення нових користувачів	Фокус на унікальних перевагах (нижчі комісії, безпека через блокчейн), таргетований маркетинг на технічно підкованих користувачів
2	Невизначеність регуляторного середовища щодо криптовалют	Зміни в законодавстві можуть обмежити використання блокчейн-технологій або вимагати додаткових ліцензій	Моніторинг регуляторних змін, адаптація до нових вимог, можливість роботи з фіатними валютами через партнерства
3	Технічні бар'єри для користувачів (необхідність криптогаманця)	Не всі користувачі готові встановлювати та використовувати MetaMask, що обмежує цільову аудиторію	Розробка спрощеного процесу підключення гаманця, навчальні матеріали, можливість використання без гаманця для перегляду оголошень
4	Високі витрати на Gas при роботі з блокчейном	Високі витрати на транзакції можуть зробити мікротранзакції економічно невигідними	Оптимізація Smart Contract для мінімізації Gas Cost, використання L2-рішень, батчинг транзакцій
5	Можливість появи конкурентів з аналогічними рішеннями	Великі технологічні компанії можуть розробити подібні рішення з більшими ресурсами	Акцент на швидкому впровадженні, формуванні спільноти користувачів, постійному покращенні продукту

Таблиця 6.7 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання популярності блокчейн-технологій та DeFi	Збільшення кількості користувачів, які готові використовувати криптогаманці та децентралізовані рішення	Активне позиціювання як інноваційного рішення, інтеграція з популярними DeFi-протоколами
2	Недовільність користувачів високими комісіями	Користувачі шукають альтернативи з нижчими витратами, що створює попит на децентралізовані рішення	Підкреслення економічних переваг, прозорість витрат, маркетинг спрямований на економію коштів
3	Розвиток інфраструктури блокчейн-мереж	Покращення швидкості та зниження витрат на транзакції в блокчейн-мережах робить рішення більш привабливими	Використання найбільш ефективних блокчейн-мереж, міграція на L2-рішення при необхідності
4	Зростання попиту на P2P-обмін після пандемії	Пандемія прискорила перехід до онлайн-торгівлі та збільшила попит на P2P-платформи	Позиціювання як безпечної альтернативи для обміну, акцент на безконтактних транзакціях
5	Можливість партнерств з існуючими платформами	Існуючі платформи можуть бути зацікавлені в інтеграції блокчейн-технологій для покращення безпеки	Розробка API для інтеграції, пропозиція блокчейн-рішень як сервісу для інших платформ

Таблиця 6.8 — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Монополістична конкуренція — багато продавців пропонують схожі, але диференційовані продукти	Необхідно підкреслити унікальні переваги (блокчейн-безпека, нижчі комісії), формувати бренд та спільноту користувачів
За рівнем конкурентної боротьби	Національний та міжнародний рівень — конкуренція з великими платформами, що працюють в Україні та за кордоном	Фокус на локальному ринку з можливістю масштабування, використання переваг гібридної архітектури для швидкого розвитку
За галузевою ознакою	Міжгалузева конкуренція — конкуренція між централізованими маркетплейсами, соціальними мережами з функцією продажу та децентралізованими рішеннями	Позиціювання як інноваційного рішення, що поєднує переваги різних підходів, акцент на технологічних перевагах
Конкуренція за видами товарів:	Товарно-видова конкуренція — конкуренція між різними типами P2P-маркетплейсів (централізовані vs децентралізовані vs гібридні)	Підкреслення переваг гібридної моделі, демонстрація кращого балансу між швидкістю та безпекою порівняно з конкурентами
За характером	Нецінова конкуренція — конкуренція за рахунок	Акцент на унікальних технологічних рішеннях (Smart

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
конкурентних переваг	технологічних переваг, безпеки, зручності використання, а не тільки ціни	Contracts, гібридна архітектура), покращення UX, формування репутації надійної платформи
За інтенсивністю	Марочна конкуренція — конкуренція між відомими брендами (OLX, Prom.ua) та новими технологічними рішеннями	Формування сильного бренду як інноваційної та безпечної платформи, активний маркетинг, будівництво спільноти користувачів

Конкурентне середовище характеризується монополістичною конкуренцією з акцентом на нецінові переваги. Для конкурентоспроможності важливі технологічні переваги, формування бренду та спільноти, фокус на унікальних можливостях гібридної архітектури.

Таблиця 6.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Характеристика
Прямі конкуренти в галузі	OLX, Prom.ua, Rozetka (централізовані маркетплейси), Bisq, OpenBazaar (децентралізовані P2P-платформи)
Потенційні конкуренти	Великі технологічні компанії (Google, Amazon), фінтех-стартапи з блокчейн-експертизою, існуючі маркетплейси, що інтегрують блокчейн

Складові аналізу	Характеристика
Постачальники	Провайдери блокчейн-інфраструктури (BNB Smart Chain, Ethereum), хмарні сервіси (MongoDB Atlas, AWS), розробники бібліотек (Ethers.js, React)
Клієнти	Користувачі P2P-платформ, малі та середні підприємства, ентузіасти блокчейн-технологій
Товари-замінники	Традиційні маркетплейси, соціальні мережі з функцією продажу (Facebook Marketplace), прямі угоди без посередників, офлайн-торгівля

Інтенсивність конкурентної боротьби висока. Централізовані платформи мають значну частку ринку та лояльність користувачів, але мають слабкі сторони (високі комісії, ризик цензури). Децентралізовані рішення мають технологічні переваги, але поступаються за зручністю. Гібридна модель може зайняти нішу між цими підходами. Бар'єри входження помірні.

Технічні бар'єри (необхідність блокчейн-експертизи) та економічні (інвестиції в розробку) можуть затримати вихід конкурентів на ринок на 1-2 роки. Великі компанії можуть швидше розробити подібні рішення, але вони зазвичай фокусуються на масштабних проєктах.

Сила постачальників середня. Блокчейн-мережі та хмарні сервіси мають багато альтернатив, що знижує їхню силу. Розробники бібліотек надають відкриті рішення, що зменшує залежність. Можливість міграції між різними блокчейн-мережами забезпечує гнучкість. Сила клієнтів середня. Користувачі мають багато альтернатив, що збільшує їхню силу, але унікальні переваги проєкту (безпека, нижчі комісії) можуть зменшити цю силу. Важливо формувати лояльність через якісний сервіс та технологічні переваги.

Загроза з боку замінників середня. Традиційні маркетплейси залишаються популярними через звичність, але зростаюча недовіра до високих комісій та цензури створює можливості для альтернатив. Соціальні мережі з функцією продажу можуть бути конкурентами, але вони не пропонують блокчейн-безпеку.

Принципова можливість роботи на ринку є. Конкурентна ситуація дозволяє зайняти нішу гібридних рішень. Для конкурентоспроможності проєкт має мати: технологічні переваги (гібридна архітектура, оптимізовані Smart Contracts), економічні переваги (нижчі транзакційні витрати), зручність використання (швидкий інтерфейс, простий процес підключення гаманця), надійність системи (відсутність SPOF, захист від цензури).

Щоб P2P-маркетплейс з блокчейн ескроу був конкурентоспроможним на ринку, він має мати такі характеристики та сильні сторони:

1) унікальна ціннісна пропозиція — поєднання швидкості централізованих систем з безпекою блокчейну, що відрізняє проєкт від чистих P2P-рішень та централізованих платформ, а також використання Escrow Smart Contract для автоматичного виконання угод без необхідності довіри до посередника, що є унікальною перевагою порівняно з традиційними маркетплейсами;

2) економічна ефективність — конкурентні транзакційні витрати. Орієнтація на користувачів, які шукають альтернативи з нижчими комісіями, пропонуючи оптимізовані Gas Costs порівняно з конкурентами.

3) зручний та інтуїтивний інтерфейс — забезпечити зручний Web 2.0 інтерфейс, що не вимагає глибоких технічних знань, особливо для користувачів, які вперше використовують блокчейн-технології, та забезпечити простий та зрозумілий процес інтеграції з MetaMask, що знижує бар'єри входу для нових користувачів;

4) високий рівень безпеки та надійності — децентралізація критичних операцій забезпечує захист від односторонніх рішень платформи, що важливо для користувачів, які цінують свободу торгівлі, та гібридна архітектура усуває єдину

точку відмови, забезпечуючи стійкість системи, що критично для фінансових транзакцій;

5) прозорість та довіра — блокчейн-репутація забезпечує об'єктивну оцінку користувачів, захищену від маніпуляцій, що важливо для формування довіри в P2P-обміні, та всі критичні операції фіксуються в блокчейні, що забезпечує повну прозорість та можливість аудиту;

6) гнучкість та швидка адаптація — гібридна модель дозволяє масштабувати централізований компонент, залишаючи блокчейн для критичних операцій, що забезпечує зростання без втрати продуктивності, та інтеграція з блокчейн-екосистемою. Можливість взаємодії з DeFi-протоколами та іншими блокчейн-сервісами відкриває додаткові можливості для користувачів;

7) фокус на цільову аудиторію — орієнтація на технічно підкованих користувачів та ентузіастів блокчейну, розуміння специфічних потреб цієї аудиторії та надання рішень, що відповідають цим потребам, та підтримка малого та середнього бізнесу, зменшення витрат на комісії та забезпечення прозорості транзакцій для бізнесу;

8) маркетинг та побудова спільноти — інвестування в маркетингові заходи для підвищення обізнаності про продукт та формування позитивного іміджу як інноваційного та безпечного рішення, створення активної спільноти, що сприяє розвитку платформи та збільшенню довіри;

9) відповідність технічним вимогам — забезпечення швидкого пошуку та відображення оголошень через централізований індекс, що критично для користувацького досвіду, та оптимізація Smart Contracts для зниження транзакційних витрат, що робить платформу економічно ефективною для мікротранзакцій;

10) диференціація від конкурентів — унікальні функції, пропонувати функціонал, якого немає у конкурентів, зокрема автоматичне виконання угод через Smart Contracts, P2P-комунікацію через WebRTC, та гібридну архітектуру, що поєднує переваги обох підходів.

Отже, щоб успішно конкурувати на ринку P2P-маркетплейсів, проєкт має зосередитися на сильних сторонах: гібридна архітектура, автоматична безпека через Smart Contracts, економічна ефективність, захист від цензури та прозорість репутації. Важливо постійно вдосконалювати продукт, оптимізувати Gas Costs, покращувати UX та підтримувати високий рівень безпеки, щоб виділитися серед конкурентів та зайняти стійку позицію на ринку.

Таблиця 6.10 — Обґрунтування факторів конкурентноспроможності

№ п/п	Фактор конкуренто-спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
1	Швидкість пошуку та відображення оголошень	Централізований індекс забезпечує швидкість пошуку, критичну для користувацького досвіду. Чисті P2P-рішення мають високу латентність через необхідність запитів до багатьох вузлів, що знижує їх привабливість для масового користувача
2	Транзакційні витрати (комісії)	Економічна ефективність є ключовим фактором вибору платформи. Централізовані платформи встановлюють високі комісії (10-15%), що створює попит на альтернативи з нижчими витратами. Оптимізація Gas Cost робить проєкт конкурентоспроможним
3	Безпека транзакцій та захист від шахрайства	Автоматичне забезпечення безпеки через Escrow Smart Contract усуває необхідність довіри до посередника, що є критичним для P2P-обміну. Централізовані платформи залежать від репутації оператора, що створює ризики
4	Захист від цензури та маніпуляцій	Децентралізація критичних операцій забезпечує захист від односторонніх рішень платформи, що важливо для користувачів, які цінують свободу торгівлі. Централізовані

№ п/п	Фактор конкуренто-спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
		платформи можуть блокувати користувачів або заморожувати кошти
5	Зручність інтерфейсу та простота використання	Web 2.0 UX забезпечує звичний інтерфейс, що знижує бар'єри входу для користувачів порівняно з технічними P2P-клієнтами. Простота використання є критичною для масового прийняття
6	Прозорість репутації та незмінність історії	Блокчейн-репутація забезпечує об'єктивну оцінку користувачів, захищену від маніпуляцій, що важливо для формування довіри. Централізовані системи вразливі до маніпуляцій та атак Sybil
7	Надійність системи та відсутність SPOF	Гібридна архітектура усуває єдину точку відмови, забезпечуючи стійкість системи, що критично для фінансових транзакцій. Централізовані платформи вразливі до DDoS-атак та технічних збоїв
8	Масштабованість	Гібридна модель дозволяє масштабувати централізований компонент, залишаючи блокчейн для критичних операцій, що забезпечує зростання без втрати продуктивності. Чисті P2P-рішення мають обмежену масштабованість
9	Інтеграція з блокчейн-екосистемою	Можливість використання криптогаманців та взаємодії з DeFi-протоколами відкриває додаткові можливості для користувачів, що цінують децентралізацію. Централізовані платформи не пропонують такої інтеграції

№ п/п	Фактор конкуренто-спроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)
10	Швидкість підтвердження транзакцій	Використання PoS-блокчейну забезпечує швидке підтвердження транзакцій (3-10 секунд), що важливо для користувацького досвіду порівняно з повільними PoW-мережами або централізованими системами з ручним арбітражем

Таблиця 6.11 — Порівняльний аналіз сильних та слабких сторін P2P-маркетплейсу з блокчейн ескроу

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з P2P-маркетплейсом						
			–3	–2	–1	0	+1	+2	+3
1	Швидкість пошуку та відображення оголошень	18						X	
2	Транзакційні витрати (комісії)	17						X	
3	Безпека транзакцій та захист від шахрайства	19						X	
4	Захист від цензури та маніпуляцій	18						X	
5	Зручність інтерфейсу та простота використання	16					X		
6	Прозорість репутації та незмінність історії	19						X	
7	Надійність системи та відсутність SPOF	18						X	

8	Масштабованість та здатність обслуговувати зростаючу кількість користувачів	17					X		
9	Інтеграція з блокчейн-екосистемою	15						X	
10	Швидкість підтвердження транзакцій	14				X			

Таблиця 6.12 — SWOT-аналіз стартап-проєкту

Сильні сторони:	Слабкі сторони:
– Гібридна архітектура, що поєднує швидкість централізованих систем з безпекою блокчейну; – Автоматичне забезпечення безпек и через Escrow Smart Contract; – Низькі транзакційні витрати порівняно з централізованими платформами; – Захист від цензури та маніпуляцій через децентралізацію критичних операцій; – Прозорість репутації та незмінність історії транзакцій.	– Залежність від криптогаманця (MetaMask), що створює бар'єри для нетехнічних користувачів; – Час підтвердження транзакцій (3-10 секунд) порівняно з миттєвим підтвердженням на централізованих платформах; – Необхідність технічних знань для використання блокчейн-технологій; – Високі витрати на Gas при роботі з блокчейном можуть зробити мікротранзакції економічно невигідними;
Можливості:	Загрози:
– Зростання популярності блокчейн-технологій та DeFi серед користувачів;	– Висока конкуренція з боку встановлених централізованих платформ з великою часткою ринку;

– Недовільність користувачів високими комісіями централізованих платформ; – Розвиток інфраструктури блокчейн-мереж, що знижує витрати на транзакції4. Зростання попиту на P2P-обмін після пандемії; – Можливість партнерств з існуючими платформами для інтеграції блокчейн-технологій; – Розвиток регуляторного середовища, що може сприяти легалізації блокчейн-маркетплейсів.	– Невизначеність регуляторного середовища щодо криптовалют може обмежити використання; – Технічні бар'єри для користувачів (необхідність криптогаманця) обмежують цільову аудиторію; – Можливість появи конкурентів з аналогічними рішеннями від великих технологічних компаній.

Проєкт має сильні технологічні переваги та може скористатися зростанням попиту на децентралізовані рішення. Основні ризики — технічні бар'єри для користувачів та висока конкуренція. Стратегія має фокусуватися на зниженні бар'єрів входу, формуванні спільноти та підкресленні унікальних переваг гібридної архітектури.

Таблиця 6.13 — Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Поступовий розвиток за рахунок власних коштів, мінімальний маркетинг, фокус на валідації концепції	Висока (100%) - власні ресурси	6-12 місяців
2	Залучення приватних інвесторів після демонстрації MVP, отримання капіталу для прискорення розвитку	Середня (40-60%) - залежить від якості MVP та презентації	3-6 місяців після MVP

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
3	Залучення венчурних фондів на етапі масштабування, значні інвестиції для швидкого зростання	Низька (10-20%) - високі вимоги до метрик зростання	12-18 місяців після запуску
4	Співпраця з існуючими маркетплейсами або технологічними компаніями, інтеграція блокчейн-технологій	Середня (30-50%) - залежить від переговорів та взаємної вигоди	6-9 місяців після запуску

Найбільш ймовірна та швидка альтернатива — самостійний запуск (Bootstrap) з подальшим залученням ангельських інвесторів після валідації MVP.

6.4 Розроблення ринкової стратегії проєкту

В цьому підрозділі в таблицях 6.14 — 6.17 демонструється розробка ринкової стратегії проєкту.

Таблиця 6.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Активні користувачі криптовалют	Дуже висока	Середній	Низька	Висока

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
2	IT-спеціалісти, розробники, технічні ентузіасти	Висока	Високий	Середня	Середня
3	Молоді підприємці та фрілансери	Середня-висока	Високий	Висока	Середня
4	Студенти та молодь	Середня	Високий	Дуже висока	Висока
5	Еко-свідомі споживачі	Середня	Середній	Низька-середня	Середня
6	Малі та середні підприємства	Низька-середня	Високий	Висока	Низька
7	Люди, незадоволені централізованими платформами	Висока	Високий	Середня	Середня

На основі аналізу обрано такі цільові групи для початкового етапу:

– криптоентузіасти та ранні адепти блокчейн-технологій — як первинна цільова група для швидкого старту та валідації продукту;

– технічно підковані користувачі (IT-спеціалісти) — як основна цільова група для масштабування;

– люди, незадоволені централізованими платформами — як додаткова цільова група для зростання.

Стратегія охоплення ринку — обрано стратегію диференційованого маркетингу, оскільки:

– компанія працює з кількома сегментами (криптоентузіасти, IT-спеціалісти, незадоволені користувачі);

– для криптоентузіастів — фокус на технічних перевагах блокчейну, DeFi-інтеграціях, крипто-спільнотах;

– для IT-спеціалістів — акцент на технічній документації, API, гнучкості системи, технічних форумах;Така стратегія дозволяє ефективно використовувати ресурси, зосереджуючись на найбільш готових до сприйняття сегментах, і поступово розширювати охоплення ринку.

Таблиця 6.15 — Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Самостійний запуск (Bootstrap)	Диференційований маркетинг	Гібридна архітектура, що поєднує швидкість централізованих систем з безпекою блокчейну	Створення унікального продукту з технологічними перевагами, що відрізняють його від конкурентів, з фокусом на якість, безпеку та інноваційність

Таблиця 6.16 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Шукатиме нових споживачів серед	Так, з унікальними покращеннями,	Стратегія «Інноваційного

№ п/п	Чи є проєкт «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товар у конкурента, і які?	Стратегія конкурентної поведінки*
		криптоентузіастів та IT-спеціалістів, які ще не використовують P2P-маркетплейси	копіювати зручний UI/UX централізованих платформ — але з покращенням через інтеграцію блокчейн-функцій копіювати систему репутації, копіювати швидкість обробки запитів	наслідування»: поєднання найкращих практик конкурентів з унікальними інноваційними перевагами, що дозволяє одночасно приваблювати нових користувачів та перемагати конкурентів на існуючому ринку

Таблиця 6.17 — Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1	Безпека транзакцій, прозорість репутації	Стратегія диференціації	Гібридна архітектура, Escrow Smart Contract, комісія 2%, блокчейн-репутація, відсутність SPOF	1. "Інноваційний та безпечний" 2. "Справедливий та прозорий" репутація. 3. "Швидкий та зручний"

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
2	Низькі комісії, відсутність цензури	Стратегія диференціації	Комісія 2%, децентралізація, прозора репутація, справедливий арбітраж	1. "Справедливий та економічний" 2. "Незалежний та безцензурний" 3. "Надійний та швидкий"
3	Низькі витрати, швидке отримання коштів, зручність	Стратегія диференціації	Комісія 2%, автоматичний Escrow, зручний інтерфейс, блокчейн-репутація	1. "Економічний та вигідний" 2. "Безпечний та захищений" 3. "Професійний та зручний"

6.5 Розроблення маркетингової програми стартап-проєкту

В таблицях 6.18 — 6.22 наведено розробку маркетингової програми.

Таблиця 6.18 — Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Безпека транзакцій та захист від шахрайства	Автоматичне забезпечення безпеки через Escrow Smart Contract, кошти зберігаються в смарт-контракті до підтвердження доставки	Автоматичний захист через блокчейн. Неможливість шахрайства через незмінність блокчейну

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
2	Низькі транзакційні витрати	Комісія 2% від суми угоди (найнижча на ринку)	Комісія 2% vs 5-15% у конкурентів. Економія до 13% на кожній транзакції
3	Прозорість та довіра	Блокчейн-репутація, незмінна історія транзакцій, прозора система оцінок	Незмінна репутація на блокчейні (конкуренти можуть маніпулювати). Прозора історія всіх транзакцій
4	Відсутність цензури та незалежність	Децентралізація критичних операцій, відсутність центрального контролю над транзакціями	Відсутність цензури. Незалежність від центрального органу
5	Швидкість та зручність використання	Гібридна архітектура: швидкий пошук через централізовані компоненти, безпека через блокчейн	Швидкість централізованих систем + безпека блокчейну

Таблиця 6.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за задумом	Надання безпечного, справедливого та ефективного інструменту для P2P-обміну товарами та послугами, що дозволяє економити на комісіях, забезпечує автоматичний захист транзакцій через

	блокчейн, спрощує взаємодію між користувачами та покращує довіру через прозору репутацію.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Функціональні (Тх): планування та створення угод через Escrow Smart Contract, управління оголошеннями та профілями.	X	X
	Надійності (Нм): висока стабільність роботи (99%+ uptime), відсутність SPOF через гібридну архітектуру.	X	X
	Технологічні (Тл): використання сучасних технологій, гібридна архітектура, інтеграція з блокчейн-мережами, WebRTC для P2P-комунікації.	X	X
	Ергономічні (Е): інтуїтивно зрозумілий інтерфейс користувача, адаптивний дизайн для різних пристроїв, зручна навігація та пошук оголошень	X	X
	Естетичні (М): сучасний та привабливий дизайн, зручна навігація та розташування елементів, відповідність останнім тенденціям UI/UX дизайну.	X	X
	Безпеки (Б): використання захищених протоколів передачі даних, аутентифікація та авторизація користувачів за допомогою JWT, Escrow Smart Contract, відповідність	X	X

	стандартам безпеки веб-додатків, захист приватних ключів користувачів.		
	Якість: Дотримання галузевих стандартів розробки ПЗ. Ретельне тестування функціональності, навантаження та безпеки. Аудит смарт-контрактів. Відповідність стандартам блокчейн-розробки (Ethereum/Solidity best practices). Постійне оновлення та вдосконалення функціоналу. Моніторинг продуктивності та доступності.		
	Пакування: Продукт поставляється як веб-додаток (SaaS), доступний через веб-браузер. Немає потреби в фізичному пакуванні. Документація API, інструкції для користувачів, технічна підтримка.		
	Марка: Назва організації-розробника: P2P Marketplace. Назва товару: Blockchain Escrow Platform.		
III. Товар із підкріпленням	До продажу: Безкоштовна реєстрація та створення облікового запису. Онлайн-демонстрації та вебінари. Доступ до документації та навчальних матеріалів. Безкоштовна консультація щодо використання платформи. Маркетингова підтримка (SEO, соціальні мережі).		
	Після продажу: Технічна підтримка користувачів через email та чат. Регулярні оновлення та додавання нового функціоналу. Моніторинг безпеки та швидке вирішення проблем. Система зворотного зв'язку. Спільнота користувачів (форум, соціальні мережі). Документація та FAQ. Навчальні вебінари.		
	За рахунок чого потенційний товар буде захищено від копіювання: Власний захищений програмний код. Унікальна гібридна архітектура як технологічне рішення. Реєстрація торгової марки та патентування унікальних рішень. Укладення договорів про нерозголошення з працівниками та партнерами. Дотримання		

	ліцензійних угод. Ексклюзивні партнерства з блокчейн-мережами. Бренд та репутація як нематеріальні активи. Швидкість інновацій та постійне вдосконалення продукту.

Таблиця 6.20 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Безкоштовні платформи: комісія 0%. Децентралізовані маркетплейси: комісія 0-1%	Централізовані P2P-платформи: eBay (10-12%), Amazon Marketplace (8-15%), Etsy (6.5%), Airbnb (3% + 10-15%), Uber (20-25%). Середня комісія: 8-15%	Криптоентузіасти та IT-спеціалісти: 30 000-80 000 грн/міс. Молоді підприємці: 20 000-50 000 грн/міс. Готовність платити: до 3% від транзакції	Нижня межа: 1.5% від суми транзакції. Верхня межа: 3% від суми транзакції. Рекомендована ціна: 2% від суми транзакції + преміум-підписка 500 грн/міс

Таблиця 6.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Самостійний пошук платформи через інтернет, дослідження функціоналу пере	Маркетинг та реклама (SEO, соціальні мережі, контент-маркетинг), технічна підтримка	Прямий канал (0 рівнів): Виробник → Користувач. Веб-платформа	Власна система збуту через веб-платформу (SaaS модель). Безпосередній збут

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	д реєстрацією, перевага безкоштовному тестуванню, потребують технічної підтримки під час першого використання	користувачів, навчальні матеріали та демонстрації, забезпечення безпеки та надійності платформи, постійне вдосконалення функціоналу	дозволяє безпосередній доступ користувачів без посередників	через офіційний веб-сайт, без залучення сторонніх посередників. Маркетингові партнерства з крипто-спільнотами та IT-спільнотами для просування

Таблиця 6.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Криптоентузіасти та IT-спеціалісти: активні в технічних спільнотах, шукають інноваційні рішення, аналізують	Технічні форуми (Reddit, Stack Overflow, GitHub), крипто-спільноти (Telegram, Discord), професійні соціальні мережі	"Інноваційний та безпечний" — блокчейн-технології, автоматична безпека чер	Привернути увагу технічно підкованих користувачів, продемонструвати технічні переваги, сформувати довіру через	"Перший гібридний P2P-маркетплейс, що поєднує швидкість централізованих систем з безпекою бло

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	технічні характеристики перед вибором, довіряють рекомендаціям спільноти	(LinkedIn), технічні блоги та YouTube-канали, конференції та мітапи	ез Smart Contracts	технічну експертизу	кчейну. Автоматичний захист через Escrow Smart Contract. Спробуйте безкоштовно."
2	Незадоволені користувачі централізованих платформ: шукають альтернативи через соціальні мережі, читають відгуки, порівнюють комісії та умови, потребують доказів переваг	Соціальні мережі (Facebook, Instagram, Twitter), пошукові системи (Google), форуми та спільноти користувачів, email-розсилки, контент-маркетинг (статті, відео)	"Справедливий та економічний" — найнижчі комісії (2%), відсутність цензури, справедливий арбітраж	Переконати користувачів у перевагах перед конкурентами, показати економічну вигоду, сформувати мотивацію для переходу	"Комісія 2% замість 10-15%. Без цензури. Справедливий арбітраж. Переходьте на платформу нового покоління. Економте на кожній транзакції."
3	Молоді підприємці та	Бізнес-спільноти (Facebook	"Економічний та	Показати економічну	"Знижте витрати на

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	фрілансери: активні в бізнес-спільнотах, шукають ефективні інструменти для бізнесу, цінують швидкість та зручність, потребують доказів надійності	Groups, LinkedIn), бізнес-блоги та вебінари, email-маркетинг, партнерські програми, реферальні програми	вигідний" — низькі комісії, швидкі платежі, зручний інтерфейс	вигоду для бізнесу, продемонструвати зручність використання, сформувати довіру до платформи	комісії до 2%. Швидкі платежі. Зручний інтерфейс для вашого бізнесу. Реєструйтеся та отримуйте перші 10 транзакцій без комісії."

Стратегія комунікацій базується на диференційованому підході до різних цільових груп з використанням відповідних каналів та повідомлень. Для технічно підкованих користувачів — акцент на інноваційності та технічних перевагах через технічні канали. Для незадоволених користувачів — акцент на економічній вигоді та справедливості через соціальні мережі. Для бізнесу — акцент на ефективності та зручності через бізнес-канали.

Висновки до розділу 6

Проведений аналіз показує, що проєкт має можливість ринкової комерціалізації. Попит підтверджується зростанням популярності блокчейн-технологій, недовільністю високими комісіями централізованих платформ (5-15%)

та зростанням попиту на P2P-обмін після пандемії. Ринок децентралізованих маркетплейсів демонструє позитивну динаміку, а гібридна архітектура проєкту поєднує переваги централізованих та децентралізованих систем, що створює унікальну цінність для користувачів. Рентабельність роботи на ринку забезпечується рекомендованою комісією 2%, що є значно нижчою за конкурентів, але достатньою для забезпечення прибутковості. Прогноз показує досягнення беззбитковості на 14-15 місяці, а на другому році очікується рентабельність 25-30% завдяки зростанню користувачів та обсягу транзакцій.

Перспективи впровадження проєкту є позитивними з огляду на потенційні групи клієнтів, які визначені та демонструють високу готовність сприйняти продукт. Криптоентузіасти та ранні адепти блокчейн-технологій мають готовність 90-95%, технічно підковані користувачі (IT-спеціалісти) — 70-80%, а незадоволені користувачі централізованих платформ — 75-85%. Бар'єри входження на ринок існують, зокрема технічні бар'єри через необхідність використання криптогаманця, що обмежує цільову аудиторію, але обрані цільові групи є технічно підкованими та готовими до використання блокчейн-технологій. Стан конкуренції є високим з боку встановлених централізованих платформ, але унікальні переваги проєкту, такі як гібридна архітектура, низькі комісії (2%), автоматична безпека через Escrow Smart Contract, прозора блокчейн-репутація та відсутність цензури, створюють стійку конкурентну перевагу, що дозволяє конкурувати на ринку.

Подальша імплементація проєкту є доцільною, оскільки проєкт має всі необхідні передумови для успішного впровадження. Технологічна база проєкту (гібридна архітектура, Smart Contracts, блокчейн-інтеграція) є інноваційною та конкурентоспроможною. Ринкова позиція проєкту чітко визначена та відповідає потребам цільових сегментів. Цільові сегменти визначені та демонструють високу готовність сприйняття продукту.

ВИСНОВКИ

У ході виконання роботи досягнуто поставленої мети — розроблення та реалізація інформаційної системи обміну товарами та послугами на основі гібридної P2P-архітектури з інтеграцією блокчейн-технологій для забезпечення безпеки транзакцій та усунення системних ризиків централізованих платформ. Розроблене рішення забезпечує безпечний та ефективний P2P-обмін через гібридну архітектуру, що поєднує швидкість централізованих систем з безпекою та прозорістю блокчейну, автоматичний захист транзакцій через Escrow Smart Contract, низькі комісії (2%) та прозору блокчейн-репутацію.

Перший розділ містить теоретичні засади децентралізованих систем та P2P-економіки. Розглянуто проблематику централізованих P2P-маркетплейсів, концептуальні основи блокчейн-технологій, механізми консенсусу (PoS/PoW), поняття Trustless-систем та їх роль у P2P-обміні, а також механізми Smart Contracts для автоматизації угод (Escrow-механізм). Це сформувало розуміння ролі децентралізації та блокчейну в P2P-обміні.

Другий розділ присвячено огляду існуючих рішень. Проведено аналіз чистої P2P-архітектури (Bisq, OpenBazaar) та гібридних блокчейн-рішень, порівняння за критеріями Gas Cost, Latency та надійності. Аналіз показав, що існуючі рішення мають обмеження: чисті P2P-системи — технічні бар'єри та повільність, централізовані — високі комісії та цензура. Це обґрунтувало вибір гібридної моделі, що поєднує переваги обох підходів.

Третій розділ описує функціональні етапи проєктування та моделювання системи. Проведено декомпозицію функціональних вимог, специфікацію ключових модулів (фронт-енд, бек-енд, блокчейн-інтерфейс), розробку та обґрунтування гібридної архітектурної моделі, моделювання функціональних процесів (Use Case, Activity, Sequence діаграми). Забезпечено структурування системи та визначено інтерфейси взаємодії між компонентами.

Четвертий розділ присвячено нефункціональним етапам реалізації та експериментальній оцінці. Визначено вимоги до безпеки, надійності та

масштабованості, реалізовано ключовий елемент — Escrow Smart Contract на платформі BNB Smart Chain, розроблено методику імітаційного моделювання та вимірювання метрик Gas Cost та Latency. Результати підтвердили ефективність гібридної архітектури та досягнення поставлених нефункціональних вимог.

П'ятий розділ продемонстрував розробку P2P-маркетплейсу. Описано розроблення серверної частини (Node.js, Express, MongoDB, Socket.io) та клієнтської частини (React, Redux, Ethers.js, WebRTC). Проілюстровано архітектуру системи, ключові компоненти та їх взаємодію, що підтвердило реалізацію гібридної моделі та забезпечення функціональності платформи.

Шостий розділ містить маркетингове дослідження стартап-проєкту, за результатами якого проєкт виявився перспективним для ринкової реалізації. Визначено цільові сегменти (криптоентузіасти, IT-спеціалісти, незадоволені користувачі), розроблено стратегію позиціювання ("Інноваційний, справедливий та швидкий P2P-маркетплейс нового покоління"), визначено оптимальну систему збуту та концепцію маркетингових комунікацій. Фінансовий аналіз показав можливість досягнення беззбитковості на 14-15 місяці та рентабельність 25-30% на другому році. Проєкт має передумови для успішного виходу на ринок та створення конкурентних переваг.

Таким чином, поставлені завдання виконано, мета роботи досягнута. Отримані результати можуть бути корисними для організацій та користувачів, які прагнуть безпечного та ефективного P2P-обміну товарами та послугами з використанням блокчейн-технологій. Перспективи подальшого розвитку системи включають розширення функціональності (додаткові блокчейн-мережі, DeFi-інтеграції), впровадження методів штучного інтелекту для оптимізації рекомендацій та виявлення шахрайства, розробку мобільних застосунків та покращення користувацького досвіду, що сприятиме подальшому підвищенню ефективності та популярності платформи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 25.11.2025).

2. Salek, M., Shayandeh, S., Kempe, D. You Share, I Share: Network Effects and Economic Incentives in P2P File-Sharing Systems. URL: https://arxiv.org/abs/1107.5559 (дата звернення: 26.11.2025).

3. Buterin, V. Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform. URL: https://ethereum.org/en/whitepaper/ (дата звернення: 27.11.2025).

4. Zheng, P., Zheng, Z., Chen, L. Selecting Reliable Blockchain Peers via Hybrid Blockchain Reliability Prediction. URL: https://arxiv.org/abs/1910.14614 (дата звернення: 28.11.2025).

5. Antonopoulos, A. M. Mastering Bitcoin: Programming the Open Blockchain. URL: https://github.com/bitcoinbook/bitcoinbook (дата звернення: 29.11.2025).

6. Park, J., van der Schaar, M. A Game Theoretic Analysis of Incentives in Content Production and Sharing over Peer-to-Peer Networks. URL: https://arxiv.org/abs/0910.4618 (дата звернення: 30.11.2025).

7. Antonopoulos, A. M., Wood, G. Mastering Ethereum: Building Smart Contracts and DApps. URL: https://github.com/ethereumbook/ethereumbook (дата звернення: 01.12.2025).

8. Futoransky, A., Sarraute, C., Fernandez, D., Travizano, M., Waissbein, A. Fair and Decentralized Exchange of Digital Goods. URL: https://arxiv.org/abs/2002.09689 (дата звернення: 02.12.2025).

9. Khorasany, M., Dorri, A., Razzaghi, R., Jurdak, R. Lightweight Blockchain Framework for Location-aware Peer-to-Peer Energy Trading. URL: https://arxiv.org/abs/2005.14520 (дата звернення: 03.12.2025).

10. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. URL: https://ethereum.github.io/yellowpaper/paper.pdf (дата звернення: 04.12.2025).

11. Ethereum Developer Documentation. URL: https://ethereum.org/en/developers/docs/ (дата звернення: 05.12.2025).

12. ConsenSys. Smart Contract Best Practices. URL: https://consensys.github.io/smart-contract-best-practices/ (дата звернення: 06.12.2025).

13. Solidity Documentation. URL: https://docs.soliditylang.org/ (дата звернення: 07.12.2025).

14. Khorasany, M., Mishra, Y., Ledwich, G. Design of auction-based approach for market clearing in peer-to-peer market platform. URL: https://arxiv.org/abs/1902.09277 (дата звернення: 08.12.2025).

15. Gopalan, A., Sankararaman, A., Walid, A., Vishwanath, S. Stability and Scalability of Blockchain Systems. URL: https://arxiv.org/abs/2002.02567 (дата звернення: 07.12.2025).

16. React Documentation. URL: https://react.dev/ (дата звернення: 08.12.2025).