Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: **«Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн»**

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**.
(підпис) (прізвище та ініціали)

Виконав:
студент групи КУ– 61
Галузь знань: 15 – Автоматизація та приладобудування
спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»
МЕДВЕДЕНКО Владислав
Олексійович _____

Керівник: к.т.н., доцент
БУЛАВІН Дмитро
Олексійович_____

Рецензент: д.т.н., доц., професор кафедри теоретичної і прикладної інформатики
РУККАС Кирило Маркович_____

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, чотирьох розділів, висновку, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 81 сторінку, із яких 54 сторінки основної частини з 28 рисунками, 4 таблицями, 1 найменуваннями списку використаних джерел та чотирма додатками.

Метою дослідження є підвищення рівня безпеки та досягнення прозорості транзакцій цифрової валюти в комп'ютерній моделі соціальної мережі, застосовуючи технологію блокчейн.

Об’єкт дослідження – процес підвищення рівня безпеки та досягнення прозорості цифрової валюти в соціальній мережі.

Методи дослідження: моделювання, розробка, вдосконалення та аналіз комп'ютерної моделі соціальної мережі, використовуючи концепцію web 3.0 та технологію блокчейн. Визначення архітектурних особливостей та механізмів взаємодії.

Предмет – моделі та методи застосування технології блокчейн для розв’язання задачі підвищення рівня інформаційної безпеки та прозорості транзакцій цифрової валюти.

Ключові слова: web 3.0, блокчейн , архітектура, розробка, комп’ютерна модель, безпека, прозорість, цифрова валюта, соціальна мережа.

ABSTRACT

The explanatory note to the master's thesis consists of an introduction, four chapters, a conclusion, a list of used sources and four appendices. The total volume of the work is 81 pages, of which 54 pages are the main part with 28 figures, 4 tables, 1 names of the list of used sources and four appendices.

The purpose of the study is to increase the level of security and achieve transparency of digital currency transactions in a computer model of a social network, using blockchain technology.

The object of the study is the process of increasing the level of security and achieving transparency of digital currency in the social network.

Research methods: simulation, development, improvement and analysis of a computer model of a social network using the concept of web 3.0 and blockchain technology. Definition of architectural features and interaction mechanisms.

The subject is models and methods of applying blockchain technology to solve the problem of increasing the level of information security and transparency of digital currency transactions.

Keywords: web 3.0, blockchain, architecture, development, computer model, security, transparency, digital currency, social network.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

КМ - комп’ютерна модель;

WEB - World Wide Web;

СКБД - Система керування базами даних;

ASCII - American Standard Code for Information Interchange;

BPMN - Business Process Model Notation;

DOM - Document Object Model;

ID - Identify document;

IDE - Integrated Development Environment;

JSON - JavaScript Object Notation;

SQL - Structured Query Language;

UML - Unified Modeling Language;

ВСТУП

Соціальні мережі та месенджери стають активною запорукою людського життя, тому їх актуальність на сьогоднішній день неможливо не переоцінити. Кожен день люди створюють публікації, активно спілкуються, та проводять вільний час у соціальних мережах. Так як кожного дня технології швидко розвиваються, покращуються та поширюються, є потреба у спробах застосування їх у різних сферах людського життя. Тому я вирішив реалізувати комп'ютерну модель соціальної мережі з використанням концепції web 3.0, та технології блокчейн.

Актуальність роботи.

Актуальність роботи полягає в тому, що соціальні мережі є невідкладною частиною сучасного життя людей, а тому їх реалізація на новітніх технологіях і концепціях є поштовхом до розвитку та впровадження нових функцій у вже існуючих, та майбутніх соціальних мережах та месенджерах.

Сучасні соціальні мережі стають все більш однотипними, і їхня еволюція може бути прискорена за допомогою новаторських технологій та концепцій.

Зростання інтересу до блокчейн технології. Блокчейн не тільки революціонізує фінансовий сектор, але і має потенціал трансформувати різні галузі, включаючи соціальні мережі. Використання цієї технології може привертати увагу дослідників та індустрії. З плином часу централізовані соціальні мережі стикаються з проблемами конфіденційності даних, цензури та недостатньої безпеки.

Тренд web 3.0. Концепція web 3.0 стає все більше актуальною, оскільки сучасний інтернет виходить за межі централізованого контролю та переходить до децентралізації, блокчейн технології та інших інновацій.

Моя робота це комп'ютерна модель соціальної мережі з використанням нової концепції Web 3.0 та технології блокчейн. В комп'ютерній моделі соціальної мережі користувачі зможуть спілкуватися у чатах, ділитися

думками в своїх блогах. Буде можливість оформити підписку на платні повідомлення, тобто автор буде отримувати умовну валюту, за свої публцікації. А користувачі матимуть змогу отримати цю валюту. Всі операції з умовною валютою будуть відбуватися за допомогою технології блокчейн.

Метою дослідження є підвищення рівня безпеки та досягнення прозорості транзакцій цифрової валюти в комп'ютерній моделі соціальної мережі, застосовуючи технологію блокчейн.

Об’єкт дослідження – процес підвищення рівня безпеки та досягнення прозорості цифрової валюти в соціальній мережі.

Методи дослідження: моделювання, розробка, вдосконалення та аналіз комп'ютерної моделі соціальної мережі, використовуючи концепцію web 3.0 та технологію блокчейн. Визначення архітектурних особливостей та механізмів взаємодії.

Предмет – моделі та методи застосування технології блокчейн для розв’язання задачі підвищення рівня інформаційної безпеки та прозорості транзакцій цифрової валюти.

Завдання дослідження

1. Виконати аналіз технології блокчейн та сфер її застосування.
2. Виконати аналіз концепції web 3.0 та сфер її застосування.
3. Розробка та оптимізація архітектури моделі, використовуючи концепцію web 3.0 та блокчейн.
4. Визначення та реалізація механізмів, технології блокчейн, для створення транзакцій(обміном) умовної валюти.
5. Розробка нових методів та інтерфейсів для поліпшення взаємодії між користувачами на основі концепції Web 3.0.
6. Реалізація розробленої моделі та проведення тестів для оцінки її функціональності.
7. Аналіз отриманих результатів тестування.
8. За потреби, проведення поліпшення моделі, для її вдосконалення.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ І КОНЦЕПЦІЙ ДЛЯ РОЗРОБКИ МОДЕЛІ

1.1 Концепція Web

1.1.1 Web 1.0

Веб 1.0 — це термін, який використовується для позначення початкової фази Інтернету, яка виникла з агентства передових оборонних дослідницьких проектів (DARPA) і вперше стала глобальною мережею, яка представляла майбутнє цифрових комунікацій.

Під час цього етапу інтернет в основному використовувався для інформаційних цілей, зі статичними веб-сайтами, які відображали інформацію в односпрямованому вигляді. Користувачі могли лише переглядати вміст і не могли з ним взаємодіяти.

Цей період, часто називають "Read-Only" веб, він тривав з початку 1990-х до початку 2000-х. Ось деякі ключові характеристики Веб 1.0:

- Статичний контент: Веб-сторінки в епоху Веб 1.0 були в основному статичними, що означає, що контент був фіксований і не змінювався динамічно в залежності від дій користувача. Більшість веб-сайтів представляли собою прості HTML-сторінки з текстом та зображеннями.

- Обмежена інтерактивність: Взаємодія на Веб 1.0 була мінімальною. Користувачі могли клікати на гіперпосилання для переходу між сторінками, але за межами цього взаємодія була обмеженою. Веб-сайти більше нагадували цифрові брошури або документи.

- Одностороннє спілкування: Веб був в основному каналом одностороннього спілкування, де інформацію надавав власник чи видавець веб-сайту, а користувачі споживали цю інформацію. Було обмежено внесенням користувачів або взаємодією між ними.

- Централізоване створення контенту: Створення контенту було централізованим, і невелика кількість осіб чи організацій відповідали за створення та публікацію інформації в мережі. Більшість користувачів були споживачами контенту, а не учасниками.

- Прості технології: Технології, які використовувались в Веб 1.0, були відносно простими порівняно із сучасними стандартами. Основною мовою розмітки був HTML (Мова розмітки гіпертексту), і дизайн часто був простим, з мінімальним використанням графіки чи мультимедійних елементів.

- Зародження електронної комерції: Коли завершувалася епоха Веб 1.0, розпочалося зародження електронної комерції. Компанії почали досліджувати онлайн-продажі, але досвід часто обмежувався базовою інформацією про продукти та статичними формами замовлення.

- Популяризація вебу: Незважаючи на його обмеження, Веб 1.0 відіграв важливу роль у популяризації використання інтернету. Багато перших прихильників почали створювати особисті та бізнес-сайти, вкладаючи фундамент для експансії вебу в наступні роки.

- Приклади сайтів Веб 1.0: Деякі іконічні приклади сайтів Веб 1.0 включають ранні версії Yahoo!, GeoCities та перші ітерації онлайн-новин.

Веб 1.0 заклав фундамент для подальших розвитків у еволюції вебу, що призвело до більш динамічних та інтерактивних вражень, які ми бачимо у Веб 2.0 та пізніших етапах.[1]

Це був ключовий етап, який визначив трансформацію інтернету у більш динамічну платформу.

1.1.2 Web 2.0

Появу терміну «web 2.0» прийнято пов'язувати із статтею Tim O'Reilly - What Is Web 2.0. У цій статті автор наводить порівняння сайтів веб 1.0 та аналогічних їм прикладів веб 2.0. Розглядаючи докладно це порівняння, Тім О'Рейлі виділяє основні принципи Інтернету 2.0, сформулювавши основний принцип інтернет як платформа.

Веб 2.0 вказує на друге покоління Всесвітньої павутини, що характеризується переходом від статичних веб-сторінок до динамічного та інтерактивного контенту, створеного користувачами.

Основні характеристики:

- Контент, створений користувачами: на відміну від Веб 1.0, де контент переважно створювався власниками веб-сайтів, Веб 2.0 сприяє контенту, створеному самими користувачами. Стала активною участь у соціальних мережах, блогах і вікі.

- Інтерактивність: збільшена інтерактивність через функції, такі як коментарі, лайки, поділ і спільне редагування.

- Збагачений користувацький досвід*:* Увага перейшла на створення більш насиченого та динамічного користувацького досвіду за допомогою AJAX (асинхронний JavaScript та XML) для більш плавних та швидших взаємодій.

- Співпраця: З'явилися інструменти та платформи співпраці, що дозволяють користувачам спільно створювати та редагувати вміст в реальному часі.

Визначні технології та платформи:

Соціальні мережі: Платформи, такі як Facebook, Twitter і LinkedIn, трансформували спосіб з'єднання людей та обміну інформацією.

Блоги та платформи блогів: Служби, такі як WordPress і Blogger, полегшили індивідам публікацію контенту без значних технічних знань.

Вікі: Wikipedia, колективна енциклопедія, є прикладом концепції вікі, де користувачі вносять свій внесок і редагують контент спільно.

Багатофункціональні Інтернет-додатки (RIAs): З'явилися додатки з покращеними інтерфейсами та інтерактивністю, часто розроблені за допомогою технологій, таких як Flash, а пізніше HTML5.

Бізнес-наслідки:

- Залучення користувачів: Бізнес використовував Веб 2.0 для взаємодії з користувачами напряму, будуючи спільноти навколо своїх продуктів чи послуг.

- Соціальний маркетинг: Зародження концепції соціального маркетингу, коли бізнеси прагнули досягти аудиторії через соціальні платформи.

- Гнучкий розвиток: Ітеративний та співпрацюючий характер Веб 2.0 вплинув на методології розробки, викликаючи появу практик гнучкого розвитку.

Виклики та загрози:

- Проблеми конфіденційності: Збільшення обміну особистою інформацією породжувало питання щодо конфіденційності та безпеки даних.

- Перевантаження інформацією: Завдяки росту контенту, створеного користувачами, іноді виникали проблеми з фільтрацією релевантної інформації.

Приклади сервісів Web 2.0:

Соціальні мережі: Facebook, Twitter, Instagram. Платформи блогів: WordPress, Blogger. Інструменти співпраці: Google Docs, Dropbox. Вікі: Wikipedia. Багатофункціональні Інтернет-додатки: Gmail, Google Maps.

Web 2.0 відзначив перехідний етап у еволюції вебу, зосереджений на співпраці, участі користувачів та більш динамічному онлайн-досвіді. Він поклав основу для сучасного соціального та інтерактивного інтернету.[2]

1.1.3 Web 3.0

Поняття «Web 3.0» одне із тих, які кожен трактує собі по-різному. Так, згідно з визначенням Джейсона Калаканіса, «Web 3.0 – високоякісний контент та сервіси, які створюються талановитими професіоналами на технологічній платформі Web 2.0»[3]. Інше визначення ідентифікує Web 3.0 як «Семантичне павутиння» (Semantic Web).

Головна думка Semantic Web базується на впровадженні метамови, яка описує зміст сайтів для організації автоматичного обміну між серверами.

Інакше кажучи, ці прикріплені метадані повинні бути зрозумілі системою, піддані машинній обробці, логічно співвіднесені один з одним, проаналізовані і включені в результати запиту. Його мета - зробити інформацію більш взаємопов'язаною та зрозумілою для машин.[4]

Більшість проектів які позиціонують себе як проект з використанням концепцій Web 3.0 працюють на блокчейн технологіях.

Основні характеристики:

- Семантичне розуміння: web 3.0 акцентує на наданні значення та контексту інформації, що дозволяє машинам розуміти та інтерпретувати дані більш розумно.

- Децентралізація: на відміну від web 2.0, який часто ґрунтується на централізованих платформах, web 3.0 досліджує децентралізовані технології, такі як блокчейн, для збільшення безпеки, конфіденційності та контролю.

- Персоналізація: покращена персоналізація завдяки штучному інтелекту та машинному навчанню, що налаштовує контент та послуги під індивідуальні вподобання користувачів.

- Інтероперабельність: Збільшена взаємодія між різними платформами та сервісами, що дозволяє безперешкодний обмін даними.

Технології та концепції:

- Блокчейн: web 3.0 часто використовує технологію блокчейн для децентралізованого та безпечного зберігання даних і проведення транзакцій.

- Штучний Інтелект (ШІ): ШІ відіграє ключову роль у web 3.0, дозволяючи машинам аналізувати, вивчати та розуміти поведінку користувачів для створення більш персоналізованих вражень.

- З'єднані Дані: концепція з'єднання даних з різних джерел для створення більш об'єднаної та взаємопов'язаної мережі інформації.

- Розумні Контракти: самовиконуючі контракти з умовами угод, написаними безпосередньо у коді, часто використовуються в децентралізованих додатках (DApps) на блокчейн.

Сценарії застосування:

- Децентралізовані додатки (DApps): web 3.0 сприяє розвитку децентралізованих додатків, що працюють на мережах блокчейн, гарантуючи прозорість та довіру.

- Семантичний Пошук: більш розумні пошукові системи, які розуміють наміри та контекст користувача, надаючи більш точні та актуальні результати.

- Персоналізовані Рекомендації: рекомендації, що працюють на базі ШІ, які передбачають потреби користувачів на основі їх вподобань та поведінки.

- Токенізація Активів: активи, включаючи нерухомість та мистецтво, можуть бути представлені у вигляді цифрових токенів на блокчейні, дозволяючи дробову власність та легший перехід.

Виклики та розгляди:

- Конфіденційність даних: якщо більше даних стає взаємопов'язаним, забезпечення конфіденційності користувача та безпеки даних стає критичним викликом.

- Стандарти та інтероперабельність: встановлення стандартів для взаємодії між різними платформами та технологіями є важливим для успіху web 3.0.

- Перешкоди прийняття: перехід до web 3.0 може зіткнутися із викликами, пов'язаними із прийняттям нових технологій та інтеграцією децентралізованих систем.

Розвиток від web 2.0: в той час як web 2.0 акцентувався на створенні контенту користувачами та співпраці, web 3.0 покладає наголос на розуміння машин, децентралізацію та інтелектуальну обробку даних.

Перспективи майбутнього: бачення web 3.0 включає більш розумний, децентралізований та спрямований на користувача інтернет-досвід. З розвитком технологій, таких як блокчейн та штучний інтелект, реалізація концепцій web 3.0 стає більш можливою.[5]

Web 3.0 уявляє собою інтернет, де інформація не тільки представлена, але і зрозуміла, створюючи більш персоналізоване та безпечне цифрове

середовище. Його розвиток передбачає спільну роботу різних технологічних галузей для створення більш розумного та взаємопов'язаного онлайн-досвіду.[6]

Таблиця 1.1

Порівняння WEB 1.0, WEB 2.0 та WEB 3.0

	WEB 1.0	WEB 2.0	WEB 3.0
Зміст	Пасивна взаємодія для користувача	Платформи спільноти та контент користувачів	Право власності для творців контенту
Технології	HTML	Динамічний HTML, Javascript	Блокчейн, штучний інтелект, машинне навчання
Віртуальні середовища	Немає	Деяке базове використання 3D	3D, VR, AR
Реклама	Нав'язлива (банери і т.д.)	Інтерактивна	Таргетинг на основі поведінки користувача
Зберігання даних	Зберігання на серверах окремих вебсайтів	Належить великим технологічним гігантам	Розподіллено серед користувачів
Аудиторія	Індивідуальні користувачі	Конкретні спільноти користувачів	Взаємопов'язані користувачі на різних платформах та пристроях[7]

1.2 Технологія блокчейн

Блокчейн або децентралізований цифровий реєстр – це особливий вид бази даних, який підтримується численними комп'ютерами, розміщеними по

всьому світу. Дані блокчейну організовані в блоки, які розташовані в хронологічному порядку і захищені криптографією.

Перша модель блокчейну була створена на початку 1990 років, коли вчений-комп'ютерних Стюарт Хабер і фізик В. Скотт Сторнетта використовували криптографічні методи в ланцюгу блоків для захисту цифрових документів від підробки даних.

Хабер і Сторнетта надихнули на роботу багатьох інших вчених і криптоентузіастів, що зрештою призвело до створення першої криптовалюти, заснованої на блокчейн-технології - Bitcoin. З того часу прийняття блокчейн-технології поступово розширювалося, і криптовалюти використовуються все більшою кількістю людей у всьому світі.

Хоча блокчейн-технологія часто використовується для запису криптовалютних транзакцій, вона підходить для запису багатьох інших типів цифрових даних і може застосовуватися в різних випадках.[8]

1.2.1 Децентралізація у блокчейн.

Децентралізація у блокчейні – це ідея про те, що контроль і повноваження щодо прийняття рішень у мережі розподіляються між її користувачами, а не контролюються однією особою, такою як уряд чи корпорація. Це може бути корисним у ситуаціях, коли людям необхідно координувати свої дії з незнайомцями або коли вони хочуть подбати про безпеку й цілісність своїх даних.

У децентралізованій блокчейн-мережі немає центрального органу чи посередника, який контролює потік даних чи транзакцій. Натомість транзакції перевіряються й записуються розподіленою мережею комп'ютерів, які працюють разом для підтримки цілісності мережі.

Коли люди говорять про блокчейн-технологію, вони часто мають на увазі не лише базу даних. Блокчейн-технологія підтримує криптовалюти й невзаємозамінні токени (NFT), дозволяючи людям співпрацювати й здійснювати угоди один з одним, не покладаючись на центральний орган.

1.2.2 Загальний алгоритм роботи блокчейн

За своєю суттю блокчейн є цифровим реєстром, який надійно записує транзакції між двома сторонами із захистом від несанкціонованого доступу. Ці дані транзакцій записуються глобальною розподіленою мережею спеціальних комп'ютерів, які називаються нодами.

Коли користувач ініціює транзакцію, наприклад, переказує певну кількість криптовалюти іншому користувачеві, ця транзакція транслюється в мережу. Кожна нода проводить автентифікацію транзакції, перевіряючи цифрові підписи й інші дані транзакції.

Щойно транзакцію перевірено, вона додається до блоку разом з іншими вже перевіреними транзакціями. Блоки поєднуються в ланцюг з використанням криптографічних методів, утворюючи блокчейн. Процес перевірки транзакцій і додавання їх у блокчейн здійснюється за допомогою механізму консенсусу – набору правил, які визначають, як ноди в мережі досягають згоди про стан блокчейну і дійсності транзакцій.

Криптографія є ключем до блокчейну для підтримки безпечного, прозорого й захищеного від несанкціонованого доступу запису транзакцій. Наприклад, хешування – це найважливіший криптографічний метод, який використовується в блокчейнах. Це криптографічний процес, який перетворює вхідні дані будь-якого розміру у рядок символів фіксованого розміру.

Хеш-функції, що використовуються в блокчейнах, зазвичай стійкі до колізій, а це означає, що шанси знайти два фрагменти даних, що дають однаковий результат, астрономічно малі. Інша особливість називається лавинним ефектом, що означає явище, коли будь-яка незначна зміна вхідних даних призведе до іншого результату.

Давайте проілюструємо це за допомогою SHA256 – функції, що широко використовується у Bitcoin. Зверніть увагу у табл. 1.1, зміна великих літер призвела до різкої зміни результату. Хеш-функції є також односторонніми функціями, тому що з обчислювальної точки зору неможливо отримати вхідні дані шляхом зворотного проєктування хешу результату.

Таблиця 1.2

Результати хешування

Вхідні дані	Результат SHA256
Vlad Medvedenko	886c5fd21b403a139d24f2ea3669ff5c0df42d5f873a56d04dc480808c155af3
Vlad medvedenko	4733a0602ade574551bf6d544d94e091d571dc2fcfd8e39767d38301d2c459a7
vlad medvedenko	a780cd8a625deb767e444c6bec34bc86e883acc3cf8b7971138f5b25682ab181

Кожен блок у блокчейні містить хеш попереднього блоку, створюючи надійний ланцюг блоків. Будь-хто, хто хоче змінити один блок, повинен змінити всі наступні блоки, що є не лише технічно складним, а й надзвичайно дорогим завданням.

Інший криптографічний метод, який широко використовується у блокчейні – це криптографія з публічним ключем (асиметрична криптографія). Вона допомагає здійснювати безпечні транзакції між користувачами, які можна перевірити.

Ось як це працює. Кожен учасник має унікальну пару ключів: приватний ключ, який він тримає у секреті, і публічний ключ, яким він відкрито ділиться. Коли користувач ініціює транзакцію, він підписує її своїм приватним ключем, створюючи цифровий підпис.

Потім інші користувачі мережі можуть перевірити справжність транзакції, застосувавши до цифрового підпису відкритий ключ відправника. Такий підхід гарантує безпеку транзакцій, оскільки лише законний власник приватного ключа може авторизувати транзакцію, але кожен може перевірити підписи за допомогою публічного ключа.

Ще однією особливістю блокчейну є його прозорість. Як правило, будь-хто може перевірити дані блокчейну, зокрема всі дані транзакцій і блоків, на

публічних сайтах блокчейнів. Наприклад, ви можете побачити кожну транзакцію, коли-небудь зареєстровану в мережі Bitcoin, на сайтах блокчейн explorer, зокрема ідентифікатор відправника й одержувача, суму переказу і список власників будь-якого Bitcoin. Ви також можете простежити блоки від сьогоднішнього дня (блок 788 995 станом на 18:52:21 29.05.2023 (GMT)) аж до першого блоку, відомого як генезис-блок.

1.2.3 Переваги блокчейну

Децентралізація: децентралізований характер блокчейну означає відсутність єдиної точки контролю або збою, що може зробити його більш безпечним і стійким до атак чи витоку даних.

Прозорість: транзакції у блокчейні видно всім учасникам, що спрощує відстеження і перевірку транзакцій, а також забезпечує їхню точність.

Незмінність: після того, як транзакцію зареєстровано в блокчейні, її не можна змінити або видалити. Це створює постійний запис усіх транзакцій, який може перевірити будь-хто, хто має доступ до блокчейн-мережі. Це значний відхід від традиційних систем, у яких транзакції є зворотними.

Ефективність: блокчейн може забезпечити швидші й ефективніші транзакції, оскільки він не вимагає посередників, таких як банки.

Нижчі комісії: усуваючи посередників і автоматизуючи процеси, блокчейн може знизити транзакційні витрати й підвищити ефективність деяких бізнес-операцій.

Система "без довіри": блокчейн-технологія забезпечує прозорі транзакції, що перевіряються і підтверджуються самими учасниками мережі без нав'язаних посередників.

Блокчейн-технологія пропонує безпечний і прозорий спосіб запису транзакцій та зберігання даних. Вона може здійснити революцію в промисловості, вивівши цифровий світ на новий рівень довіри й безпеки.

Блокчейн-технологія відкриває цілий світ можливостей, незалежно від того, чи це забезпечення peer-to-peer транзакцій, створення нових форм

цифрових активів чи підтримання децентралізованих програм. Оскільки технологія продовжує розвиватися й набувати ширшого прийняття, ми можемо очікувати появи більш інноваційних варіантів використання в найближчі роки.[9]

1.3 Безпечність блокчейн технології.

1.3.1 Мнемонічна фраза

Безпека доступу до активів, досягається за рахунок мнемонічної фрази. Що ж таке мнемонічна фраза? Мнемонічна фраза представляє собою набір слів, який забезпечує доступ до ваших активів та служить як резервна копія для вашого крипто-гаманця, також відома як Seed-фраза або BIP39. Ця фраза може складатися з 12, 18 або 24 слів.

Уявіть собі, що ви не зробили резервну копію свого крипто-гаманця, який використовували на смартфоні, і втратили цей смартфон. Як отримати доступ до коштів? На жаль ніяк. У світі криптовалюти немає централізованого органу, який може допомогти відновити доступ до активів, як це робиться в банківській системі. Втрата доступу та резервної копії в криптосвіті означає, що ваші активи, не будуть повернені з втраченого гаманця.

Наразі 4,8 мільйонів біткоїнів вже не в обігу, що становить приблизно 25,5% всіх біткоїнів. На жаль, точна статистика тих, хто втратив активи, відсутня, але часто можна зустріти гучні заголовки про втрати через те що ви забули мнемонічну фразу.

Надійно збережена мнемонічна фраза - це головне завдання після створення криптогаманця. Вона дозволяє відновити доступ до коштів на іншому смартфоні чи апаратному гаманці, якщо це стане необхідним.

Але, на жаль, є й ризики. Кожен, хто отримає вашу фразу, зможе отримати доступ до ваших коштів. Тому важливо надійно зберігати фразу в таємниці та уникати передачі її третім особам.

Чи можна отримати чужу мнемонічну фразу за рахунок підбору слів?

Для початку, уявіть ПІН-код із 4 цифр. На місці кожної цифри ми можемо вказати значення від 0–9. Спробуємо розрахувати кількість спроб, яка знадобиться нам для того, щоб підібрати код.

10^4 = 10 000 комбінацій. Такий перебір можливий навіть у ручному режимі. Якщо говорити про ПК, така операція займе не більше 1 секунди.

А тепер перевіримо скільки комбінацій може бути щоб отримати мнемонічну фразу, тільки тепер у нас буде не 4 цифри, а 12 мнемонічних слів. І замість значення з 0-9, у нас буде BIP39 словник, який складається з 2048 слів. 2048^12 = 5 444 517 870 735 015 415 413 993 718 908 291 383 296 комбінацій. Теоретично підбір можливий, але для цього вам знадобиться орендувати суперкомп’ютер на кілька сотень тисяч років.

А якщо у вас мнемонічна фраза з 24 слів? 2048^24 = 29 642 774 844 752 946 028 434 172 162 224 104 410 437 116 074 403 984 394 101 141 506 025 761 187 823 616 комбінацій. У даному випадку навіть суперкомп’ютер безсилий.[10]

1.3.2 Безпечність транзакцій

Для захисту транзакцій використовуються криптографічні засоби захисту. Роздивимось популярні методи:

1. Хеш-функції: Кожна транзакція отримує унікальний "хеш", який генерується за допомогою хеш-функції. Це дозволяє визначити транзакцію та виявити будь-які зміни в ній.

2. Цифрові підписи: Кожна транзакція підписується приватним ключем власника, і цей підпис може бути перевірений за допомогою його публічного ключа. Це забезпечує автентичність та не змінюваність даних.

3. Симетричне та асиметричне шифрування: Використання шифрування дозволяє захистити дані в транзакціях. Симетричне шифрування використовує один ключ для шифрування та розшифрування, тоді як асиметричне використовує пару ключів - публічний та приватний.

Всі ці елементи разом створюють безпечне середовище для здійснення транзакцій у блокчейні.

Висновки за розділом 1

Блокчейн та web 3.0 представляють суттєві технологічні зрушення, змінюючи спосіб, яким ми взаємодіємо в цифровому просторі. Блокчейн, з своєю децентралізованою природою, перетворює спосіб обміну цифровими активами та встановлення довіри, викликаючи інновації в фінансах, логістиці та багатьох інших галузях. З іншого боку, web 2.0 та 3.0 позначають етапи розвитку Інтернету, відзначаючись соціальною інтеракцією та інтернетом речей, що стає все розумнішим. Разом вони визначають майбутнє цифрової ери, де відкритість, децентралізація та інтерактивність грають ключову роль у формуванні нового, більш передового та гнучкого цифрового світу.

РОЗДІЛ 2
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Соціальна мережа Sapien .

Sapien - це соціальна мережа, яка спрямована на те, щоб надати користувачам більше контролю над своїм змістом і приватністю. Вони пропонують блокчейн-технології для забезпечення безпеки і конфіденційності, а також мають систему заохочення за корисний контент[11], інтерфейс Sapien зображений на рисунку 2.1 .

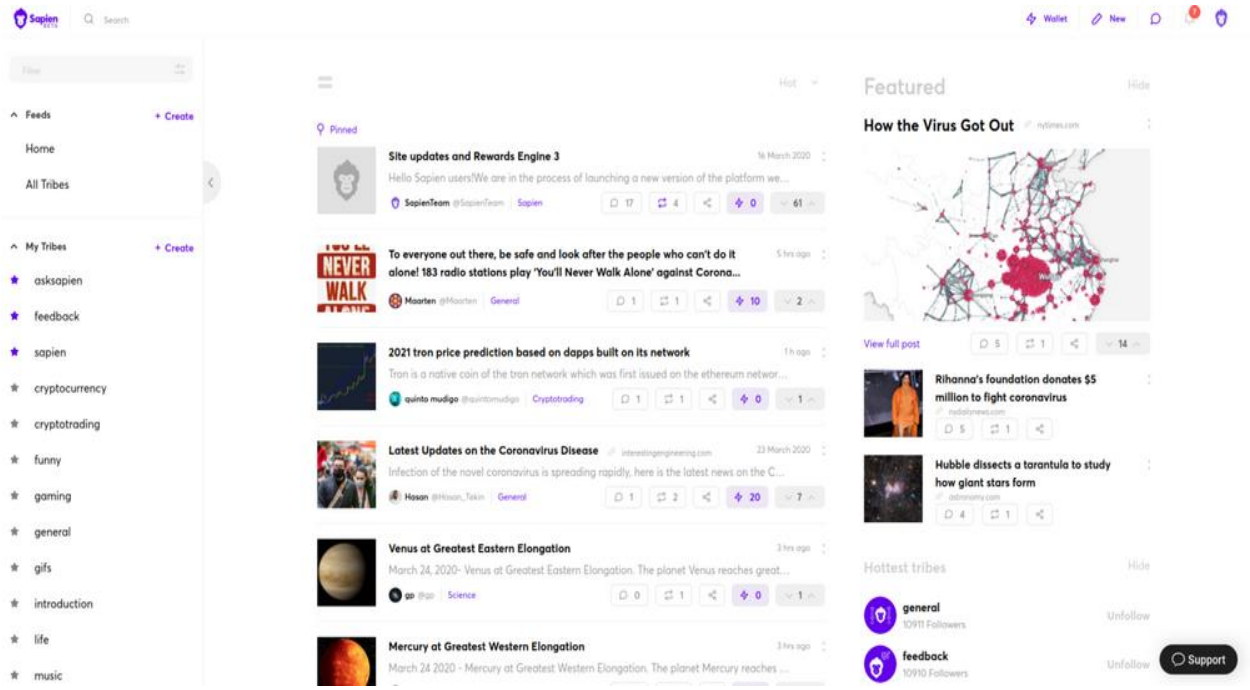

Рисунок 2.1 – Інтерфейс соціальної мережі Sapien

Тобто, користувачі можуть мати більше влади над тим, чим вони діляться і як це використовується, і отримувати винагороду за свою активність у мережі. Це одна з багатьох спроб змінити динаміку соціальних мереж на користь користувачів.

Sapien використовує технології блокчейну для реалізації своїх ідей. Зокрема, вони використовують Ethereum, одну з найпопулярніших платформ для розробки децентралізованих застосунків та смарт-контрактів. Ethereum

дозволяє створювати розподілені додатки (DApps), які працюють на базі блокчейну, забезпечуючи безпеку та непереборну історію транзакцій.

Це також дозволяє Sapien використовувати токени (криптовалюту), які можна використовувати для взаємодії з платформою, отримання винагород за контент чи інші активності, а також за збереження приватності користувачів за допомогою криптографії та розподілених систем.

Основні риси та особливості Sapien включають:

1. Блокчейн та Ethereum**:** Використання технології блокчейну, зокрема Ethereum, для забезпечення безпеки, прозорості та непереборності.
2. Приватність: Захист особистих даних та конфіденційності користувачів за допомогою криптографії та децентралізованих систем.
3. Винагороди за контент: Встановлення механізму винагородження за корисний та якісний контент для стимулювання активності користувачів.
4. Токени SPA: Використання власного токена SPA (Sapien) для проведення транзакцій та стимулювання участі у платформі.
5. Децентралізоване управління: Залучення користувачів до процесу прийняття рішень та розвитку платформи.
6. Модераторська система**:** Використання голосу користувачів у визначенні правил та модерації контенту.

Sapien ставить своєю метою створення більш справедливої та користувацькоорієнтованої соціальної мережі, де учасники мають більше влади та можливостей управління своїм власним простором в інтернеті[11].

2.2 Соціальна мережа Steemit.

Steemit - це соціальна мережа на базі блокчейну, створена для обміну контентом та винагороди творців контенту за їхній внесок у спільноту. Вона була запущена у 2016 році і використовує технологію блокчейну на основі Graphene.

Основні риси Steemit:

1. Токени STEEM: Steemit використовує власний токен STEEM для винагородження користувачів за створення та кураторство контенту.
2. Децентралізоване управління: Рішення про розподіл винагород визначається самою спільнотою на основі принципів голосування.
3. Контент-платформа: Користувачі можуть публікувати різноманітний контент, такий як статті, фотографії, відео, та отримувати за це винагороду в токенах STEEM.
4. Кураторство: Крім створення контенту, користувачі можуть отримувати винагороду за кураторство - визначення та підтримку якісного контенту.
5. Анонімність: Steemit надає відносний рівень анонімності для своїх користувачів.

Система винагородження та децентралізований підхід до управління роблять Steemit унікальним серед соціальних мереж. Кожен користувач може впливати на розподіл винагород, залежно від свого внеску та активності в спільноті [12], Інтерфейс Steemit показаний на рисунку 2.2 .

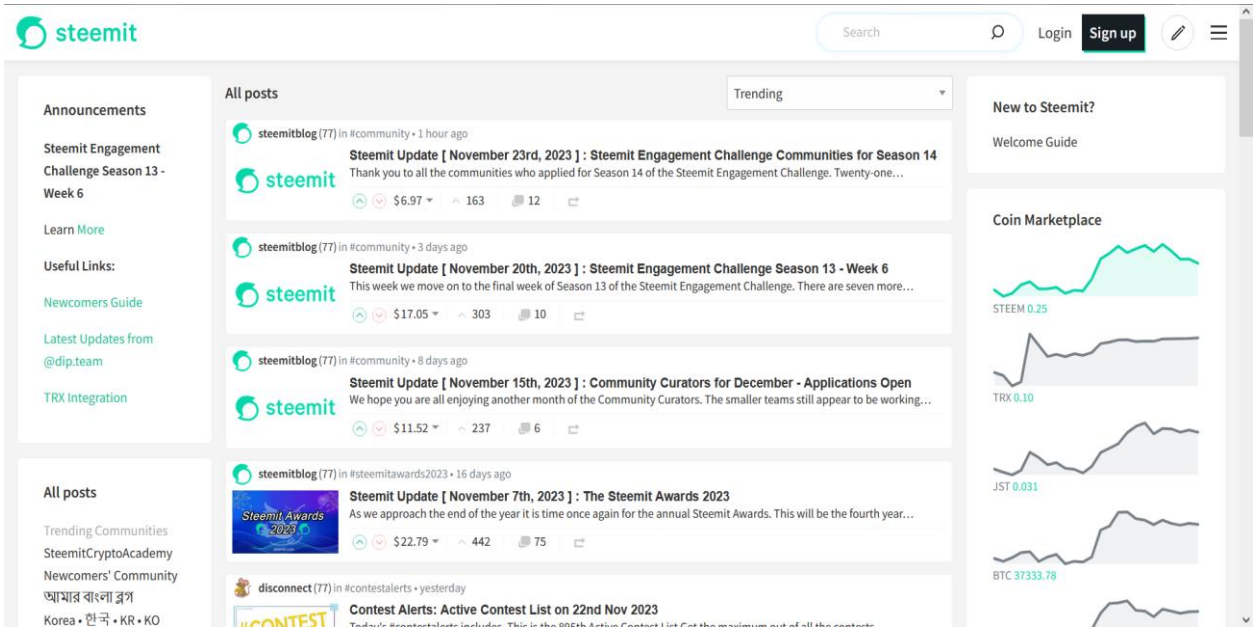

Рисунок 2.2 – Інтерфейс соціальної мережі Steemit

2.3 Соціальна мережа Sola.

Sola - це соціальна мережа, яка ставить своєю метою перетворення способу, яким користувачі взаємодіють з контентом. Основною особливістю є використання штучного інтелекту для індивідуалізації рекомендацій та вмісту, щоб кожен користувач отримував контент, який відповідає його інтересам, інтерфейс можна побачити на рисунку 2.3 .

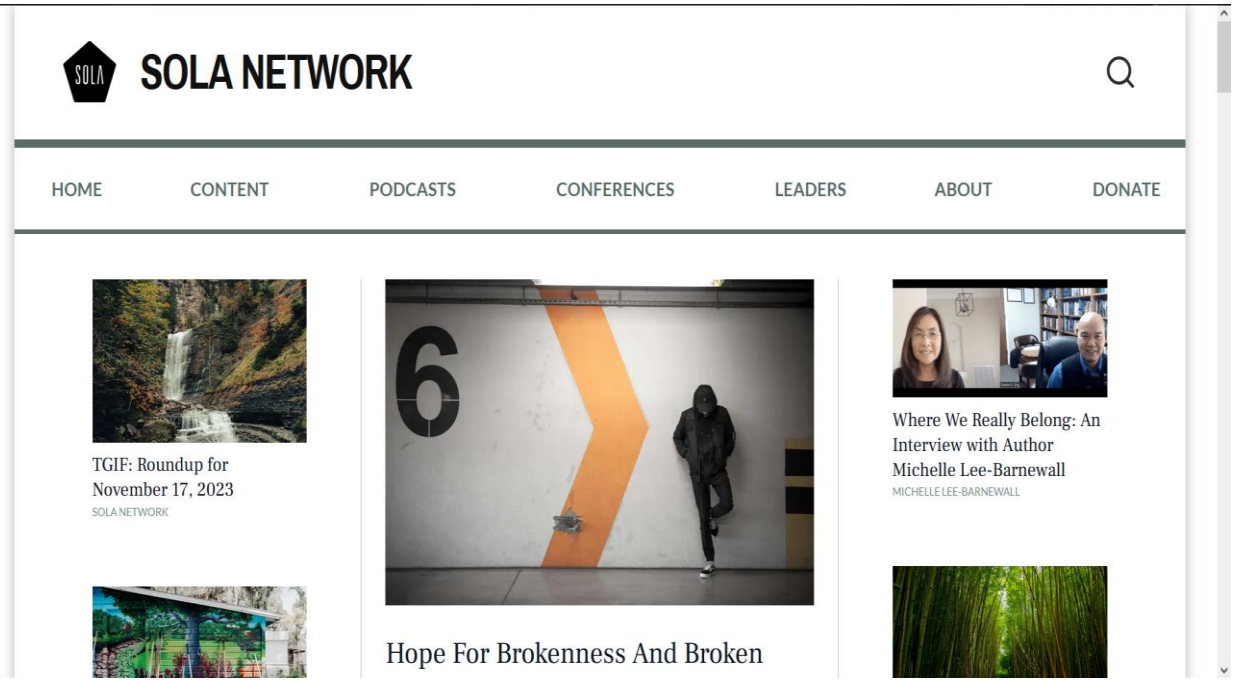

Рисунок 2.3 – Інтерфейс соціальної мережі Sola

Основні риси Sola:

1. Алгоритм AI: Використання штучного інтелекту для аналізу та рекомендацій контенту.
2. Принцип “Енергії”: Валюта на платформі, що відображає активність користувача та взаємодію з контентом.
3. Геопривабливість: Здатність контенту привертати увагу користувачів на основі їхнього місцезнаходження.
4. Децентралізована система рекомендацій: Користувачі отримують рекомендації від інших користувачів, а не від централізованого алгоритму.[13]

2.4 Соціальна мережа Indorse

Indorse - це платформа, яка використовує технології блокчейну для створення децентралізованої соціальної мережі. Основна ідея полягає в тому, щоб надати користувачам більше контролю над своєю освітою та професійною репутацією.

Основні особливості Indorse можуть включати:

1. Підтвердження навичок: Користувачі можуть додавати свої навички до профілю, і ці навички підтверджуються іншими користувачами. Це може служити як підтвердження професійної компетентності.
2. Блокчейн: Використання технології блокчейну для забезпечення надійності інформації та підтверджень.
3. Винагороди за контент: Користувачі можуть отримувати винагороду за цікавий та якісний контент, який вони додають до мережі.
4. Приватність та власність даних: Акцент на захисті приватності користувачів та наданні їм контролю над своєю освітньою та професійною інформацією.
5. Децентралізована архітектура: Спроба створити мережу, що не залежить від централізованих структур, де користувачі володіють своєю інформацією.[14]

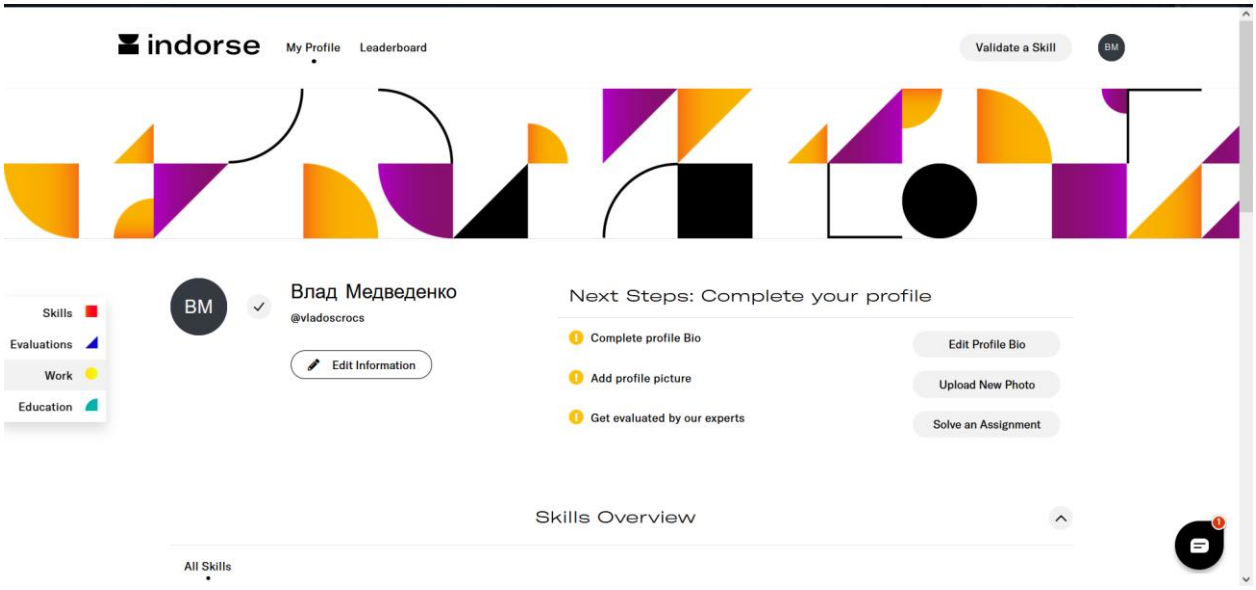

Рисунок 2.4 - Інтерфейс соціальної мережі Indorse

Висновки за розділом 2

Виходячи з проаналізованих соцальних мереж, зробимо їх порівння по основним критеріям:

1. Технологія блокчейн:
    - Sapien: Використовує Ethereum для децентралізації та забезпечення безпеки.
    - Steemit: Використовує блокчейн на основі Graphene для винагородження користувачів.
    - Sola: Має підтримку блокчейну, але конкретні технічні деталі можуть змінюватися.
    - Indorse: Використовує блокчейн для підтвердження навичок та забезпечення децентралізованості.
2. Модель винагород:
    - Sapien: Використовує токени SPA для винагороди користувачів за контент та активність.
    - Steemit: Використовує токени STEEM для винагороди за контент та кураторство.
    - Sola: Має внутрішню валюту (SOL), але конкретні механізми винагород за контент можуть різнитися.

- Indorse: Винагородження за підтвердження навичок, але докладніше - невідомо.

3. Приватність та безпека:
   - Sapien: Акцент на захисті особистих даних та приватності користувачів.
   - Steemit: Задовільний рівень анонімності.
   - Sola: Застосовується геоприваблівість та акцент на приватності.
   - Indorse: Використовує блокчейн для підтвердження та забезпечення безпеки даних.
4. Цільова аудиторія та спрямованість:
   - Sapien: Зорієнтований на контроль над власним контентом та приватністю.
   - Steemit: Орієнтований на створення та винагородження контенту.
   - Sola: Зосереджений на індивідуалізації рекомендацій та контенту.
   - Indorse: Спрямований на розвиток професійної репутації.

Загалом, кожна соціальна мережа має свої унікальні риси та принципи. Вибір між ними може залежати від вашого підходу до взаємодії з соціальними платформами, важливості приватності, бажання отримувати винагороду за контент, та інших особистих переваг.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ МОДЕЛІ

3.1 Побудова діаграм для реалізації комп'ютерної моделі соціальної мережі з використанням концепції web 3.0 та технології блокчейн

Однією з важливої частини розробки будь-якого застосунку є побудова діаграм. Для кращого розуміння програмного забезпечення діаграми дають змогу візуалізувати дані. Вони чітко визначають поставлені задачі, тож розробнику буде легше орієнтуватися та представляти як повинен виглядати кінцевий продукт.

Уніфікована мова моделювання (UML) була розроблена з метою забезпечити єдину візуальну мову з багатою семантикою та розгорнутим синтаксисом для проектування та впровадження програмних систем зі складною структурою та комплексною поведінкою. Варто зазначити, що UML застосовується не тільки у розробці програм, але й в інших сферах, наприклад, схематизувати потоки виробничих процесів [15].

UML нагадує стандарти, що використовуються в інших галузях, та підтримує діаграми кількох типів. В цілому, діаграми UML описують межі, структуру та поведінку як усієї системи, так і окремих об'єктів у її складі.

UML не є мовою програмування, проте на базі діаграм UML можна згенерувати код різними мовами, і для цього існує ряд спеціальних інструментів. Натомість з об'єктно-орієнтованим аналізом та дизайном уніфікована мова моделювання пов'язана безпосередньо.

Головною метою UML-діаграм є служити засобом комунікації між командою, також для покращення спілкування з замовником. Далі буде наведено можливі варіанти використання діаграм:

1. Проектування. При моделюванні архітектури великих проектів використовують UML-діаграми, на основі яких і буде написано програмний код;

2. Реверс-інжиніринг. Створення UML-моделей з уже існуючого коду застосунку, тобто зворотна побудова. Такий підхід може застосовуватися на проектах підтримки, де є написаний код, але документація відсутня або неповна.
3. З побудованих моделей можна розписати текстову інформацію і генерувати відносно читабельні тексти, тобто документувати. В такому випадку графічні моделі і текст будуть доповнювати один одного.

UML має власний синтаксис та правила оформлення моделей, як і будь-яка інша мова. За допомогою графічної нотації UML систему можна візуалізувати, об'єднати всі компоненти в єдину структуру і покращувати модель у процесі роботи. Графічна нотація UML на загальному рівні містить 4 основні типи елементів: фігури, з’єднувальні лінії, написи та значки. UML-нотація фактично є необхідним галузевим стандартом у сфері розробки програмного забезпечення, бізнес-систем та IT-інфраструктури [15].

Мова UML має 12 типів діаграм:

– 5 типів представляють поведінкові аспекти системи;
– 4 типи діаграм представляють статичну структуру додатку;
– 3 типи представляють фізичні аспекти функціонування системи, також їх називають діаграмами реалізації.

Найбільш використовуваними та корисними для представлення системи є наступні діаграми:

– use-case diagram (діаграма використання);
– class diagram (діаграма класів);
– activity diagram (діаграма активності);
– sequence diagram (діаграма послідовності);
– deployment diagram (діаграма розгортання);
– collaboration diagram (діаграма співробітництва);
– object diagram (діаграма об'єктів);
– statechart diagram (діаграма станів).

3.1.1 Побудова діаграми прецедентів

Згідно аналізу предметної області зроблено висновок про необхідність розгляду одного користувача. На рисунку 3.1 зображено розроблену діаграму варіантів використання:

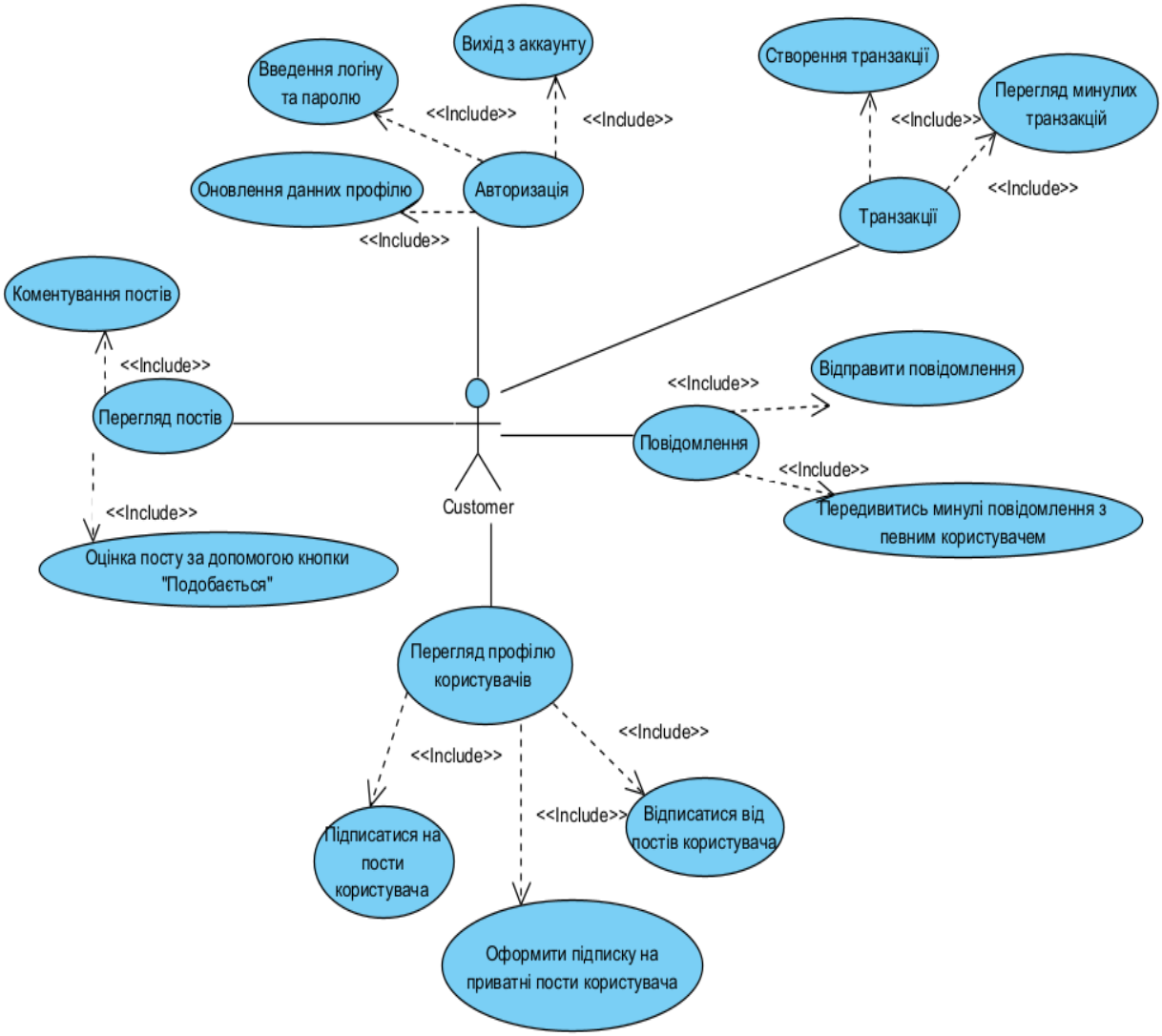

Рисунок 3.1 – Діаграма варіантів використання

3.1.2 Побудова діаграми класів

Діаграма класів заснована на аналізі предметної області. Вона визначає структуру системи, показуючи її класи, їх атрибути та методи, а також відносини між класами, на рисунку 3.2 наведено таку діаграму.

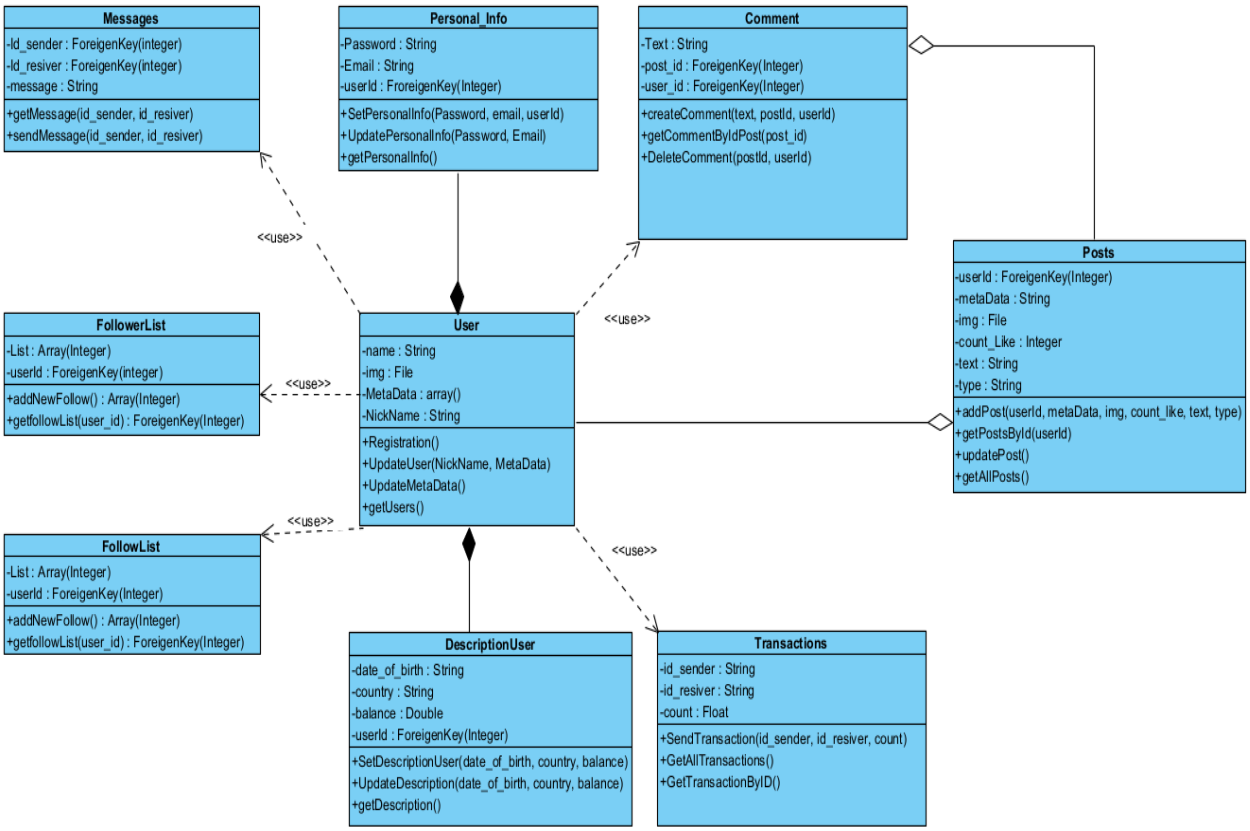

Рисунок 3.2 – Діаграма класів

3.2 Розробка структури бази даних

База даних, побудована з правильною структурою надає доступ до актуальних та точних відомостей. Оскільки правильна структура є дуже важливою для досягнення мети під час роботи з базою даних, варто розглянути основні принципи правильної структури .

Є кілька принципів для розробки структури бази даних.

Першим принципом є те, що дані, які повторюються (також їх називають надлишкові дані) – це погано, оскільки вони збільшують можливість виникнення помилок та займають зайве місце в системі.

Другий принцип полягає в тому, що повнота інформації і її правильність відіграють дуже важливу роль. Якщо база даних (БД) містить помилкові відомості, усі ресурси, які отримують інформації з неї, також будуть помилковими. І всі, хто виносить рішення на підставі цих ресурсів, буде дезінформований.

Отже, правильною структурою бази даних є та, яка:

– економить місце, тому що не містить зайвої інформації
– підтримує цілісність та точність даних
– забезпечує зручний доступ до даних

Розробка БД складається з основних дій, які описано нижче.

Огляд призначення бази даних. Це допомагає підготуватися до інших кроків та краще зрозуміти майбутню її структуру.

Пошук та упорядковування необхідної інформації. Збір усіх типів відомостей, які можуть знадобитись в БД, наприклад, назва лекції або назва курсу.

Розподіл інформації за сутностями. Потрібно розподілити елементи інформації системи на основні сутності, наприклад, «users» або «posts». З кожної сутності потім буде створено таблицю.

Перетворення елементів інформації у стовпці. Розподілення відомостей по таблицям. Кожен елемент становиться полем та відображається як стовпець таблиці. Наприклад, таблиця «Users» може містити поля «email», «password» та інші.

Вказівка первинних ключів. Вибір первинних ключів з кожної таблиці. Первинний ключ – це стовпець, який використовується як унікальний ідентифікатор кожного рядку. Прикладом може бути поле "id " в таблиці «users» або поле "id" в таблиці «posts».

Поєднання таблиць. Потрібно розібратися як дані в одній таблиці, зв’язані з даними з інших таблиць. За необхідності додати поля до таблиць або створити нові таблиці для уточнення зв'язків.

Удосконалення структури. Аналіз структури та виявлення помилок. Створення таблиць та додавання кількох записів із зразками даних. Перевірка на отримання результатів з таблиць. За потреби вносяться зміни до структури БД. На рисунку 3.3 зображено структуру бази даних.

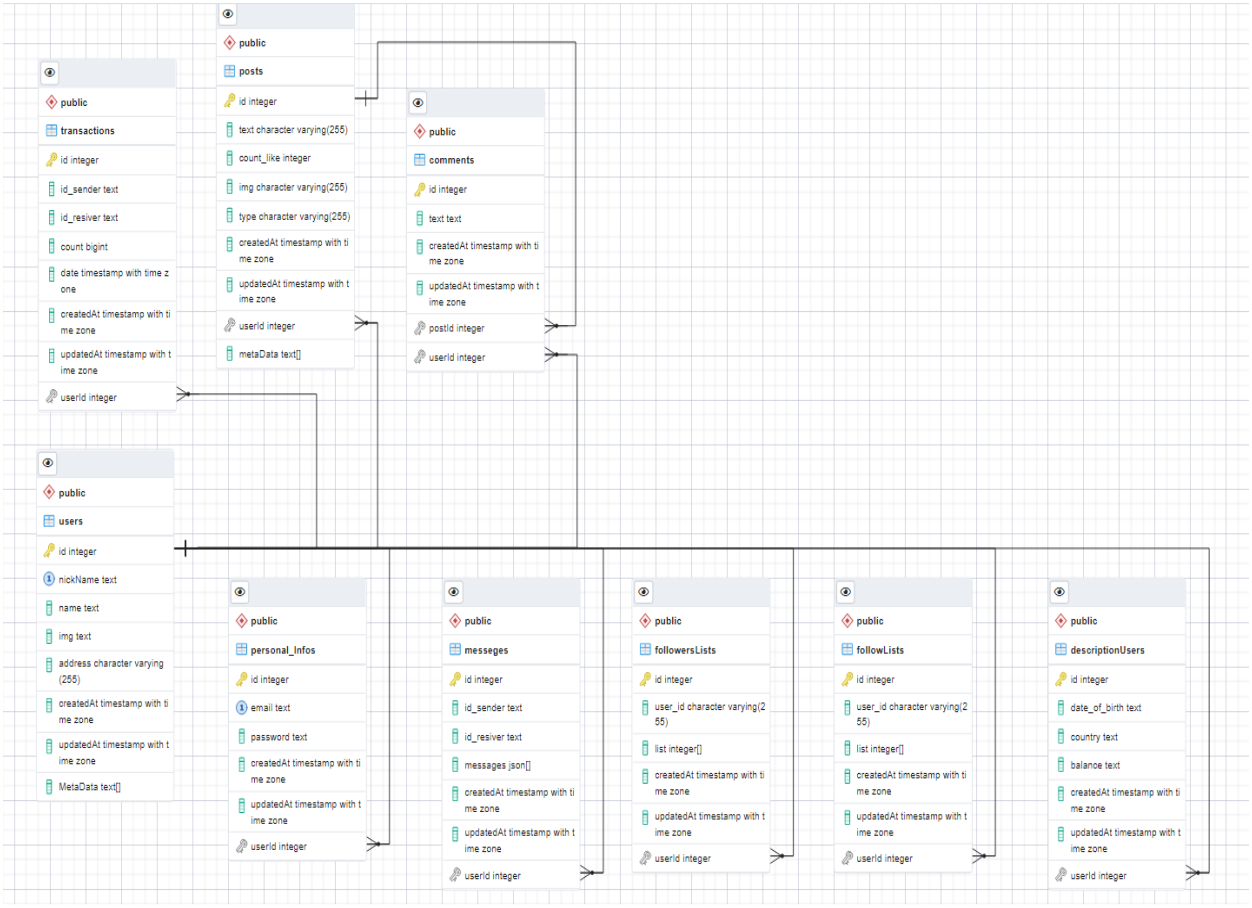

Рисунок 3.3 – Схема бази даних.

Наступним кроком проектування структури бази даних буде застосування правил нормалізації даних. Ці правила дають змогу перевірити, чи правильно було структуровано таблиці.

Правила нормалізації слід використовувати після того, як було створено попередню структуру та представлено всі елементи інформації. Так можна впевнитися, що елементи інформації розподіллено по відповідних таблицях. Це потрібно зробити, тому що нормалізація не дає змоги перевірити, чи є всі правильні елементи інформації, з яких варто починати .

Ці правила слід застосовувати послідовно, перевіряючи на кожному кроці, чи структура відповідає відповідним критеріям «нормальних форм». Загальноприйнятими є п’ять нормальних форм: від першої нормальної форми до п’ятої. Розглянемо перші три з них, тому що їх достатньо для більшості структур баз даних.

Перша нормальна форма. Відношення знаходиться в першій нормальній

формі, якщо всі його атрибути є простими, всі домени повинні містити тільки скалярні значення. Не повинно бути повторень рядків у таблиці.

Друга нормальна форма. Відношення знаходиться в другій нормальній формі, якщо воно знаходиться в першій нормальній формі і кожен не ключовий атрибут безпосередньо залежить від Первинного ключа. Безпосередність означає, що у складі потенційного ключа відсутня менша підмножина атрибутів, яку можна вивести цю функціональну залежність.

Припустимо, є таблиця, що містить наведені нижче стовпці, де «ID user» та «ID post» утворюють первинний ключ. Приклад наведено в таблиці 3.1.

Таблиця 3.1

Приклад порушення другої нормальної форми

ID user	Первинний ключ
ID post	Первинний ключ
name	

Структура порушує другу нормальну форму, тому що стовпець «name» залежить від стовпця «ID user», але не залежить від стовпця «ID post», тому не залежить від усього первинного ключа. Необхідно видалити з таблиці стовпець «name». Він належить до іншої таблиці «user».

Третя нормальна форма. Відношення знаходиться в третій нормальній формі, коли знаходиться в другій нормальній формі і кожен ключовий атрибут нетранзитивно залежить від первинного ключа. Проще кажучи, друге правило вимагає виносити всі ключові поля, вміст яких може ставитися до кількох записів таблиці в окремі таблиці. Наприклад, таблиця має такий вигляд, як наведено у таблиці 3.2.

Таблиця 3.2

Приклад порушення третьої нормальної форми

ID user	Первинний ключ
ID post	Первинний ключ
text	

user_name	

«user_name» залежить від поля «name», яке знаходиться в іншій таблиці. Тоді, ця таблиця буде порушувати третю звичайну форму, оскільки стовпець без ключа, «user_name», залежить від іншого стовпця, який не є ключем. Незалежність стовпця означає, що ви зможете змінити будь-який стовпець, який не є ключем, не впливаючи на інші стовпці.

Застосування правил нормалізації є необхідною частиною проектування та розробки бази даних. Вони використовуються для оптимізації та проектування бази даних. Потрібно використовувати правила нормалізації даних, щоб перевірити структуру таблиць. За необхідності внести зміни до таблиці. Також правила нормалізації забезпечують цілісність даних та спрощують підтримку бази даних.

3.3 Розробка вимог до програмного забезпечення

3.3.1 Функціональні вимоги

Функціонал взаємодії з веб-застосунком може бути представлений у вигляді такого списку прецедентів:

– авторизація користувача;

– реєстрація користувача;

– користувач може писати повідомлення;

– користувач може створювати публікації;

– користувач може писати коментарі до публікацій які йому доступні;

– користувач може добувати монети(умовна валюта в комп’ютерній моделі) за допомогою блокчейн серверу;

– користувач може відправити та отримати монети(умовна валюта в комп’ютерній моделі);

– коричтувач може оновити данні про себе;

– користувач може видалити публікацію який він сам створив;

– користувач може підписатися на публікації інших користувачів;

– користувач може підписатися на приватні публікації за певних умов

3.3.2 Нефункціональні вимоги

Далі наведено нефункціональні вимоги до програмного забезпечення:

1. Програмне забезпечення (ПЗ) повинно бути простим в обслуговуванні та підтримці.

2. ПЗ повинно бути сумісним з різними платформами, портативним. Тож перехід від однієї ОС до іншої ОС не створює жодних проблем. Це стосується програмного забезпечення як на стороні сервера, так і на стороні клієнта. Програмне забезпечення на стороні клієнта повинне бути сумісною з браузерами Microsoft Edge, Google Chrome, Opera , Mozilla Firefox.

3. Веб-додаток повинен бути достатньо спроможним обробляти 1000 користувачів без помітної втрати швидкості. При цьому мінімальна затримка відповіді від сервера повинна не перевищувати 3 секунди.

4. Комп’ютерна модель повинна бути легкою для засвоєння як досвідченими, так і початковими користувачами.

5. Веб-додаток повинен забезпечувати простий, навігаційний та зручний інтерфейс, який дозволяє використовувати основні функції з достатньою легкістю.

3.4 Обгрунтований вибір стеку технологій для створення комп`ютерної моделі

3.4.1 Архітектура веб-застосунку

Архітектурою системи обрано клієнт-серверну архітектуру.

Клієнт-серверна архітектура - це структурний підхід до побудови програмного забезпечення, де функції системи розділені між клієнтськими пристроями, які ініціюють запити, та серверами, які відповідають на ці запити, забезпечуючи необхідні послуги чи ресурси через мережу.

Вона дозволяє ефективно розподіляти завдання та ресурси між клієнтами і серверами, сприяючи швидкій обробці запитів та полегшуючи зміни в системі. Клієнти взаємодіють із серверами, відсилаючи їм запити, а сервери відповідають, забезпечуючи необхідні дані чи сервіси. Це дозволяє створювати розподілені та масштабовані системи, що забезпечують зручний доступ до ресурсів через мережу.

Крім того, клієнт-серверна архітектура спрощує розвиток і супровід програмного забезпечення, оскільки зміни в логіці чи функціональності можна вносити незалежно в клієнтській чи серверній частині. Це забезпечує модульність і гнучкість системи, а також дозволяє різним типам пристроїв та платформ взаємодіяти з однією серверною системою. Клієнт-серверна архітектура є основою багатьох сучасних додатків та сервісів, забезпечуючи ефективну комунікацію та обмін даними у великих мережевих середовищах.

Ця архітектура також підвищує безпеку, оскільки сервер може контролювати доступ до ресурсів і встановлювати правила автентифікації та авторизації. Вона розвивається разом із зростанням обсягів даних та популярністю хмарних технологій, де сервери можуть надавати послуги через Інтернет. Клієнт-серверна архітектура стала важливим елементом в інтернет-екосистемі, забезпечуючи ефективну співпрацю та обмін інформацією між різними пристроями та користувачами.

На рисунку 3.4 зображено принцип роботи дволанкової архітектури.

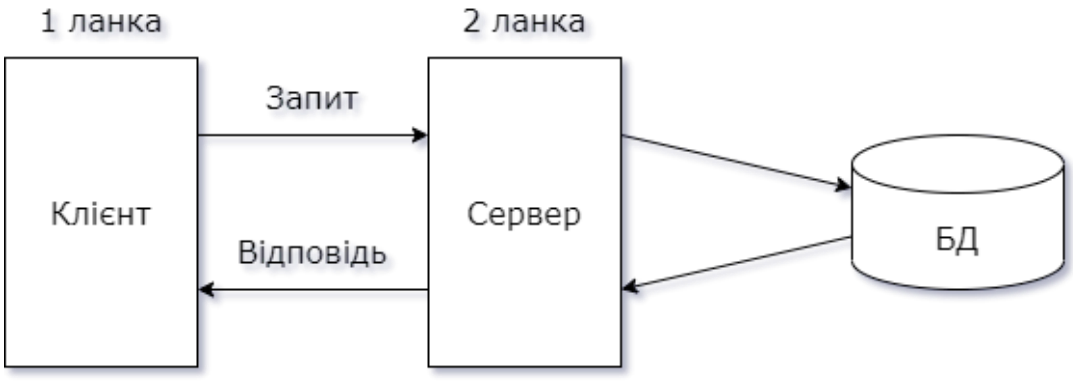

Рисунок 3.4 – Дволанкова архітектура

Розглянемо її основні переваги:

– можливість, в більшості випадків, розподілити функції обчислювальної системи між декількома незалежними комп'ютерами в мережі;
– всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів, а також на сервері простіше забезпечити контроль повноважень, щоб вирішувати доступ до даних тільки користувачам з відповідними правами доступу;
– підтримка багатокористувацької роботи;
– гарантія цілісності даних.

3.4.2 Вибір технологій для створення веб-додатку

Для розробки серверної частини веб-застосунку обрано Node.JS, що є одним із найпопулярніших серверних фреймворків мови JavaScript. Node.js використовує асинхронну модель вводу/виводу, що дозволяє ефективно обробляти багатозадачні операції без блокування виконання коду. Це особливо важливо для високовідмінних додатків, таких як веб-сервери. Також Node.js дозволяє легко масштабувати додатки, особливо завдяки асинхронності та можливості розгортати додатки на різних серверах для балансування навантаження.[16]

Express - це мінімалістичний та гнучкий веб-фреймворк для Node.js, існуючий з метою спростити розробку серверної частини веб-додатків.[16] Ось кілька причин, чому розробники вибирають Express:

Для моделювання інтерфейсу користувача буде використано фреймворк React, який оснований на мові програмування JavaScript. React базується на компонентній моделі, де інтерфейс розділяється на невеликі ізольовані компоненти. Це полегшує розробку, тестування та підтримку коду, а також сприяє його перевикористанню. React використовує віртуальний DOM, що дозволяє зменшити витрати ресурсів на маніпулювання реальним DOM. Це призводить до поліпшеної продуктивності та швидкої відповіді на зміни стану додатку. Також React чудово підходить для розробки односторінкових

додатків, де весь інтерфейс завантажується один раз, і зміни сторінок відбуваються без перезавантаження сторінки.

Для зберігання деяких тимчасових даних на стороні клієнта та керування станом програми буде використано бібліотеку керування станами Mobx. MobX - це бібліотека для керування станом застосунків у JavaScript-додатках. Вона надає ефективний та декларативний спосіб управління станом, що дозволяє автоматизувати оновлення користувацького інтерфейсу при зміні даних. MobX надає простий та зрозумілий інтерфейс для управління станом, що робить його популярним в розробці веб-додатків на базі React та інших бібліотек. Він підтримує реактивний підхід до програмування, де інтерфейс користувача оновлюється автоматично при зміні даних, що сприяє прозорості та ефективності в розробці.

3.4.3 Вибір середовища розробки

Для розробки даного проекту обрано інтегроване середовище розробки (IDE) Microsoft Visual Studio Code, оскільки воно має багато зручних для розробки розширень та зрозумілий інтерфейс.

Основні переваги вибору даного середовища:

- розумне доповнення коду, яке значно прискоряє процес написання програм;
- багатофункціональність середовища;
- інтеграція з Git;

3.4.4 Огляд і вибір СКБД

Системи керування базами даних (СКБД) різняться за своєю призначеністю та характеристиками. Ось кілька типових СКБД:

- **Реляційні СКБД (RDBMS):** найпоширеніший тип, в якому дані організовані у вигляді таблиць з рядками і стовпцями. Приклади: MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server.

- **Нереляційні СКБД (NoSQL):** використовуються для зберігання та обробки неструктурованих даних або великих обсягів даних. Приклади: MongoDB (документ-орієнтована), Cassandra (ключ-значення), Redis (структуровані дані).

- **Об'єктно-орієнтовані СКБД:** спрямовані на роботу з об'єктами, які можуть бути збережені і відновлені з бази даних. Приклади: db4o, ObjectDB.
- **Вбудовані СКБД:** інтегровані безпосередньо в додатки чи пристрої. Приклади: SQLite (легковагова реляційна СКБД, яка часто використовується для вбудовування в мобільні додатки).
- **Колоночні СКБД:** орієнтовані на зберігання даних у вигляді стовпців, а не рядків, що може бути ефективним для операцій аналізу даних. Приклади: Google Bigtable, Apache HBase.
- **Тимчасові СКБД:** використовуються для тимчасового зберігання та обробки даних. Приклади: Redis (як в кеш-пам'ять).

Ці типи СКБД відображають різноманітні потреби та сценарії використання, дозволяючи вибрати той, який найкраще відповідає конкретним вимогам проекту.

Я обрав PostgreSQL, у порівнянні з MySQL вона більше відповідає стандарту SQL. Якщо MySQL насамперед орієнтована на стабільність, надійність та простоту, то PostgreSQL – на інновації та розширену функціональність.

Будучи об'єктно-реляційною, PostgreSQL забезпечує такі функції, як успадкування таблиць та навантаження функцій. Підтримує безліч типів даних, включаючи JSON, XML, геопросторові дані, "ключ-значення" та інші.

Це система, що розширюється, можна скористатися одним з безлічі готових розширень або створити власне.

Рекомендується для завдань, де потрібна багатофункціональна БД, здатна зберігати масивні обсяги даних та обробляти складні запити:

– побудова невеликих DWH (Data Warehouse) для аналітичних систем;
– сховище для геоінформаційних систем - спільно з розширенням PostGIS;
– основне сховище для веб-додатків, мобільних програм, ігор.

3.4.5 Детальний опис розробленої технології блокчейн

Блокчейн у перекладі - це ланцюжок блоків який не переривається. Містить усі записи транзакцій. На відміну від традиційних баз даних, ці записи не можна змінювати або видаляти, можна лише додавати нові записи на рисунку 3.5 зображено програмну реалізацію класу “Block”.

```
class Block {
  constructor(index, timestamp, transactions, previousHash = '') {
    this.index = index;
    this.timestamp = timestamp;
    this.transactions = transactions;
    this.previousHash = previousHash;
    this.hash = this.calculateHash();
    this.nonce = 0;
  }

  calculateHash() {
    return SHA256(
      this.index +
      this.timestamp +
      JSON.stringify(this.transactions) +
      this.previousHash +
      this.nonce
    ).toString();
  }
```

Рисунок 3.5 – програмна реалізація класу “Block”.

Блокчейн також відомий як технологія розподілених реєстрів, оскільки весь ланцюжок транзакцій і поточний список власників зберігаються на комп’ютерах багатьох незалежних користувачів. Жодна інформація не втрачається, навіть якщо один або кілька комп’ютерів виходять з ладу.

Облік операцій — це запис усіх передач активів або прав на ці активи від однієї особи до іншої, програмна реалізація конструктора класу “Blockchain” зображена на рисунку 3.6 .

```
class Blockchain {
  constructor() {
    this.chain = [this.createGenesisBlock()];
    this.difficulty = 2;
    this.pendingTransactions = [];
    this.miningReward = 100;
    this.nodes = {};
    this.users = {};
    this.secretKey = process.env.BLOCKCHAIN_SECRET_KEY
```

Рисунок 3.6 – конструктор класу “Blockchain”

І тут виникає важливе питання: наскільки надійним і конфіденційним є механізм підтвердження переходу прав? Технологія блокчейн зменшує ці ризики, оскільки забезпечує систему обліку на основі розподілених реєстрів.

У блокчейні записи власників не зберігаються на сервері окремої організації. Його копії оновлюються одночасно на багатьох незалежних комп'ютерах, підключених через Інтернет. Таким чином, неможливо підробити реєстр, що містить дані про власників активів у блокчейні. Адже ці дані зберігаються на комп'ютерах великої кількості учасників мережі. Щоб гарантувати, що всі користувачі мають абсолютно повну і достовірну інформацію, в блокчейн введено поняття консенсусу, програмна реалізація якого зображена на рисунку 3.7 .

```
proofOfWork(lastProof) {
  let proof = 0;
  while (!this.validProof(lastProof, proof)) {
    proof++;
  }
  return proof;
}

validProof(lastProof, proof) {
  const guess = `${lastProof}${proof}`;
  const guessHash = CryptoJS.SHA256(guess).toString();
  return guessHash.substr(0, 4) === '0000';
}
```

Рисунок 3.7 – програмна реалізація алгоритму консенсусу.

Якщо деякі учасники мережі вимикають свої комп'ютери, а деякі транзакції на них не відображаються або їхні записи неправильні, це не вплине на роботу мережі. Процес консенсусу, тобто досягнення згоди, допоможе відновити точність інформації. У реальній мережі блокчейн протягом певного періоду часу відбувається багато транзакцій. І записи транзакцій включені в єдиний блок.

Блок - це запис у розподіленому реєстрі кількох транзакцій, програмна реацізація методу створення транзакції показана на рисунку 3.8 . Він показує, кому, кому і коли скільки активів передати.

```
createTransaction(req, res) {
  const { fromAddress, toAddress, amount } = req.body;
  if (!fromAddress || !toAddress || !amount) {
    return res.status(400).json({ error: 'Invalid transaction data' });
  }
  this.users[fromAddress].balance = 200;
  const transaction = new Transaction(fromAddress, toAddress, amount);

  // Check if the sender has sufficient balance
  const senderBalance = this.users[fromAddress] ? this.users[fromAddress].balance : 0;
  if (senderBalance < amount) {
    return res.status(400).json({ error: 'Insufficient funds' });
  }
  const secretKey = this.secretKey
  this.createTransaction(transaction, secretKey);
  this.saveTransactionToDatabase(transaction);

  // Update user balances
  this.users[fromAddress].balance -= amount;
  this.users[toAddress] = this.users[toAddress] || { balance: 0 };
  this.users[toAddress].balance += amount;

  return res.status(201).json({ message: 'Transaction created successfully' });
}
```

Рисунок 3.8 – програмна реалізація метода створення транзакції.

Усі блоки з'єднані послідовно відповідно до схеми послідовності. Ланцюжок блокчейну неможливо зламати, оскільки кожен блок містить посилання на попередній блок. Блоки не можна редагувати або видаляти, ви можете лише додавати нові блоки. Тож ви завжди можете відновити історію переходу того чи іншого активу з одних рук в інші та дізнатися його поточного власника.

Майнери додають нові блоки в ланцюг. Майнер виконує кілька функцій у блокчейні:

- зберігає копії блокчейну і таким чином захищає інформацію від втрати або підробки;
- підтверджує операцію;
- перевіряє транзакції, записані іншими майнерами.

За загальним правилом кількість майнерів не обмежена. Чим більше, тим краще: така мережа надійніша. Будь-хто може стати майнером. Для цього їм потрібні комп'ютери та спеціальне програмне забезпечення. Але що спонукає майнерів записувати нові транзакції? Майнери отримують винагороду за підтримку мережі. Зазвичай це комісія, яку сплачують усі учасники транзакції, записаної в блоці, і винагорода від самої мережі.

Мережа генерує цю винагороду за певним алгоритмом. Це те, що зазвичай відбувається з криптовалютами: винагородою є певна сума криптовалюти. Таким чином випускаються нові одиниці віртуальної валюти, і загальна сума віртуальної валюти збільшується.

Але найчастіше є обмеження: коли загальна кількість монет досягає певного максимуму, їх виробництво припиняється. Після цього, майнери можуть працювати лише для отримання винагороди від учасників.

Але хто з багатьох майнерів матиме право додати блок і отримати за це винагороду? Для цього більшість блокчейн-мереж створюють спеціальні завдання. Імовірність успіху майнера - чи вирішить він першим завдання, запропоноване мережею, приєднає блок і отримає за це винагороду. Найчастіше це залежить від: потужності його обладнання. Чим ефективніші їхні комп'ютери, тим більші шанси заробити монети.

Як ми можемо переконатися, що інформація про транзакції та статус гаманця точні, повні та безпечні? Для вирішення цих проблем існує ціла наука: криптографія. Одним із методів є шифрування. У мережі блокчейн покупці та продавці активів підтверджують транзакції за допомогою криптографічного ключа – спеціального унікального цифрового коду. Майже неможливо вгадати рядок символів криптографічного ключа. Це робить технологію блокчейн

однією з найкращих технологій для фінансових операцій. Для генерації публічного і закритого ключа я використовував бібліотеку bip39.

3.4.6 Опис методів

Програмна реалізація блокчейну міститься у додатку Г.

Метод *createUser(address: string): Promise<User>* - Створює нового користувача з заданим ім'ям та початковим балансом.

Метод *getUserById(userId: number): Promise<User | null>* - Отримує користувача за ідентифікатором.

Клас **Transaction:**

Метод *createTransaction(senderId: number, recipientId: number, amount: number): Promise<Transaction>* - Створює нову транзакцію між двома користувачами.

Метод *getTransactionsByUserId(userId: number):Promise <Transaction[]>* - Отримує всі транзакції для конкретного користувача (як відправника або одержувача).

Клас **Blockchain**:

Метод *createBlock(previousHash?: string): Block* - Створює новий блок з попереднім хешем. Включає всі транзакції, що чекають на підтвердження.

Метод *mineBlock(): Promise<Block>* - "Добуває" новий блок, викликаючи метод createBlock. Збільшує баланс майнера.

Метод *addTransaction(senderId: number, recipientId: number, amount: number): Promise<void>* - Додає нову транзакцію до очікуваних транзакцій у блокчейні.

Метод processPendingTransactions(): Promise<void> - Обробляє всі очікувані транзакції, додаючи їх до нового блоку.

Висновок за розділом 3

Було спроектовано базу даних та побудовано схему до неї. Після чого було встановлено функціональні та не функціональні вимоги до реалізації

комп’ютерної моделі соціальної мережі з використанням концепції web 3.0 та технології блокчейн. До бази даних було застосовано три правила нормалізації. Було побудовано дві UML діаграми, а саме діаграма прецедентів та діаграма класів, було продемонстровано основні аспекти алгоритму програмної реалізації технології блокчейн для комп’ютерної моделі соціальної мережі.

Для реалізації комп’ютерної моделі соціальної мережі було обрано наступний стек технологій та застосунків:

- Архітектура: клієнт-серверна архітектура, тому що всі дані зберігаються на сервері, має можливість розподілити функції обчислювальної системи між декількома незалежними комп'ютерами в мережі.
- СКБД: PostgreSQL, тому що вона підтримує безліч типів даних, включаючи JSON, XML, геопросторові дані, "ключ-значення". Має розширену функціональність
- Середовище розробки: Microsoft Visual Studio Code, тому що це найбільш зручне для мене середовище, та тому що можна встановити безліч додатків, які прискорюють розробку.
- Бібліотека керування станами: MobX він підтримує реактивний підхід до програмування та надає простий та зрозумілий інтерфейс для управління станом
- Моделювання інтерфейсів користувача: бібліотека React, тому що весь інтерфейс завантажується один раз, і зміни сторінок відбуваються без перезавантаження сторінки.
- Бібліотека для розробки серверної частини: Node.js використовує асинхронну модель вводу/виводу, що дозволяє ефективно обробляти багатозадачні операції без блокування виконання коду, та легко масштабувати додатки.

РОЗДІЛ 4
ТЕСТУВАННЯ ОСНОВНИХ ФУНКЦІЙ КОМП'ЮТЕРНОЇ МОДЕЛІ

4.1 Запуск проекту

Тестування комп'ютерної моделі відбуватиметься на локальній машині, тобто сервери та клієнт будуть запущені на одному комп'ютері.

Для того щоб запустити проект, потрібно відкрити термінал, і в ньому відкрити основну папку серверу, і ввести команду "npm run dev", після чого запуститься основний сервер, щоб переконатися що основний сервер запустився треба подивитися в термінал, і там повинно бути наступне повідомлення: "Server started on port 5000" як на рисунку 4.1 , що вказуватиме на те що сервер працює на порті 5000.

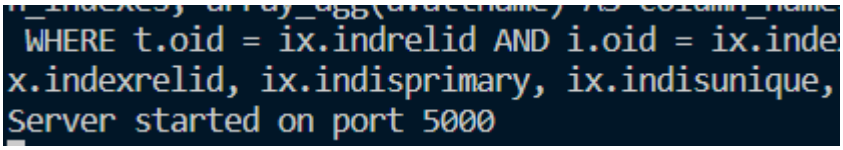

Рисунок 4.1 - Повідомлення про успішний запуск серверу

Відкриваємо новий термінал, і також знаходимо основну папку серверу. Наступною дією буде запуск серверу блокчейн. Запускається він наступним чином: потрібно ввести команду "npm run blockchain". Перевірити успішність можна також подивившись в термінал, і побачити наступне повідомлення: "Server started on port 4000" як на рисунку 4.2 , що вказуватиме на те що сервер працює на порті 4000.

Рисунок 4.2 – Повідомлення про успішний запуск серверу блокчейн.

Наступною дією буде запуск клієнта. Для цього потрібно відкрити ще один термінал, і перейти в основну папку клієнта, потім ввести команду “npm start” після чого webpack “збере” проект і клієнтська частина буде доступна за адресою “http://localhost:3000/”.

4.2 Тестування взаємодії клінту з сервером та перегляд інтерфейсу.

Будемо тестувати як новий користувач комп’ютерної моделі, тобто почнемо з регістрації в комп’ютерній моделі соціальної мережі. Для цього нам потрібно натиснути на кнопку “Login” у верхній частині вікна браузера що можна побачити на рисунку 4.3 .

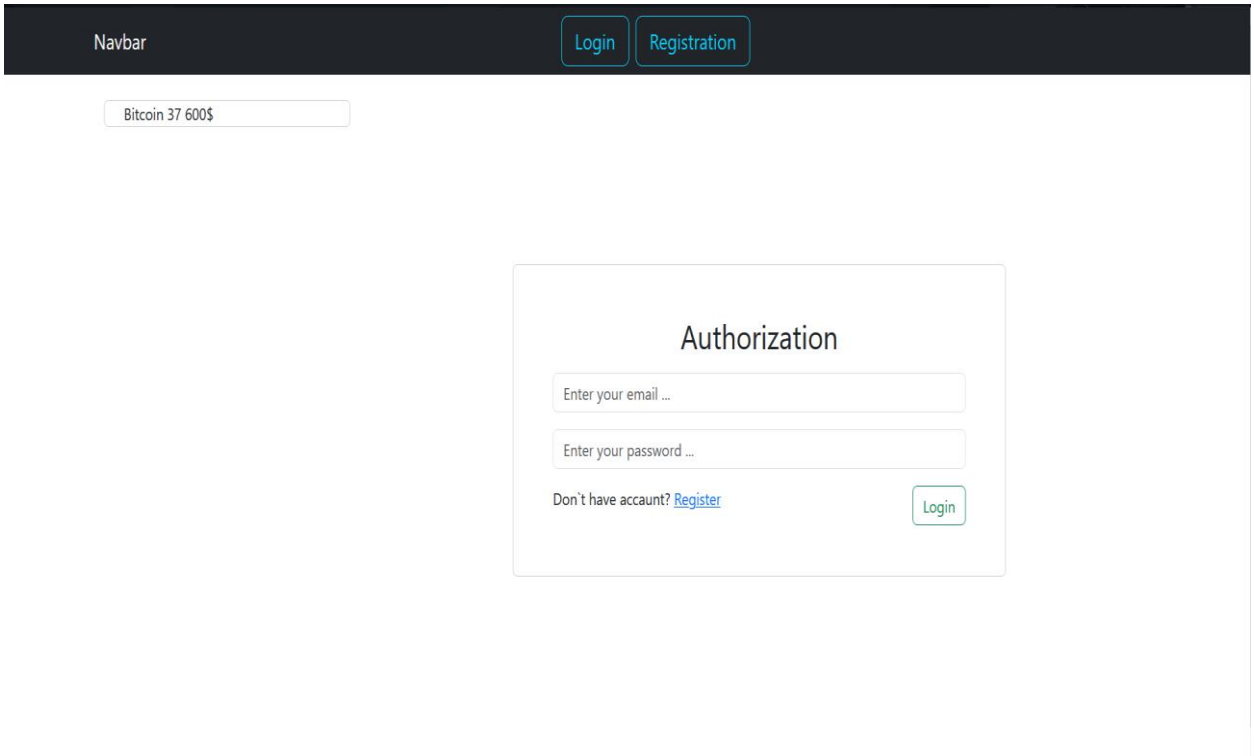

Рисунок 4.3 – Інтерфейс початкової сторінки користувача.

Далі, так як ми ще немаємо аккаунту, натискаємо на кнопку “Register”, у вікні авторизації(див. рис. 5.2.1), відбувається автоматичне перенаправлення за маршрутом “http://localhost:3000/registration” на сторінку регістрації інтерфейс якої можна побачити на рисунку 4.4 . В формі реєстрації ми вводимо данні яких потребує реєстрація.

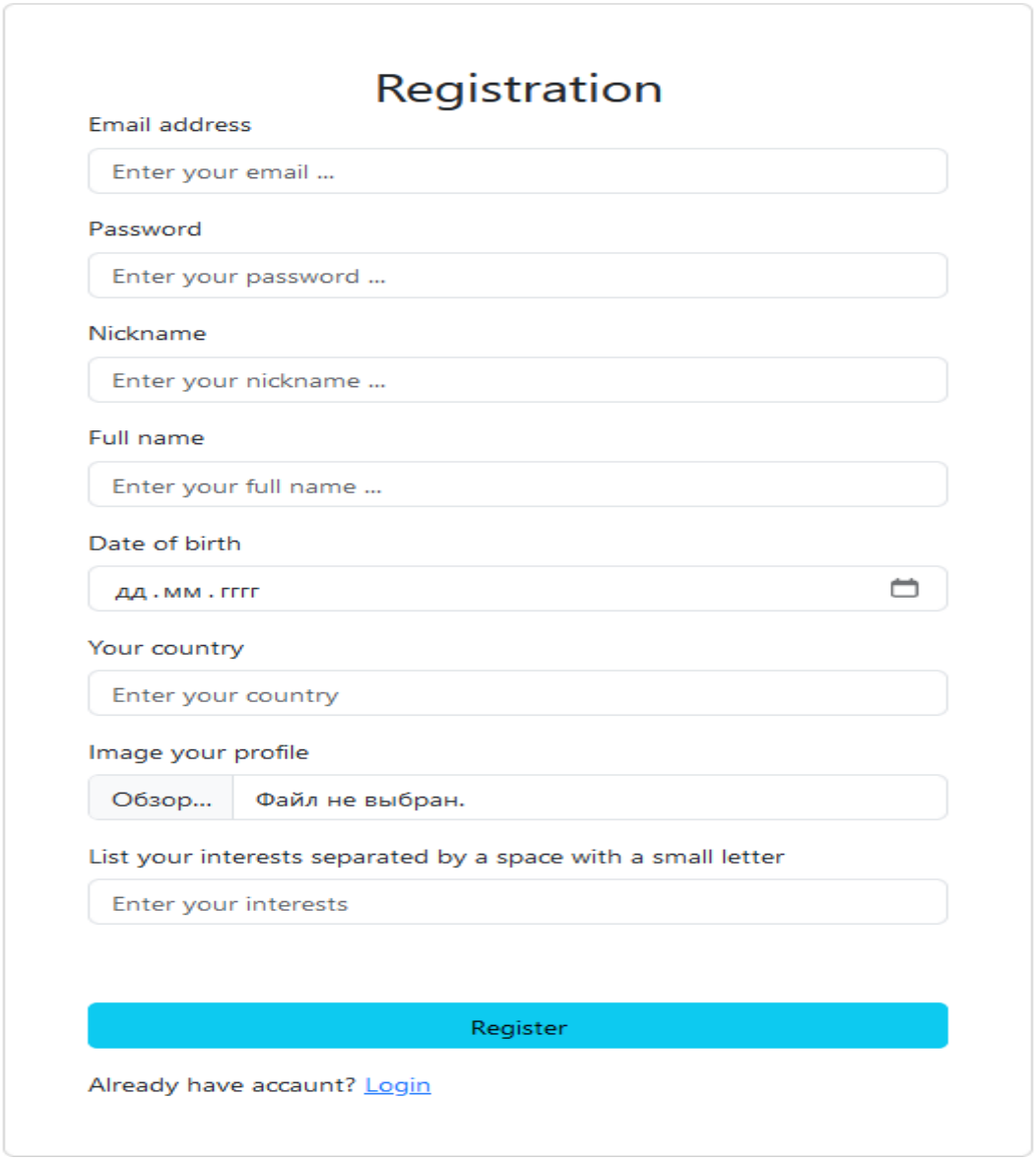

Рисунок 4.4 – Форма сторінці реєстрації.

Після вводу даних запит на реєстрацію з даними, які ввів користувач, відправляється на сервер, де проходить обробка запиту. Тобто на сервері відбувається валідація даних, на можливі помилки, наприклад якщо e-mail або nickName вже існують, то тоді реєстрація неможлива, так як ці данні мають бути унікальними для кожного окремого користувача. Також під час реєєстрація відбувається запит на сервер з блокчейном, і там автоматично користувачу присвоюється адреса згенерована за допомогою функції “uuid”, і також присвоюється баланс у 200 (умовна валюта в комп’ютерній моделі) монет для можливості користування системою. Для того щоб впевнитись що користувач був доданий до бази даних, можемо відкрити застосунок “PgAdmin

4”(застосунок для управлінням базами даних PostgresQL), і передивитись данні таблиці “User” на рисунку 4.5 можемо побачити що користувач дійсно з’явився в базі даних, і має ті значення, які ми вводили при реєстрації.

	id [PK] integer	nickName text	name text	img text	address character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	VladDjarah...	Vlad Dobrinin	e5459b92-a7be-45fa-86fa-6f0bd3dee475.j...	d077400a-938e-4225-9b6e-817ea94561...	2023-11-18 21:37:03.317+...	2023-11-18 21:37:03.317+...

Рисунок 4.5 – Запис в таблиці “User”.

Після реєстрації користувач може починати користуватись веб-застосунком. Для цього у верхній частині відкритого вікна(див. рис. 4.3) є кнопка для авторизації, натиснувши на яку відкриється вікно для авторизації(див. рис. 4.3).

В формі авторизації користувач вводить данні які він вводив під час реєстрації. Після авторизації до localStorage(веб-сховище яке дозволяє зберігати дані в браузері після повного закриття і нового відкриття вікна браузера, у вигляди пар ключ/значення[18]) записується jwt token в якому містяться e-mail користувача та його id(порядковий ідентифікатор в базі данних).

Після авторизації користувач отримує доступ до всього функціоналу, щоб скористатись ним, користувач повинен натиснути на певну кнопку на панелі, яка знаходиться вгорі вікна браузера можна побачити на рисунку 4.6 .

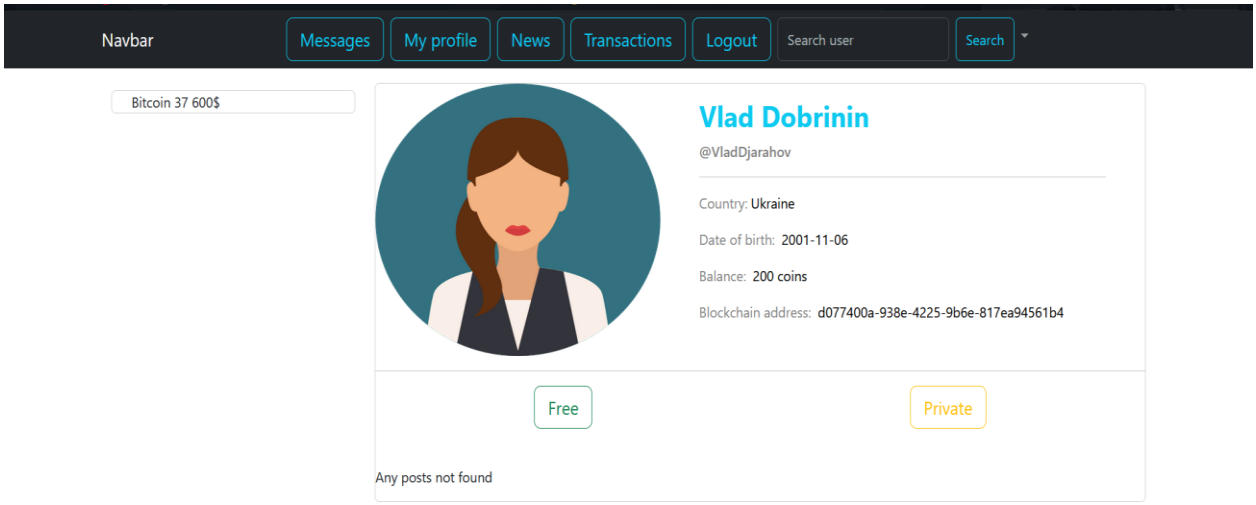

Рисунок 4.6 – Інтерфейс авторизованого користувача.

Так як ми знаходимось на сторінці користувача який автроизувався, ми не можемо підписати користувача самого на себе.

Спробуємо відправити йому певну кількість монет(умовної валюти). Для цього натиснемо на кнопку транзакції. В результаті відкриється сторінка транзакцій, інтерфейс якої можна побачити на рисунку 4.7 .

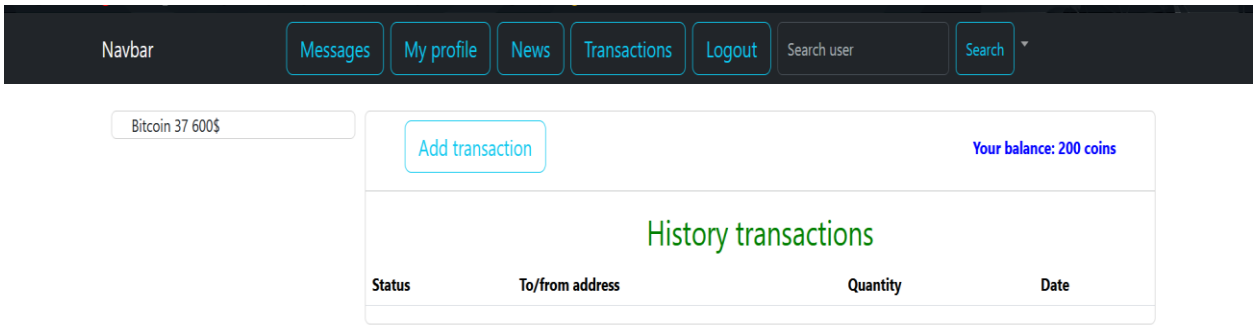

Рисунок 4.7 – Інтерфейс сторінки транзакцій з пустою історією.

Після цього ми можемо натиснути на кнопку “Add transactions” та побачити модальне вікно, яке відкрилось, детальніше на рисунку 4.8 .

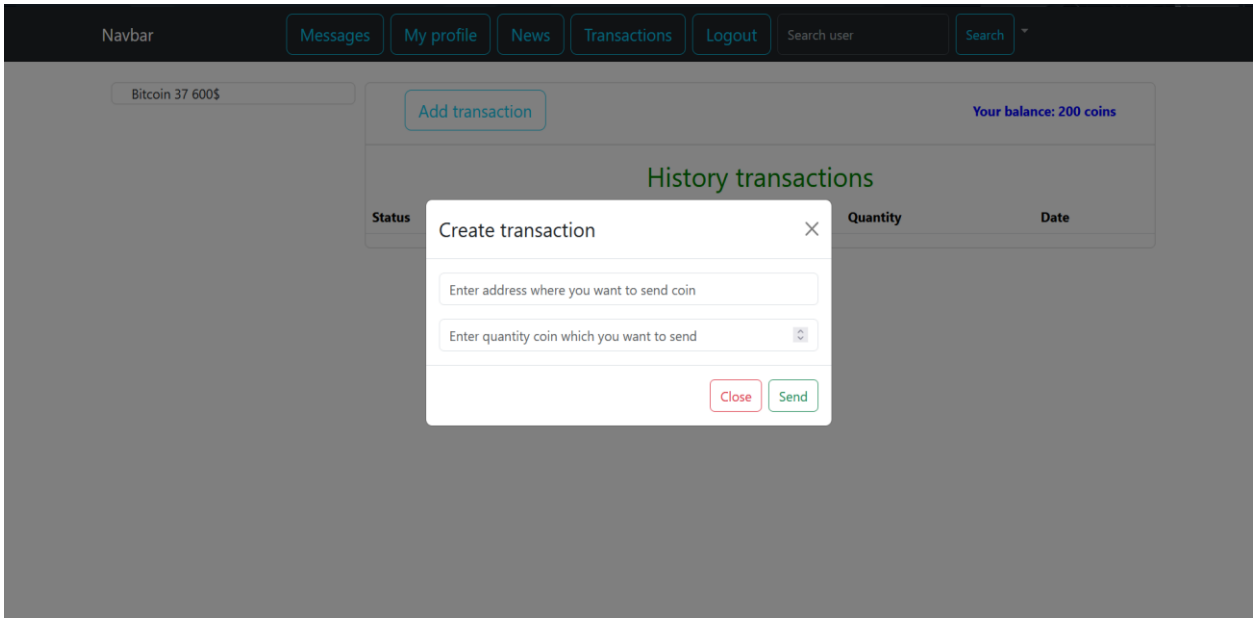

Рисунок 4.8 – Інтерфейс модального вікна для створення транзакції.

Щоб зробити транзакцію, користувач повинен ввести адрес отримувача, можна знайти його на сторінці профілю будь-якого користувача в описі профілю, і ввести кількість монет яку він хоче відправити. Якщо користувач введе кількість більшу ніж він має на балансі, він отримає повідомлення що транзакція не можлива, через недостатню кількість монет на балансі. При успішному відправленні монет, користувач зможе побачити транзакцію в історії транзакцій, що можна побачити на рисунку 4.9 .

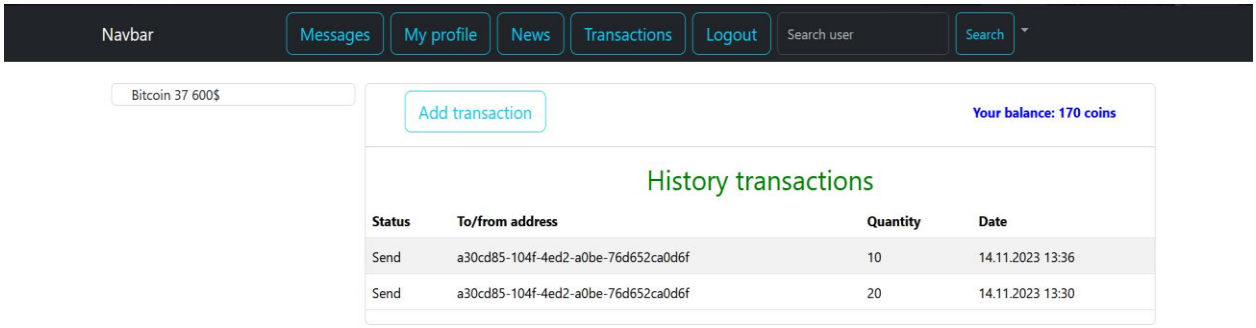

Рисунок 4.9 - Інтерфейс сторінки транзакцій з наповненою історією.

На рисунку 4.9 ми бачимо, що в нас є статус транзакції, тобто відправлена чи отримана, адрес на який вона відправлена, чи з якого отримана, та кількість монет які були в транзакції, а також дати цієї транзакції.

Щоб впевнитися що транзакції дійсно були записані до бази даних, перейдемо до застосунку для управління базами даних PostgresQL, і подивимось поля таблиці “Transactions”(див. рис. 4.10).

	id [PK] integer	id_sender text	id_resiver text	count bigint	date timestamp with time zone
1+	1	d077400a-938e-4225-9b6e-817ea94561b4	a30cd85-104f-4ed2-a0be-76d652ca0d6f	20	2023-11-20 21:10:03.317+02

Рисунок 4.10 – Запис в БД в таблиці “Transactions”.

4.3 Тестування методом навантаження

Навантажувальне тестування використовується для оцінки роботоздатності та стійкості системи, програмного забезпечення або додатків під час роботи в умовах збільшеного навантаження.

Його проводять з метою визначення того, як система або програмне забезпечення працюють під великим навантаженням та які можливості масштабування вони мають.

Це важливо для бізнес-застосувань, де багато користувачів можуть одночасно використовувати систему. Якщо система не може витримати навантаження, це може призвести до збоїв, падінь продуктивності або недоступності системи. Для тестування ми будемо використовувати застосунок “Jmeter”.[17]

Для того щоб відправляти параметри у форматі Json нам потрібно додати HTTP Header Manager, в якому вказати параметр такий як Content-Type зі значенням application/json, рисунок 4.11.

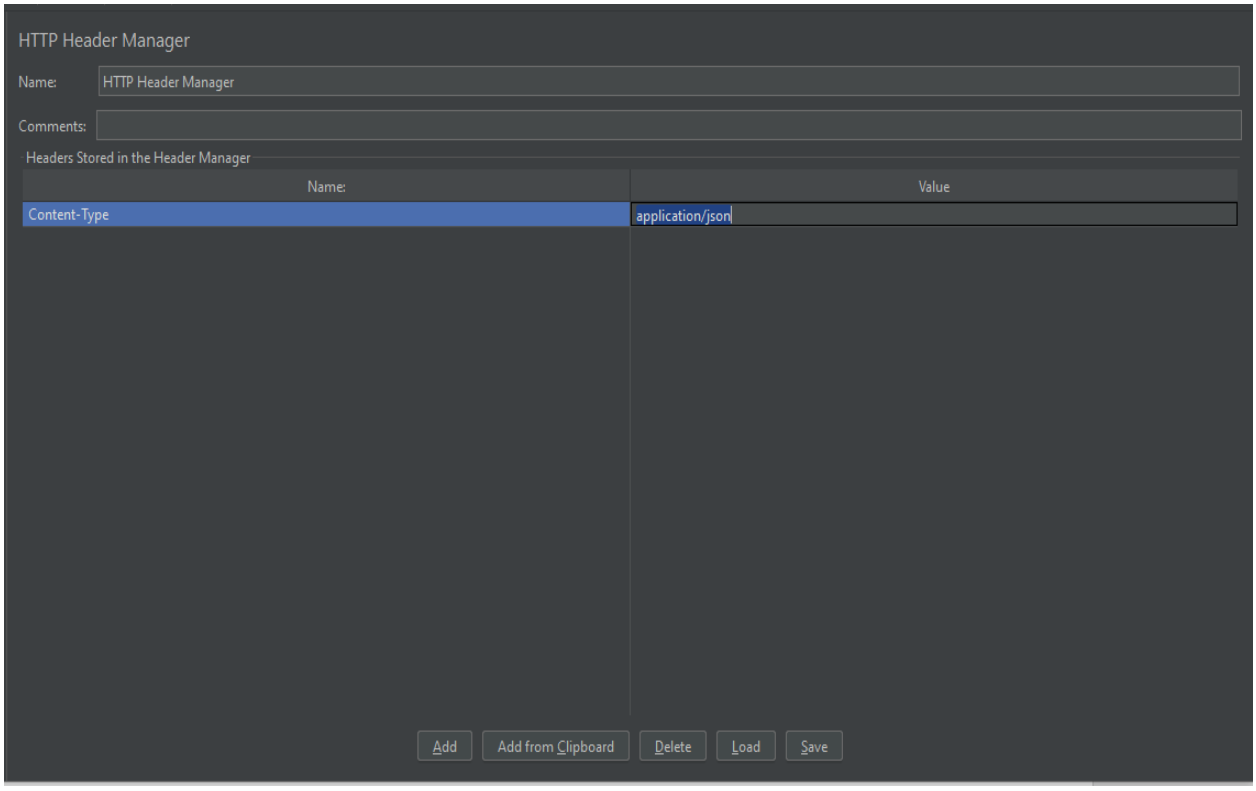

Рисунок 4.11 - Параметри HTTP Header Manager.

Для створення запитів, ми повинні встановити данні необхідні для запиту, наприклад тіло запиту, маршрут або якщо це потрібно додати файл, наприклад із зображенням, детальніше на рисунку 4.12

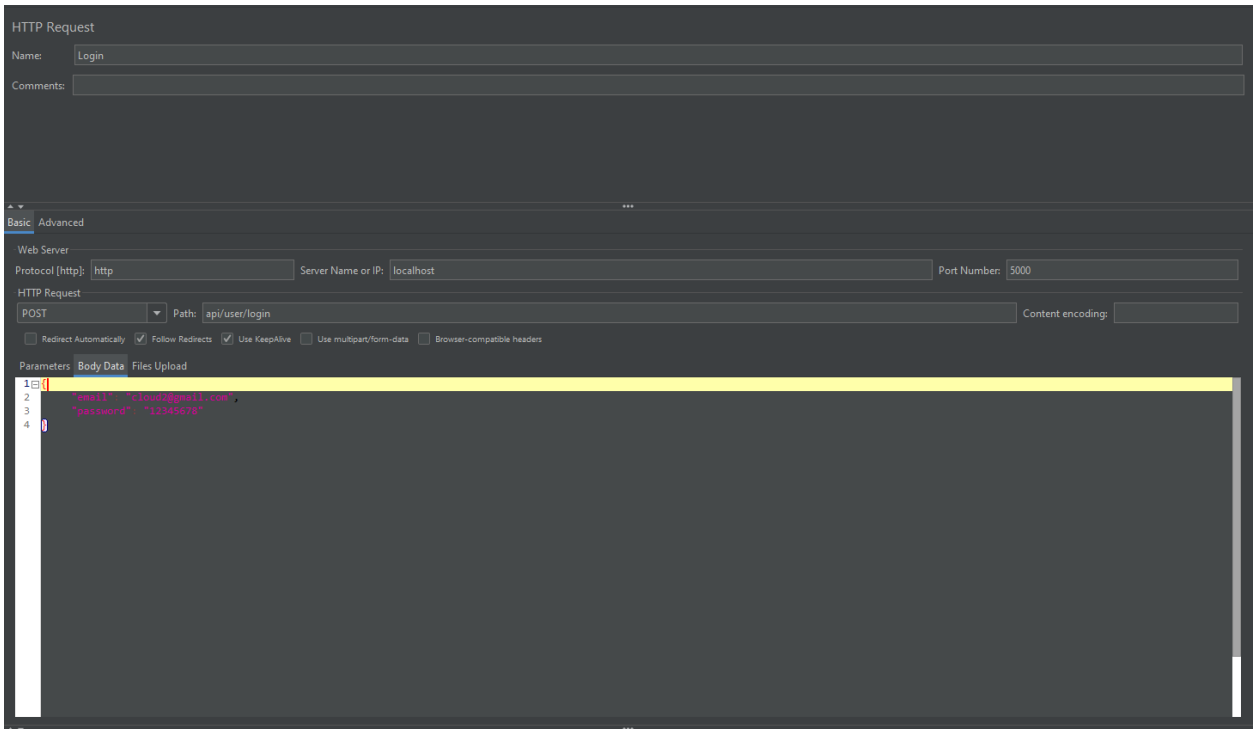

Рисунок 4.12 – Приклад запиту на авторизацію.

Роздивимось запит на створення транзакції детальніше, для цього треба подивитись на рисунок 4.13 .

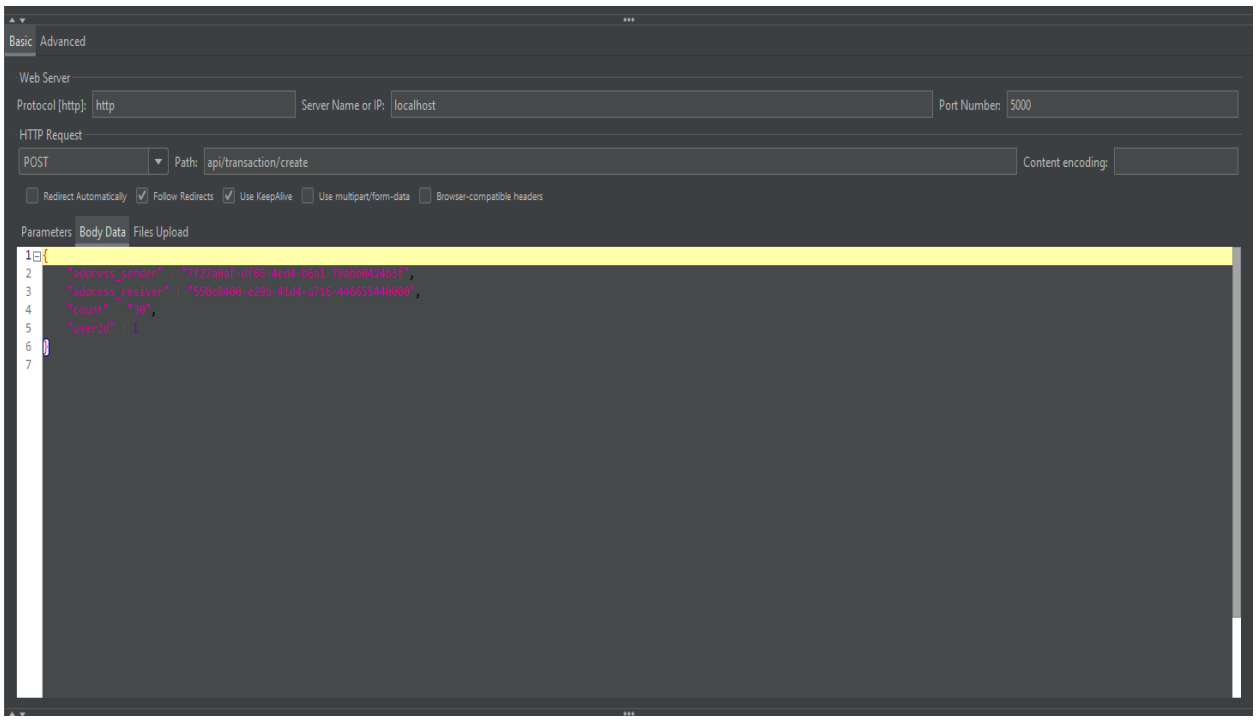

Рисунок 4.13 – Приклад запиту на створення транзакції.

У запиті ми вказуємо параметри, які при відправці з кліенту автоматично можуть підставитися, наприклад “address_sender”, завдяки чому користувач у клієнті вносить тільки кількість монет, та адресу отримувача.

Проведемо тестування серверу, за наступнимим параметрами (див. рис. 4.14).

Рисунок 4.14 – Параметри для проведення тесту.

Де параметр “Action to be taken after a Simpler error” означає, як тесту вчинити в разі виникнення помилки.

Параметр “Number of Threads (users)” означає кількість віртуальних користувачів, яких ми очікуємо підключити до сервера.

Параметр “Ramp-up period (seconds)” вказує на період нарощування, який вказує JMeter, скільки часу потрібно для «нарощування» до повної кількості вибраних потоків. Якщо використовується 10 потоків, а період нарощування становить 100 секунд, тоді JMeter знадобиться 100 секунд, щоб запустити та запустити всі 10 потоків. Кожен потік розпочнеться через 10 (100/10) секунд після початку попереднього.

Інший параметр “Loop count” вказує на кількість ітерацій для кожного користувача відповідно до наших вимог, які ми можемо встановити.

Після встановлення необхідних параметрів, розпочнемо проведення тесту.

У результатах тесту за встановленими параметрами(див. рис 4.14), ми отримали наступні результати (див. рис. 4.15) .

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Login	200	2320	1004	3708	755.67	0.00%	37.7/sec	16.67	8.79	453.0
Create post	200	3276	2449	4190	498.41	0.00%	25.7/sec	11.90	570.04	474.7
Create comment	200	2773	1658	3963	708.54	0.00%	28.8/sec	11.17	7.21	396.5
Create transaction	200	1631	924	2165	292.11	0.00%	42.4/sec	20.41	15.28	493.0
Get transactions by a...	200	5946	2484	9172	2167.71	0.00%	16.3/sec	2864.51	3.97	179779.4
Get posts by user id	200	4450	2050	7074	1538.64	0.00%	15.9/sec	14.17	3.13	914.0
Get comments by po...	200	2249	628	5532	725.80	0.00%	22.0/sec	564.12	4.33	26295.0
TOTAL	1400	3235	628	9172	1789.75	0.00%	55.0/sec	1603.08	186.18	29829.4

Рисунок 4.15 – Результати проведення першого тесту.

В результатах можна побачити таблицю, де є різні колонки:

- Label – це назва запиту;
- Samples – це кількість разів, скільки запитів було зроблено;
- Avarage - середнє арифметичне для всіх відповідей (сума всіх часів / кількість);
- Min - мінімальний час відгуку (мс);
- Max - Максимальний час відгуку (мс);
- Error - відсоток невдалих тестів;
- Throughput - скільки запитів за секунду обробляє ваш сервер. Чим більше, тим краще.
- Avg. Bytes - середній розмір відповіді.

Збільшимо кількість віртуальних користувачів до 700.

В результаті (див. рис. 4.16) :

- Кількість помилок так і залишилась нульовою;
- Кількість оброблених запитів в секунду спала;
- Максимальна кількість часу на обробку запиту стала більшою, через те, що об’єм бази данних при кожному запиті на запис стає більшим;
- Мінімальна кількіть часу на обробку запиту стала в деяких запитах більшою, а в деяких меншою, це можна зв’язати з навантаженням на сервер, затосунками які відкриті на комп’ютері.

Label	# Samples ↓	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Login	700	6965	15	16645	5070.28	0.00%	8.3/sec	3.64	1.97	452.0
Create post	700	5391	73	9581	2526.01	0.00%	7.9/sec	3.64	174.41	474.5
Create comment	700	4276	341	9548	1853.52	0.00%	7.5/sec	2.93	1.89	397.7
Create transaction	700	4533	581	8479	1616.38	0.00%	7.0/sec	3.36	2.51	493.9
Get transactions by a...	700	17368	4332	36686	9273.26	0.00%	5.1/sec	1630.87	1.25	325512.9
Get posts by user id	700	13826	2332	23062	6710.78	0.00%	4.9/sec	2.31	0.97	481.0
Get comments by po...	700	12878	2550	23262	5850.24	0.00%	4.5/sec	760.01	0.90	171071.3
TOTAL	4900	9319	15	36686	7274.96	0.00%	29.7/sec	2065.13	100.41	71269.0

Рисунок 4.16 – Результати проведення другого тесту.

Висновки за розділом 4

За результатом проведення тестів на навантаження можна сказати, що застосунок при навантаженні працює в штатному режимі, тобто без помилок, але час на обробку одного запиту зростає, що є погано, але логічно.

При проведенні тестів взаємодії клієнту з сервером застосунок відпрацював без помилок, всі дані з запитів на запис, були успішно записані до бази даних. При запитах на отримання даних, клієнт отримав все без помилок.

Запуски серверів та клієнта були виконані без помилок, тобто сервери та клієнт успішно запустилися.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи я проаналізував можливості застосування технології блокчейн та концепції web 3.0, для соціальних мереж. В результаті розробки і аналізу я прийшов до наступних висновків:

- Концепція web 3.0, а саме семантичне павутиння, дуже гарно підходить під потреби соціальних мереж, в питанні утримки зацікавленості користувача, через те, що за допомогою метаданих користувач отримує контент, який дійсно йому цікавий. Тобто він буде отримувати тематичні публікації відповідно своїм інтересам. Якщо аналізувати ці інтереси за допомогою штучного інтелекту, то користувачу не буде потреби самому вносити інформацію до списку своїх інтересів. А користувачу який робить публікації не буде потреби вносити метадані вручну. Таким же чином можна аналізувати публікації які робить користувач, і рекомендувати йому користувачів зі схожими інтересами, та метаданими публікацій.
- Технологія блокчейн, забезпечує дуже надійний захист транзакцій, через свої методи відправлення та отримання даних, а дані можуть бути будь які, як інформація про власність на будь-який об’єкт, до криптовалют. Також серед важливих показників безпеки блокчейн технології, можна виділити те, що застосування криптографічних методів дозволяють досягти дуже великих показників безпеки даних. А сама технологія блокчейн робить транзакції прозорими. Але використовувати блокчейн для відправки повідомлень або створення постів не є доцільним, тому що дані про минулі блоки в блокчейні може отримати будь хто, а видалити чи оновити блоки неможливо за суттю роботи блокчейн технології.

Під час виконання кваліфікаційної роботи була розроблена комп’ютерна модель соціальної мережі, а саме її програмна реалізація, розроблені вимоги, проаналізовані технології для виконання програмної реалізації, і зроблений

вибір на користь технологій які найбільше підходять для реалізації поставленої задачі. Також модель була протестована.

Серед перспектив подальших досліджень теми, можна виділити такі:

- Використання штучного інтелекту для аналізу публікацій, що на мою думку має призвести до покращень користувацького досвіду та утримання людини в соціальній мережі.
- Використання штучного інтелекту для аналізу інтересів користувачів на основі їх дій.

Узагальнюючи, можна сказати що концепція web 3.0 та технологія блокчейн має перспективи використання в соціальних мережах. А швидкий розвиток блокчейн та web 3.0 може створити нові можливості, та перспективи для соціальних мереж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tim Berners-Lee History of the Web URL: https://webfoundation.org/about/vision/history-of-the-web/ (Дата звернення: 01.11.2023);

2. Will Kenton What Is Web 2.0? Definition, Impact, and Examples URL: https://www.investopedia.com/terms/w/web-20.asp (Дата звернення: 02.11.2023);

3. Werner Vermaak What is Web 3.0? Decentralized Internet Explained URL: https://coinmarketcap.com/academy/article/what-is-web-3-0 (Дата звернення: 04.11.2023);

4. George Lawton, “Semantic Web” URL: https://www.techtarget.com/searchcio/definition/Semantic-Web (Дата звернення: 02.11.2023);

5. Web 3: Що це таке та як це змінить Інтернет? URL: https://aboutcrypto.info/ua/education-ua/web-3-shho-cze-take-ta-yak-cze-zminit-internet/ (Дата звернення: 04.11.2023);

6. Евгенія Смерека. Що таке web 3.0, чому великі компанії в нього інвестують і коли настане епоха інтернету нового покоління. URL: https://mc.today/uk/web-3-0/ (Дата звернення: 05.11.2023);

7. Binance academy. What Is Web 3.0 and Why Does It Matter? URL:https://academy.binance.com/en/articles/the-evolution-of-the-internet-web-3-0-explained (Дата звернення: 05.11.2023);

8. “What is blockchain technology?” URL: https://www.ibm.com/topics/blockchain (Дата звернення: 08.11.2023);

9. Julija Golosova, Andrejs Romanovs, “The Advantages and Disadvantages of the Blockchain Technology” URL: https://ieeexplore.ieee.org/document/8592253 (Дата звернення: 08.11.2023);

10. Дмитро Караченцов Що таке мнемонічна фраза? Словник BIP39 URL:https://lwallet.com.ua/22505-що-таке-мнемонічна-фраза-словник-bip39/?lang=uk (Дата звернення: 09.11.2023);

11. “The sapien network” URL: https://www.sapien.network/ (Дата звернення:10.11.2023);

12. “Steem An incentivized, blockchain-based, public content platform” URL: https://steem.com/SteemWhitePaper.pdf (Дата звернення: 10.11.2023);

13. “What Is SOLA Network?” URL: https://sola.network/vision/ (Дата звернення: 10.11.2023);

14. “Indorse Code Validation Network” URL: https://indorse-staging-bucket.s3.amazonaws.com/Indorse+2.0+Light+Paper.pdf (Дата звернення: 10.11.2023);

15. Craig Larman Applying UML and patterns Видавництво: Addison Wesley Professional, 1997. 736с.

16. Greg Lim Beginning Node.js, Express and MongoDB 2019. 155с.

17. Ishan Gaba JMeter Load Testing: A Comprehensive Guide URL: https://www.simplilearn.com/tutorials/jmeter-tutorial/jmeter-load-testing (Дата звернення: 18.11.2023);

18. LocalStorage, sessionStorage URL: https://javascript.info/localstorage . (Дата звернення: 15.11.2023);

19. Кравченко П. Блокчейн і децентралізовані системи : навч. посібник у 3 ч. Ч. 1 / П. Кравченко, Б. Скрябін, О. Дубініна. – Харків : ПРОМАРТ, 2019. – 452 с.

20. Daniel Drescher Blockchain Basics: A Non-Technical Introduction in 25 Steps, Apress, 2018, - 56с.

21. Antony Lewis The Basics of Bitcoins and Blockchains, Mango, 2018. - 152 с.

22. Winston Ma, Ken Huang Blockchain and the Web: Building the Cryptocurrency, Privacy, and Security Foundations of the Metaverse, Wiley ,2022. – 102 с.

23. Imran Bashir Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, Packt Publishing; 2nd Revised edition, 2018. 30-87 c.

24. Jared Tate Andrew Knapp Blockchain 2035: The Digital DNA of Internet 3.0, BlueShed LLC; 1st edition, 2019, 45-105 c.

25. Massimo Ragnedda, Giuseppe Destefanis Blockchain and Web 3.0 Social, Economic, and Technological Challenges, 2019, 20-68 c.

26. G C Cooke Web3: The End of Business as Usual; The impact of Web 3.0, Blockchain, Bitcoin, NFTs, Crypto, DeFi, Smart Contracts and the Metaverse on Business Strategy, Web3 Book Ltd, 2022, 25-75 c.

27. Bernard Marr The Future Internet: How the Metaverse, Web 3.0, and Blockchain Will Transform Business and Society, Wiley; 1st edition, 2023, 150-200 c.

28. Kary Oberbrunner, Lee Richter Blockchain Life: Making Sense of the Metaverse, NFTs, Cryptocurrency, Virtual Reality, Augmented Reality, and Web3, Ethos Collective, 2022, 25-40 c.

29. Alex Tapscott Web3: Charting the Internet's Next Economic and Cultural Frontier, Harper Business, 2023, 128-143c.

30. Elad Elrom The Blockchain Developer: A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects, Apress; 1st ed. Edition, 2019, 200-287 c.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **15 – Автоматизація та приладобудування**
Спеціальність: **151 – «Автоматизація та комп'ютерно-інтегровані технології»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Медведенко Владислав Олексійович
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн»

керівник роботи Булавін Дмитро Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

3. Перелік питань, які потрібно розробити)

1) Аналіз предметної області та постановка задачі.
2) Огляд мов розробки та баз даних для створення моделі.
3) Побудова схеми бази даних.
4) Створення освновних алгоритмів для розробки моделі.
5) Програмна реалізація комп'ютерної моделі соціальної мережі.
6) Тестування та дослідження ефективності роботи програмної моделі.

2

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз предметної області та постановка задачі.	08.12.2022 - 05.03.2023
2	Огляд мов розробки та баз даних для створення моделі.	05.03.2023 - 01.04.2023
3	Побудова схеми бази даних.	01.04.2023 - 29.05.2023
4	Створення освновних алгоритмів для розробки моделі.	30.05.2023 - 24.06.2023
5	Програмна реалізація комп'ютеної моделі соціальної мережі.	24.06.2023 – 24.08.2023
6	Тестування та дослідження ефективності роботи програмної моделі	25.08.2023 – 31.08.2023
7	Оформлення пояснювальної записки.	01.09.2023 - 27.10.2023
8	Представлення кваліфікаційної роботи керівнику та рецензенту.	28.10.2023 - 28.11.2023

5. Дата видачі завдання 08.12.2022

Студент В.О. Медведенко _____
ініціали, прізвище підпис

Керівник роботи Д.О. Булавін _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«01» грудня 2023 р.

ІНДИВІДУАЛЬНЕ ТЕХНІЧНЕ ЗАВДАННЯ.

Технічне завдання

на розробку програмного виробу «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн**»**

1.	Введення	1.1) Назва: Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн. 1.2) Галузь застосування: статистика, комп’ютерні технології.
2.	Підстава для розробки	2.1) Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології 2.2) Завдання на кваліфікаційну роботу магістра № 4101-5/3197 від «10» листопада 2023 року (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1) Мета розробки: Аналіз використання транзакцій умовної валюти за допомогою технології блокчейн. Та результат використання мета даних для різних функцій комп’ютерної моделі. 3.2) Вихідні дані розробки: комп’ютерна модель, з можливістю обміну віртуальної валюти, та можливістю отримувати персоналізований контент
4.	Технічні вимоги до програмного виробу	4.1) Вимоги до функціональних характеристик: можливість зарєструватися, авторизуватися, відправляти приватні повідомлення, створювати публікації, підписуватись на повідомлення інших людей, обмінюватися умовною валютою. 4.2) Вимоги до надійності: забезепчувати працездатніть при умовах середнього навантаження від користувачів, шифрування даних. 4.3)Вимоги до умов експлуатації: встановити будь-який браузер. 4.4) Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання, ПК, тощо 4.5) Вимоги до інформаційної та програмної сумісності: немає 4.6) Вимоги до маркування та упаковки: немає

<table>
<tr><td></td><td></td><td colspan="2">4.7) Вимоги до транспортування і зберігання: на звичайних носіях інформації
4.8) Спеціальні вимоги: немає.</td></tr>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td colspan="2">Програмною документацією до виробу «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Алгоритм технології блокчейн представити як Додаток В до пояснювальної записки до кваліфікаційної роботи.
3) Тестування і методику випробувань розробленого виробу представити у розділі 6 до пояснювальної записки до кваліфікаційної роботи.</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td colspan="2">Програмною документацією до виробу «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн» вважати:
1) Розроблений вироб повинен бути достатньо спроможним обробляти 700 користувачів без помітної втрати швидкості.
2) Мінімальна затримка відповіді від сервера повинна не перевищувати 3 секунди.</td></tr>
<tr><td rowspan="6">7.</td><td rowspan="6">Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>12.12.2022–05.01.2023</td><td>1.Аналіз предметної області.</td></tr>
<tr><td>18.12.2022–05.01.2023</td><td>2.Аналіз існуючого науково-методичного апарату розробки і використання блокчейн технології і концепції web 3.0 .</td></tr>
<tr><td>06.01.2023–20.01.2023</td><td>3.Аналіз існуючого програмного забезпечення.</td></tr>
<tr><td>21.01.2023–20.02.2023</td><td>4.Аналіз матеріально-технічної бази для розробки комп’ютерної моделі соціальної мережі.</td></tr>
<tr><td>21.02.2023–10.03.2023</td><td>5. Аналіз матеріально-технічної бази для розробки алгоритму роботи блокчейн</td></tr>
</table>

		11.03.2023–01.06.2023 01.06.2023–20.06.2023 21.06.2023–01.09.2023 01.09.2023–12.10.2023 13.10.2023–23.10.2023 23.10.2023–05.11.2023	6. Розробка комп'ютеної моделі соціальної мережі. 7. Тестування розробленої комп'ютерної моделі соціальної мережі. 8. Розробка алгоритму блокчейн технології. 9. Впровадження метамови як головної частини концепції web 3.0 . 10. Тестування розробленої з використнням технології блокчейн та концепції web 3.0 комп'ютерної моделі соціальної мережі. 11.Представлення кваліфікаційного проекту керівнику кваліфікаційної роботи та рецензенту.
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику на CD-R компакт-диску.	

Виконавець
студент групи КУ- 61
Медведенко В.О. _____

Замовник
к. т. н., доцент,
Булавін Д.О. _____

Додаток В

Затверджую

_____

«01» грудня 2023 р.

Програма і методика випробувань
програмного виробу

«Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн»

1 Об'єкт випробувань

1. Назва програмного виробу : «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн»
2. Галузь застосування : комп’ютерні технології, аналіз поведінки користувачів, технологія блокчейн.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

1. Працювати на основних операційних системах: Windows, Linux, MacOS;
2. Вимоги до надійності;
3. Передбачити захист від некоректних дій користувача;
4. Сумісність з іншими програмними продуктами;
5. Зменшити об’єм програмного коду необхідного для створення веб-додатків;
6. Бути легко розширюваною;
7. Елементи програми повинні бути ізольовані одне від одного для зменшення їх впливу на роботи програми під час редагування програмного коду;
8. Вимоги до складу і параметрів технічних засобів;
9. Вимоги до маркування та упаковки (не висуваються);
10. Вимоги до транспортування і зберігання (не висуваються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Комп'ютерна модель соціальної мережі на основі концепції web 3.0 та технології блокчейн» вважати:

1. Програмною документацією щодо розроблюваного програмного продукту вважати:

2. справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
3. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
4. Текст програми(представити як Додаток Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідний проект та середа розробки Microsoft Visual Studio Code. Встановлена node.js останньої версії на комп'ютер. Встановлена PostgresQL база даних. Та будь-який браузер.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);
- випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.
2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
3. Перевірку комплектності складу технічних і програмних засобів.
4. Методику проведення перевірок на 1 етапі випробувань.
5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;
2. перевірку ступеня виконання функціональних вимог до програми;
3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

Додаток Г

Затверджую

_____

«01» грудня 2023 р.

Програмна реалізація

```
const express = require('express');
const bodyParser = require('body-parser');
const { Sequelize, DataTypes } = require('sequelize');
const SHA256 = require('crypto-js/sha256');
const crypto = require('crypto')
const cors = require('cors');
class Transaction {
  constructor(fromAddress, toAddress, amount) {
    this.fromAddress = fromAddress;
    this.toAddress = toAddress;
    this.amount = amount;
  }
}
class Block {
  constructor(index, timestamp, transactions, previousHash = '') {
    this.index = index;
    this.timestamp = timestamp;
    this.transactions = transactions;
    this.previousHash = previousHash;
    this.hash = this.calculateHash();
    this.nonce = 0;
  }
  calculateHash() {
    return SHA256(
```

```
        this.index +
        this.timestamp +
        JSON.stringify(this.transactions) +
        this.previousHash +
        this.nonce
      ).toString();
    }
    mineBlock(difficulty) {
      while (this.hash.substring(0, difficulty) !== Array(difficulty + 1).join('0'))
{
        this.nonce++;
        this.hash = this.calculateHash();
      }
      console.log(`Block mined: ${this.hash}`);
    }
  }
  class Blockchain {
    constructor() {
      this.chain = [this.createGenesisBlock()];
      this.difficulty = 2;
      this.pendingTransactions = [];
      this.miningReward = 100;
      this.nodes = {};
      this.users = {};
      this.secretKey = process.env.BLOCKCHAIN_SECRET_KEY
      // Sequelize setup
      this.sequelize = new Sequelize('Blockchain_Base', 'postgres', 'admin', {
        host: 'localhost',
        dialect: 'postgres',
      });
```

```
this.BlockModel = this.sequelize.define('Block', {
  index: {
    type: DataTypes.INTEGER,
    allowNull: false,
    primaryKey: true,
  },
  timestamp: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  transactions: {
    type: DataTypes.JSONB,
    allowNull: false,
  },
  previousHash: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  hash: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  nonce: {
    type: DataTypes.INTEGER,
    allowNull: false,
  },
});
this.UserModel = this.sequelize.define('User', {
  address: {
    type: DataTypes.STRING,
```

```
      allowNull: false,
      primaryKey: true,
    },
    balance: {
      type: DataTypes.INTEGER,
      allowNull: false,
    },
  });
  this.TransactionModel = this.sequelize.define('Transaction', {
    id: {
      type: DataTypes.INTEGER,
      allowNull: false,
      autoIncrement: true,
      primaryKey: true,
    },
    fromAddress: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    toAddress: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    amount: {
      type: DataTypes.INTEGER,
      allowNull: false,
    },
  });
  this.setupDatabase();
}
```

```
getLatestBlock() {
  return this.chain[this.chain.length - 1];
}
minePendingTransactions(miningRewardAddress) {
  const rewardTransaction = new Transaction(
    null,
    miningRewardAddress,
    this.miningReward
  );
  this.pendingTransactions.push(rewardTransaction);
  const newBlock = new Block(
    Date.now(),
    this.pendingTransactions,
    this.getLatestBlock().hash
  );
  newBlock.mineBlock(this.difficulty);
  console.log('Block successfully mined!');
  this.chain.push(newBlock);
  this.pendingTransactions = [];
  return newBlock;
}
async setupDatabase() {
  try {
    await this.sequelize.sync();
    console.log('Database synchronized.');
  } catch (error) {
    console.error('Error setting up database:', error);
  }
}
createGenesisBlock() {
```

```
  return new Block(0, '01/01/2022', [], '0');
}
async saveBlockToDatabase(block) {
  try {
    await this.BlockModel.create({
      index: block.index,
      timestamp: block.timestamp,
      transactions: JSON.stringify(block.transactions),
      previousHash: block.previousHash,
      hash: block.hash,
      nonce: block.nonce,
    });
    console.log('Block saved to database:', block);
  } catch (error) {
    console.error('Error saving block to database:', error);
  }
}
async createUserInDatabase(req, res) {
  try {
    let address = crypto.randomUUID();
    console.log(address)
    await this.UserModel.create({
      address: address,
      balance: 200,
    });
    res.status(200).json(address);
  }catch (e) {
    res.status(400).json('Error creating user in database:' + e);
  }
}
```

```
async saveTransactionToDatabase(transaction) {
  try {
    await this.TransactionModel.create({
      fromAddress: transaction.fromAddress,
      toAddress: transaction.toAddress,
      amount: transaction.amount,
    });
    console.log('Transaction saved to database:', transaction);
  } catch (error) {
    console.error('Error saving transaction to database:', error);
  }
}
async getAllUsersFromDatabase(req,res) {
  try {
    const users = await this.UserModel.findAll();
    res.status(200).json('All users from database:', users.map(user =>
user.balance));
  } catch (error) {
    res.status(400).json('Error fetching users from database:', error);
  }
}
createTransaction(req, res) {
  const { fromAddress, toAddress, amount } = req.body;
  if (!fromAddress || !toAddress || !amount) {
    return res.status(400).json({ error: 'Invalid transaction data' });
  }
  this.users[fromAddress].balance = 200;
  const transaction = new Transaction(fromAddress, toAddress, amount);
  // Check if the sender has sufficient balance
```

```
            const senderBalance = this.users[fromAddress] ?
this.users[fromAddress].balance : 0;
            if (senderBalance < amount) {
              return res.status(400).json({ error: 'Insufficient funds' });
            }
            const secretKey = this.secretKey
            this.createTransaction(transaction, secretKey);
            this.saveTransactionToDatabase(transaction);
            // Update user balances
            this.users[fromAddress].balance -= amount;
            this.users[toAddress] = this.users[toAddress] || { balance: 0 };
            this.users[toAddress].balance += amount;
            return res.status(201).json({ message: 'Transaction created successfully'
});
          }
          mineBlock(req, res) {
            const { miningRewardAddress } = req.body;
            if (!miningRewardAddress) {
              return res.status(400).json({ error: 'Mining reward address is required'
});
            }
            this.minePendingTransactions(miningRewardAddress);
            return res.status(200).json({ message: 'Block mined successfully' });
          }
          getBlockchain(req, res) {
            return res.status(200).json({ blockchain: this.chain });
          }
        }

        const myBlockchain = new Blockchain();
```

```
const app = express();
const port = 4000;
app.use(bodyParser.json());
app.use(cors());
app.use(express.json())
app.post('/user', (req, res) => myBlockchain.createUserInDatabase(req, res));
app.get('/users', (req, res) => myBlockchain.getAllUsersFromDatabase());
app.post('/transaction', (req, res) => myBlockchain.createTransaction(req,
res));
app.post('/mine', (req, res) => myBlockchain.mineBlock(req, res));
app.get('/blockchain', (req, res) => myBlockchain.getBlockchain(req, res));
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```