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

Факультет: **ННІ Каразінський банківський інститут**

Кафедра: **Інформаційних технологій та математичного моделювання**

Спеціальність: **122 Комп'ютерні науки**

Освітня програма: **Комп'ютерні науки та інформаційні технології в бізнесі**

Група: АК-21М **денна форма навчання**

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«СИСТЕМА МОНІТОРИНГУ ТА АНАЛІЗУ НОВИХ ТОКЕНІВ У БЛОКЧЕЙН-МЕРЕЖАХ»

ЗА НАКАЗОМ № 0210-05/1946 ВІД 29 ВЕРЕСНЯ 2023 РОКУ

здобувача вищої освіти **Кандиби Владислава Віталійовича**

Робота допущена до захисту в ЕК
протокол кафедри ІТММ № 2 від 02.12.2025р.

В.о. завідувача кафедри ІТММ
PhD
_____ **Д.М. Ковальчук**

Науковий керівник
к.т.н.
_____ **А.В. Рогов**

м. Харків 2025 р.

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

Факультет навчально-науковий інститут "Каразінський банківський інститут"
Кафедра інформаційних технологій та математичного моделювання
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки та інформаційні технології в бізнесі

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**
Підпис ініціали, прізвище

“__”_____2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Кандибі Владиславу Віталійовичу
(прізвище, ім’я, по батькові студента)

1. Тема роботи: Система моніторингу та аналізу нових токенів у блокчейн-мережах.

керівник роботи Рогов Андрій Володимирович, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “__” _____ 2025року № 4601-5/335

2. Строк подання студентом роботи _____

3. Перелік питань, які потрібно розробити:

У розділі 1: Зібрати теоретичні відомості про блокчейн-технології.

У розділі 2: Провести аналіз блокчейн-технологій та розробити метод оцінки токенів.

У розділі 3: Розробити систему моніторингу нових токенів у блокчейн мережі.

4.План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної магістерської роботи
2	Затвердження плану і завдання кваліфікаційної магістерської роботи
3	Здача кваліфікаційної магістерської роботи керівнику
4	Підпис кваліфікаційної магістерської роботи керівника
5	Підпис кваліфікаційної магістерської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної магістерської роботи
7	Захист кваліфікаційної магістерської роботи

Дата видачі завдання _____

Студент _____ Кандиба В.В.
підпис ініціали, прізвище

Керівник роботи _____ Рогов А.В.
підпис ініціали, прізвище

РЕФЕРАТ

НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ

«СИСТЕМА МОНІТОРИНГУ ТА АНАЛІЗУ НОВИХ ТОКЕНІВ У БЛОКЧЕЙН-МЕРЕЖАХ»

Канбиди Владислава Віталійовича

Кваліфікаційна магістерська робота містить: 82 сторінки, 10 таблиць, список літератури із 10 найменувань.

Об’єктом дослідження є інформаційна система, призначена для автоматичного моніторингу та початкового аналізу нових токенів у блокчейн-мережі Solana.

Предметом дослідження є методи та алгоритми збору даних (підписка на події, стріми транзакцій), а також критерії та метрики для оцінки інвестиційної привабливості новостворених токенів у мережі Solana.

Метою роботи є створення програмного забезпечення, яке здійснює автоматичний моніторинг появи нових токенів у блокчейн-мережі Solana , забезпечує збір перших транзакцій, пов'язаних із цими токенами, та виконує їх початковий аналіз за визначеними критеріями для об'єктивної оцінки потенційної інвестиційної привабливості.

Для досягнення цієї мети визначено такі завдання:

1. Зібрати теоретичні відомості про блокчейн-технології.
2. Провести аналіз блокчейн-технологій та розробити метод оцінки токенів.
3. Розробити систему моніторингу нових токенів у блокчейн мережі, оцінити її продуктивність.

Актуальність теми магістерської роботи зумовлена високою динамікою та волатильністю ринку нових токенів у сфері DeFi та необхідністю розробки інструментів, здатних забезпечити трейдерів оперативною та об'єктивною кількісною оцінкою активів, щойно з’явилися, мінімізуючи ризики маніпуляцій.

За результатами дослідження розроблено працездатний програмний модуль, здатний в автоматичному режимі оцінювати початковий попит токенів та формувати висновок про їхній потенціал. Створена архітектура дозволяє масштабувати систему та інтегрувати більш складні методи, такі як машинне навчання, для оптимізації вагових коефіцієнтів Score.

Практичне значення роботи полягає у створенні прототипу аналітичної системи, яка завдяки використанню GRPC-протоколу, має конкурентну перевагу за швидкістю перед стандартними RPC-рішеннями. Отримані результати можуть бути використані в комерційних цілях для автоматизованого прийняття рішень на високочастотних ринках, а також у подальших наукових дослідженнях архітектур моніторингу блокчейну.

Одержані результати можуть бути використані в різних сферах. У сфері освіти вони застосовані при вивченні сучасних технологій роботи з високочастотними потоковими даними блокчейн-мереж та для демонстрації принципів побудови складних аналітичних моделей скорингу на основі метрик. У комерційних проєктах створений працездатний прототип може служити основою для розробки високопродуктивних аналітичних інструментів, що дозволяють оперативно оцінювати інвестиційний потенціал токенів у момент їхньої появи, забезпечуючи конкурентну перевагу на ринках із високою частотою транзакцій.

КЛЮЧОВІ СЛОВА: SOLANA, GRPC, МОНІТОРИНГ У РЕАЛЬНОМУ ЧАСІ, SCORING-АЛГОРИТМ, PUMP.FUN, АНАЛІЗ ТРАНЗАКЦІЙ, LOW LATENCY, NODE.JS, АРХІТЕКТУРА СИСТЕМИ.

ABSTRACT

AT QUALIFICATION BACHELOR WORK

"SYSTEM FOR MONITORING AND ANALYSIS OF NEW TOKENS IN BLOCKCHAIN NETWORKS"

Kandyba Vladyslav

The qualification master's thesis contains: 82 pages, 10 tables, a list of references of 10 items.

The object of the study is an information system designed for automatic monitoring and initial analysis of new tokens in the Solana blockchain network.

The subject of the study is methods and algorithms for data collection (event subscription, transaction streams), as well as criteria and metrics for assessing the investment attractiveness of newly created tokens in the Solana network.

The purpose of the work is to create software that automatically monitors the appearance of new tokens in the Solana blockchain network, provides collection of the first transactions associated with these tokens, and performs their initial analysis according to specified criteria for an objective assessment of potential investment attractiveness.

To achieve this goal, the following tasks have been defined:

1. To collect theoretical information about blockchain technologies.
2. To conduct an analysis of blockchain technologies and develop a method for evaluating tokens.
3. To develop a monitoring system for new tokens in the blockchain network, to assess its performance.

The relevance of the topic of the master's thesis is due to the high dynamics and volatility of the market for new tokens in the DeFi sphere and the need to develop tools capable of providing traders with an operational and objective quantitative assessment of assets that have just appeared, minimizing the risks of manipulation.

According to the results of the study, a workable software module was developed that is capable of automatically assessing the initial demand for tokens and forming a conclusion about their potential. The created architecture allows you to scale the system and integrate more complex methods, such as machine learning, to optimize the Score weighting coefficients.

The practical significance of the work lies in creating a prototype of an analytical system that, thanks to the use of the GRPC protocol, has a competitive advantage in speed over standard RPC solutions. The results obtained can be used for commercial purposes for automated decision-making in high-frequency markets, as well as in further scientific research into blockchain monitoring architectures.

The results obtained can be used in various fields. In the field of education, they are applicable in the study of modern technologies for working with high-frequency streaming data of blockchain networks and for demonstrating the principles of building complex analytical scoring models based on metrics. In commercial projects, the created working prototype can serve as the basis for the development of high-performance analytical tools that allow for the prompt assessment of the investment potential of tokens at the moment of their appearance, providing a competitive advantage in markets with a high transaction frequency.

KEYWORDS: SOLANA, GRPC, REAL-TIME MONITORING, SCORING ALGORITHM, PUMP.FUN, TRANSACTION ANALYSIS, LOW LATENCY, NODE.JS, SYSTEM ARCHITECTURE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ І ТЕРМІНІВ

DeFi – децентралізовані фінанси

PoW – Proof of Work

PoH – Proof of History

PoS – Proof of Stake

PoSA – Proof of Staked Authority

BSC – Binance Smart Chain

ERC-20 – стандарт для створення взаємозамінних токенів на блокчейні Ethereum

SPL – Solana Program Library

TPS – Transactions Per Second

Merkle Root – це унікальний криптографічний хеш, який є "відбитком" всіх транзакцій у блоці даних.

Blockchain – це децентралізована цифрова база даних, яка зберігає інформацію у вигляді ланцюжка блоків, що захищені криптографічно.

GRPC - Google Remove Produce Call.

ВСТУП

Стрімкий розвиток децентралізованих фінансів (DeFi) та блокчейн-технологій спричинив появу тисяч нових цифрових активів. Мережа Solana на сьогодні є однією з провідних платформ, що пропонує високу пропускну здатність та низькі комісії, стимулюючи активний запуск нових токенів, зокрема через спеціалізовані платформи на кшталт Pump.fun. У цьому динамічному середовищі, де токени можуть з'являтися та досягати значної капіталізації за лічені хвилини, інвестори та трейдери зіштовхуються з проблемою швидкого та ефективного аналізу цих активів. Ручний моніторинг є затрудненим через швидкість ринку, а затримка в оцінці потенціалу призводить до втрати інвестиційної переваги.

Таким чином, актуальність теми дослідження полягає у необхідності розробки автоматизованих, високошвидкісних інструментів, здатних здійснювати моніторинг у режимі реального часу та виконувати початковий аналіз нових токенів одразу після їх появи. Створення такої системи є критичним для своєчасного визначення інвестиційно привабливих активів у сфері DeFi та трейдингу.

Метою дипломного проекту є створення програмного забезпечення, яке здійснює автоматичний моніторинг появи нових токенів у блокчейн-мережі Solana , забезпечує збір перших транзакцій, пов'язаних із цими токенами, та виконує їх початковий аналіз за визначеними критеріями для об'єктивної оцінки потенційної інвестиційної привабливості.

Для досягнення поставленої мети було визначено такі завдання:

1. Дослідити теоретичні основи функціонування блокчейн-мереж, зокрема архітектуру та принципи роботи мережі Solana.
2. Визначити механізм створення та розповсюдження токенів через платформу Pump.fun.
3. Розробити та реалізувати модуль підписки на події для автоматичного визначення появи нових токенів.

4. Створити модуль збору та обробки даних для фіксації перших 100 транзакцій для кожного нового активу.
5. Розробити алгоритм аналізу зібраних транзакцій на основі ключових метрик, таких як: активність купівлі/продажу, обсяг ліквідності, кількість унікальних покупців, співвідношення купівель до продажів та час між транзакціями.
6. Реалізувати систему згідно з вимогами та забезпечити структуроване збереження результатів.

Об’єктом дослідження є інформаційна система, призначена для автоматичного моніторингу та початкового аналізу нових токенів у блокчейн-мережі Solana.

Предметом дослідження є методи та алгоритми збору даних (підписка на події, стріми транзакцій), а також критерії та метрики для оцінки інвестиційної привабливості новостворених токенів у мережі Solana.

Практична цінність роботи полягає у створенні функціональної системи моніторингу та аналізу нових токенів, яка може бути безпосередньо використана у сфері DeFi та трейдингу. Розроблена система автоматично надає короткий аналітичний висновок, що є цінним інструментом для прийняття рішень щодо інвестицій або подальшого глибокого дослідження активу. У майбутньому система має потенціал до вдосконалення через інтеграцію машинного навчання та створення Telegram-бота для реальних сповіщень.

У розділі 1 викладено теоретичні основи блокчейн-технологій та платформи Solana.

Розділ 2 присвячено аналізу механізмів створення токенів та метрикам для їх оцінки.

Розділ 3 містить опис проектування та архітектури системи моніторингу.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ БЛОКЧЕЙН-ТЕХНОЛОГІЙ

1.1. Поняття та архітектура блокчейн-мереж

1.1.1. Визначення та фундаментальні принципи

Блокчейн (Blockchain) — це децентралізований, розподілений та криптографічно захищений реєстр, організований як неперервний ланцюг блоків, що містять записи про транзакції. Його основне призначення полягає у забезпеченні незмінного та прозорого обліку даних без необхідності у довіреному центральному посереднику [2].

Фундаментальні принципи, на яких ґрунтується технологія:

1. Децентралізація: дані зберігаються не на одному центральному сервері, а розподіляються між тисячами незалежних комп'ютерів (вузлів) по всьому світу. Це усуває єдину точку відмови та забезпечує стійкість системи до цензури, атак і втручання.
2. Незмінність (Immutability): після того, як блок доданий до ланцюга, інформацію в ньому неможливо змінити чи видалити. Ця властивість забезпечується криптографічним хешуванням та зв'язком блоків. Будь-яка спроба змінити старий блок вимагатиме перерахунку хешів усіх наступних блоків, що є обчислювально нездійсненним для великих мереж.
3. Прозорість (Transparency): всі транзакції у публічному блокчейні є відкритими та можуть бути перевірені будь-яким учасником мережі. При цьому, ідентифікація користувачів відбувається через криптографічні адреси (публічні ключі), забезпечуючи псевдонімність замість повної анонімності.
4. Розподілений консенсус: механізм, за допомогою якого всі учасники мережі досягають згоди щодо поточного стану реєстру та послідовності

додавання нових блоків. Це критично важливо для підтримки єдиної, достовірної версії правди в мережі.

1.1.2. Структура блоку та криптографічні механізми

Кожен блок у ланцюзі складається з двох основних частин: заголовка блоку (Block Header) та списку транзакцій.

Таблиця 1.1

Частини блоку у блокчейні

Частина Блоку	Елементи та Функція
Заголовок Блоку	Містить метадані. Основні компоненти: Timestamp (час створення), Номер блоку, Хеш попереднього блоку (зв'язок), Хеш поточного блоку (унікальний ідентифікатор), та Корінь Меркла (Merkle Root).
Список Транзакцій	Набір підтверджених та перевірених транзакцій, які відбулися з моменту створення попереднього блоку.

Корінь Меркла (Merkle Root) - це єдиний хеш, отриманий шляхом послідовного хешування всіх хешів транзакцій у блоці. Він дозволяє ефективно та швидко перевірити, чи була певна транзакція включена у блок, без необхідності завантажувати весь його вміст.

Криптографічне хешування використовується для створення унікального "цифрового відбитка" даних. Хеш-функції (наприклад, SHA-256) є односторонніми, тобто за хешем неможливо відновити початкові дані, і

будь-яка найменша зміна у вхідних даних призводить до абсолютно іншого хешу, що гарантує цілісність інформації.

1.2. Криптографічна основа та механізми консенсусу

1.2.1. Асиметрична криптографія та цифрові підписи

Безпека та автентифікація в блокчейні забезпечується за допомогою асиметричної криптографії (криптографія з відкритим ключем).

- Публічний ключ (Public Key): відкритий для всіх учасників мережі. Використовується для отримання коштів (адреса гаманця) та перевірки цифрового підпису.
- Приватний ключ (Private Key): секретний ключ, який зберігається лише власником. Використовується для створення цифрового підпису, що підтверджує дозвіл власника на здійснення транзакції.
- Цифровий підпис: створюється шляхом хешування даних транзакції та шифрування цього хешу приватним ключем. Цей підпис доводить: а) що транзакція ініційована власником приватного ключа; б) що дані транзакції не були змінені після підписання.

1.2.2. Основні механізми консенсусу

Механізми консенсусу є основою функціонування будь-якого блокчейну, забезпечуючи довіру в недовіреному середовищі [4]. Їхня ключова роль полягає у забезпеченні довіри, цілісності та узгодженості даних у середовищі, де немає єдиного центрального органу і, відповідно, немає необхідності в довірі між усіма учасниками мережі.

Механізм консенсусу — це набір правил і протоколів, які дозволяють розподіленій мережі вузлів досягти єдиної спільної угоди щодо правдивості та порядку транзакцій, які записуються у наступний блок.

Таблиця 1.2

Механізми консенсусу

Механізм	Опис	Переваги	Недоліки
Proof-of-Work (PoW)	Валідатори (майнери) змагаються у вирішенні ресурсоємної математичної задачі. Переможець додає блок і отримує нагороду.	Висока безпека, перевірена часом (Bitcoin).	Високе енергоспоживання, низька пропускна здатність, високі комісії.
Proof-of-Stake (PoS)	Право на створення блоку визначається розміром частки (кількістю заблокованих токенів), яку учасник (валідатор) вніс у стейкінг.	Енергоефективність, швидша фіналізація блоків.	Ризик централізації (акумуляція великої частки в одних руках).
Delegated Proof-of-Stake (DPoS)	Користувачі делегують свою частку (голоси) обмеженій кількості обраних валідаторів, які і створюють блоки.	Дуже висока швидкість, низькі комісії.	Висока централізація, оскільки лише невелика група контролює мережу.
Proof-of-History (PoH)	Унікальний механізм Solana. Це криптографічний годинник, який доводить послідовність і час подій, що оптимізує PoS, усуваючи необхідність у великій кількості повідомлень для синхронізації часу.	Надзвичайна швидкість, висока пропускна здатність.	Високі вимоги до апаратного забезпечення валідаторів.

1.3. Смарт-контракти та їх роль у DeFi

1.3.1. Визначення та життєвий цикл смарт-контракту

Смарт-контракт (Smart Contract) — це комп'ютеризований протокол транзакцій, який автоматично виконує умови угоди. Він зберігається, перевіряється та виконується на блокчейні, забезпечуючи надійність без посередників [5].

Життєвий цикл смарт-контракту:

1. Розробка: створення коду на спеціалізованій мові (наприклад, Solidity для Ethereum, Rust для Solana).
2. Розгортання: код публікується у блокчейні. Контракт отримує свою унікальну адресу і стає незмінним.
3. Виконання: контракт чекає на виконання заздалегідь визначених умов (наприклад, надходження коштів на його адресу або виклик певної функції). при виконанні умов, код автоматично запускає записані дії.
4. Взаємодія: користувачі та інші смарт-контракти взаємодіють з ним, надсилаючи транзакції для виклику його функцій, маючи змогу передавати потрібні аргументи.

Смарт-контракти є наріжним каменем DeFi, забезпечуючи можливість створення:

- Децентралізованих бірж (DEX): автоматичне здійснення обміну токенів без необхідності у звичайних ордербуках через технологію Automated Market Making.
- Систем кредитування: управління заставою, видача та повернення позик.
- Стейкінг і фармінг прибутковості: автоматичний розподіл винагород користувачам, які блокують свої токени в протоколі для підтримки його ліквідності або безпеки мережі.

1.3.2. Детальна структура та життєвий цикл блокчейн-транзакції

Транзакція починається з ініціації та закінчується її фіналізацією в блоці.

Етапи обробки транзакції:

1. Ініціація: користувач створює транзакцію, зазначаючи: адресу отримувача (або контракту), суму, комісію та дані (якщо це виклик контракту).
2. Підписання: транзакція підписується приватним ключем відправника.
3. Передача: транзакція відправляється у мережу, потрапляючи у Mempool (пул непідтверджених транзакцій).
4. Валідація (Підтвердження): валідатори перевіряють цифровий підпис, наявність коштів у відправника та відповідність транзакції правилам мережі.
5. Формування Блоку: валідатор, обраний механізмом консенсусу, збирає перевірені транзакції з Mempool у новий блок.
6. Фіналізація: блок додається до ланцюга. Транзакція вважається підтвердженою і незворотною.

1.4. Токенізація активів та стандарти токенів

1.4.1. Взаємозамінні токени (FT) та стандарт ERC-20 / SPL-Token

Взаємозамінні токени (Fungible Tokens) — це цифрові активи, які ідентичні за вартістю та властивостями. Вони є основою для криптовалют та більшості токенів, які використовуються для платежів чи управління [8].

Стандарт ERC-20 на Ethereum став еталоном для створення взаємозамінних токенів. Він вимагає реалізації шести основних функцій у смарт-контракті:

- totalSupply(): загальна кількість токенів в обігу.
- balanceOf(address): перевірка балансу за адресою.

- transfer(address, uint256): переказ токенів.
- transferFrom(address, address, uint256): переказ токенів від імені іншого користувача (використовується біржами).
- approve(address, uint256): надання дозволу іншому контракту/адресі на використання певної кількості токенів.
- allowance(address, address): перевірка дозволеної суми.

У мережі Solana аналогом ERC-20 є Протокол SPL-токенів (Solana Program Library Token Program). Це не смарт-контракт у звичному розумінні, а програма, розгорнута командою Solana, яка централізовано управляє всіма токенами в мережі, забезпечуючи високу ефективність і єдиний інтерфейс.

1.4.2. Невзаємозамінні токени (NFT) та стандарт ERC-721

Невзаємозамінні токени (Non-Fungible Tokens, NFT) — це унікальні цифрові активи, які підтверджують право власності на об'єкти (цифрове мистецтво, ігрові предмети).

Стандарт ERC-721:

- Кожен токен має унікальний ідентифікатор (Token ID).
- Цей стандарт включає функції для:
    - визначення власника (ownerOf(uint256)).
    - передачі права власності (transferFrom).
    - зберігання метаданих (посилання на зображення, опис).

1.5. Огляд та порівняння блокчейн-платформ рівня 1 (Layer 1)

Вибір платформи для розробки системи моніторингу нових токенів є критичним. Він повинен ґрунтуватися на балансі між швидкістю, вартістю та рівнем децентралізації. Порівняємо три найпопулярніших блокчейна по базовим метрикам[3].

1.5.1. Ethereum

Таблиця 1.3

Характеристики блокчейну Ethereum

Характеристика	Ethereum
Пропускна здатність (TPS)	~15-30
Механізм Консенсусу	Proof-of-Stake
Комісії	Високі, непередбачувані (залежать від завантаженості мережі)
Децентралізація	Найвища, тисячі вузлів
Проблема	Трилема Блокчейну: Ethereum історично жертвував масштабованістю заради безпеки та децентралізації. Це призвело до виникнення рішень Layer 2.

1.5.2. Binance Smart Chain (BNB Chain)

Таблиця 1.4

Характеристики блокчейну BSC

Характеристика	BNB Chain
Пропускна здатність (TPS)	~60-140
Механізм Консенсусу	PoSA (Proof of Staked Authority)
Комісії	Дуже низькі
Проблема	Централізація: Використовує лише 21 валідатора, що робить його швидким, але залежним від обмеженої кількості сторін, контрольованих Binance.

1.5.3. Solana

Solana була розроблена з метою вирішення проблеми масштабованості блокчейну без використання рішень Layer 2, що дозволяє їй працювати без додаткових рівнів обробки та ускладнення архітектури. Це робить мережу чи не ідеальною для високочастотних додатків, таких як моніторинг появи нових токенів, арбітражні боти, аналітичні системи та трейдингові платформи, для яких швидкість підтвердження транзакцій і мінімальні затримки є критично важливими [1].

1.5.3.1. Архітектурні інновації Solana

1. Proof-of-History (PoH): PoH є криптографічним годинником, який створює послідовний, впорядкований запис подій. Цей механізм дозволяє валідаторам обробляти транзакції без постійного обміну інформацією про час.
2. Tower BFT (Optimized PoS): механізм консенсусу, який використовує PoH як "годинник", значно скорочуючи час, необхідний для досягнення згоди, дозволяючи мережі працювати набагато швидше.
3. Sealevel (Паралельна обробка транзакцій): це механізм, який дозволяє Solana паралельно виконувати транзакції, що не перетинаються. Наприклад, якщо дві транзакції впливають на різні смарт-контракти чи рахунки, вони можуть оброблятися одночасно. Це максимізує ефективність використання ресурсів.
4. Gulf Stream (Протокол пересилання транзакцій): дозволяє відправникам надсилати транзакції валідаторам до того, як поточний блок буде повністю фіналізовано. Це зменшує час очікування для транзакцій і підвищує швидкість підтвердження.
5. Фактично Solana є першим real-time блокчейном.

1.5.3.2. Обґрунтування вибору Solana для проєкту

Для цілей даної дипломної роботи (створення системи моніторингу та початкового аналізу нових токенів) критично важливими є:

- Можливість фіксувати появу нового токена і збирати перші 100 транзакцій за лічені секунди.
- Необхідність обробляти велику кількість даних та потенційно виконувати багато запитів і підписок без високих операційних витрат.

Саме Solana пропонує унікальне поєднання технологій (PoH, Sealevel), які роблять її найкращою платформою для реалізації високочастотного моніторингового рішення.

1.4. Методика дослідження та інструментарій

Дослідження та практична реалізація кваліфікаційної роботи базувалися на комплексному застосуванні наукових методів та сучасного програмного інструментарію, які забезпечили необхідну точність, достовірність і ефективність отриманих результатів.

1.4.1. Методи дослідження

У процесі виконання роботи були використані такі наукові методи:

- Метод системного аналізу: застосовувався для формування цілісного уявлення про об'єкт дослідження — блокчейн-мережу Solana та архітектуру децентралізованих додатків (dApps). Дозволив визначити ключові компоненти системи моніторингу (модуль підписки, модуль аналізу, модуль зберігання) та встановити функціональні зв'язки між ними.
- Метод порівняльного аналізу: використовувався в Розділі 1 для критичної оцінки архітектур провідних блокчейн-платформ (Ethereum,

BNB Chain, Solana) та обґрунтування вибору Solana як найбільш оптимальної платформи для високошвидкісного моніторингу.

- Метод наукового узагальнення та абстрагування: застосовано для формування теоретичних основ, визначення понятійного апарату (транзакція, смарт-контракт, SPL-токен) та класифікації метрик, необхідних для оцінки інвестиційної привабливості нових активів.
- Метод математичного та статистичного аналізу: є основою для розробки Модуля Аналізу (Розділ 3). Використовувався для:
    - кількісної оцінки активності токенів (розрахунок середнього часу між транзакціями, дисперсія обсягів торгів).
    - формування коефіцієнта співвідношення купівель/продажів для визначення ринкового інтересу.
    - визначення порогів для класифікації гаманців (боти чи органічні покупці) на основі їхньої активності.
- Метод структурного проєктування: використовувався на етапі проєктування (Розділ 3) для візуалізації архітектури програмного забезпечення, взаємодії модулів та послідовності виконання алгоритмів.

1.4.2. Інструментарій та інформаційна база

Для реалізації поставлених завдань, збору, обробки даних та розробки програмного продукту використовувалися такі засоби:

1. Програмне середовище та Мова програмування:
    a. Node.js / TypeScript: використано як основне середовище для розробки системи завдяки його ефективності для асинхронної обробки великих потоків даних у режимі реального часу.
    b. Бібліотека @solana/web3.js: основний інструмент для взаємодії з RPC та створення WebSocket-з'єднань.
2. Інформаційна база (Джерела даних):

   a. Solana RPC: це основний канал отримання вхідних даних, що забезпечують моніторинг нових транзакцій і подій (поява токенів на Pump.fun) у режимі реального часу.
   b. GRPC-з'єднання: для отримання високошвидкісних, менш цензурованих потоків даних від спеціалізованих вузлів (Yellowstone) з метою мінімізації затримки.
   c. Відкриті блокчейн-експлорери: використовувалися для верифікації та порівняння результатів, отриманих розробленою системою.
3. Засоби зберігання та обробки даних:
   a. JSON-формат: використовувався як основний формат для збереження та обміну структурованих результатів аналізу.
   b. Текстові редактори та IDE: Visual Studio Code — для розробки, відлагодження та документування коду.

РОЗДІЛ 2
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДИ ОЦІНКИ НОВИХ ТОКЕНІВ

2.1. Механізми взаємодії з блокчейном Solana: RPC та WebSockets

Розробка високоефективної системи моніторингу вимагає глибокого розуміння протоколів, які забезпечують отримання даних з блокчейн-мережі Solana в режимі, максимально наближеному до реального часу.

2.1.1. Архітектура обробки даних у Solana

На відміну від багатьох блокчейнів, Solana використовує модель, орієнтовану на акаунти (Accounts). Акаунти є єдиним сховищем стану та даних у мережі. Кожна транзакція — це набір атомарних Інструкцій, які викликають Програми (Programs), що є аналогом смарт-контрактів [1].

- Транзакція Solana: кожна транзакція чітко оголошує список усіх аккаунтів, з якими вона взаємодіятиме. Це фундаментально для механізму Sealevel, який дозволяє паралельно виконувати транзакції, що не зачіпають одні й ті самі аккаунти. Успішна транзакція завершується включенням до блоку та зміною стану аккаунтів.
- Валідатори та Вузли RPC: валідатори є критично важливими вузлами, які не лише підтверджують блоки, але й надають зовнішній доступ до даних через протоколи Remote Procedure Call (RPC) та WebSockets. Швидкість та надійність цих вузлів безпосередньо впливає на затримку даних.
- Програми (Smart Contracts / Programs): програми в Solana визначають логіку обробки транзакцій і правила взаємодії з аккаунтами. Саме вони виконуються середовищем Sealevel у відповідь на інструкції, отримані в транзакціях, змінюючи стан даних у визначених аккаунтах. Ефективність та оптимізація цих програм безпосередньо впливають на

швидкість виконання операцій, споживання обчислювальних ресурсів і загальну пропускну здатність мережі.

2.1.2. Використання RPC для синхронного запиту даних

RPC (Remote Procedure Call) — це стандартний протокол, який використовується для запиту інформації про поточний стан мережі. В контексті Solana RPC виконує роль механізму pull (запит-відповідь) [6].

- Ключові RPC-запити для моніторингу:
  - getSignaturesForAddress: отримання сигнатур транзакцій для акаунту.
  - getParsedTransaction: отримання даних про конкретну транзакцію.
- Обмеження: RPC-запити є синхронними і мають високу затримку (latency) в умовах високого навантаження. Це робить їх непридатними для безперервного моніторингу подій у реальному часі. Система моніторингу використовує RPC лише для додаткової перевірки та отримання метаданих після первинного виявлення події.

2.1.3. WebSockets як основа моніторингу реального часу

WebSockets є двостороннім комунікаційним протоколом, який встановлює постійне з'єднання між клієнтом (нашою системою) та вузлом Solana. Це дозволяє вузлу надсилати дані клієнту одразу після їх появи (push-механізм) [7].

Функції підписки:

- logsSubscribe: критична функція для даного проєкту. Дозволяє системі підписатися на всі логи (events), що генеруються певною програмою (наприклад, адресою програми Pump.fun). Це дозволяє виявити факт створення нового токена, оскільки ця подія відображається у логах транзакції.

- programSubscribe: підписка на зміни будь-якого аккаунта, що контролюється певною програмою (використовується для відстеження змін у пулах ліквідності).
- slotSubscribe: підписка на фіналізацію нових слотів (блоків), що забезпечує найшвидше сповіщення про включення транзакцій у ланцюг.

Використання WS-підписок мінімізує затримку, дозволяючи системі почати збір перших транзакцій токена вже через мілісекунди після його появи в мережі.

2.1.4. Протокол GRPC для мінімізації затримки

Для додаткової оптимізації швидкості та надійності використовується GRPC (Google Remote Procedure Call) [9].

- Переваги GRPC: використовує Protocol Buffers, що робить його більш ефективним і швидким, ніж стандартний JSON RPC. Спеціалізовані провайдери (наприклад, Jito) надають GRPC-інтерфейси, які дають можливість отримати доступ до Mempool (пул очікуючих транзакцій) або навіть до транзакцій, які готуються до включення у блок (так звані Pre-confirmation data).
- Роль у Системі: GRPC служить як додатковий або резервний високошвидкісний канал для отримання даних. Це дозволяє системі конкурувати зі снайперськими ботами, мінімізуючи часове вікно між появою токена та початком його аналізу.

2.2. Протокол SPL-токенів

Моніторинг нових токенів у мережі Solana вимагає розуміння базового стандарту токенів та особливостей їхнього запуску через автоматизовані платформи.

2.2.1. Протокол SPL-токенів (Solana Program Library)

SPL-токен є фундаментальним стандартом взаємозамінних токенів у Solana, аналогом ERC-20 в Ethereum.

- Єдина Токен-Програма: всі SPL-токени в мережі управляються однією глобальною, незмінною Програмою SPL-токенів. Це забезпечує стандартизацію та високу ефективність.
- Аккаунти Токенів (Token Accounts): на відміну від Ethereum, де баланс токена зберігається всередині смарт-контракту, у Solana баланс кожного токена зберігається в окремому акаунті токена. Користувач володіє не токенами безпосередньо, а Аккаунтом Токена, який містить ці токени.

2.2.2. Аналіз механізму створення токенів через Pump.fun

Платформа Pump.fun стала домінуючим механізмом запуску нових спекулятивних токенів у мережі Solana. Її архітектура та логіка є ключовими для системи моніторингу.

1. Автоматизований Лістинг: Pump.fun дозволяє будь-кому запустити токен, інтегруючи автоматизований маркет-мейкер (AMM) та гарантуючи ліквідність.
2. Процес запуску (Deployment):
    a. користувач ініціює створення токена через Програму Pump.fun.
    b. створюється Mint Account та пов'язаний Аккаунт Маркет-Мейкера, який приймає SOL та обмінює його на новий токен.
    c. критичний крок — Збір ліквідності: кожна купівля токена користувачами автоматично додає SOL до пулу ліквідності токена, тоді як продажі зменшують його.

3. Тригер лістингу на PumpFun AMM: Pump.fun автоматично моніторить накопичену ліквідність (SOL). Коли вона досягає встановленого порогу (наприклад, $69 000):
    a. активується транзакція, яка переводить ліквідність на постійний пул.
    b. після цього токен стає доступним для торгівлі на Raydium, і його ліквідність є постійною.
4. Виявлення: система моніторінгу має відстежувати транзакції Програми Pump.fun. Поява у логах конкретних інструкцій (наприклад, Instruction: Initialize) є прямим сигналом про створення нового токена.

2.3. Платформа Pump.fun: Механізм запуску та Bonding Curve

Платформа Pump.fun є децентралізованим додатком (dApp) у мережі Solana, який здійснив революцію у способі запуску нових спекулятивних (мем) токенів. Pump.fun усуває необхідність у початковій ліквідності, покладаючись на Bonding Curve, та автоматизує процес створення пулів для відкритої торгівлі.

2.3.1. Концепція та переваги Pump.fun

Pump.fun дозволяє користувачам створити токен за лічені секунди. Його ключові особливості:

- Автоматизація: усуває необхідність у ручному створенні пулу ліквідності (LP) та його «залочуванні» (блокуванні).
- Bonding Curve (Крива Зв'язку): використовується для початкового ціноутворення токена.
- Гарантія Міграції Ліквідності: платформа гарантує, що вся ліквідність (SOL), яка надходить від покупців, накопичується для подальшої міграції на постійний, відкритий ринок.

2.3.2. Механізм Bonding Curve

Bonding Curve (Крива Зв'язку) — це смарт-контракт, який визначає ціну токена, пов'язуючи її з його пропозицією (кількістю токенів, викуплених з кривої) [10].

1. Алгоритмічне ціноутворення: на Bonding Curve ціна токена є функцією від його поточної пропозиції, що дозволяє торгівлю з гарантованою ліквідністю.
2. Лінійна залежність: кожна наступна купівля підвищує ціну токена, а кожна продаж знижує її. Це забезпечує постійну ліквідність та автоматичне ціноутворення, що стимулює ранніх покупців.
3. Етап Bonding Curve: токен торгується виключно в межах контракту Pump.fun до моменту, поки не буде досягнутий поріг автоматичного переведення на відкритий ринок. Цей етап є найбільш критичним для нашого моніторингу.

2.4. Формування метрик для оцінки інвестиційної привабливості

Оцінка потенціалу токена базується на кількісному аналізі його початкової активності у мережі. Система збирає та детально аналізує перші 100 транзакцій, що дозволяє зафіксувати природну динаміку попиту та пропозиції одразу після запуску і мінімізувати вплив пізніх спекулянтів або маніпулятивних дій великих учасників ринку. Такий підхід дає змогу сфокусуватися саме на «гарячому» старті токена — моменті, коли формуються його перші цінові та поведінкові патерни.

У межах цього аналізу можуть враховуватися обсяг торгів, швидкість появи нових учасників, частота транзакцій, розподіл обсягів між гаманцями, а також характер змін ціни в найперші хвилини існування активу. На основі цих метрик система формує попередню оцінку потенціалу токена та його можливих сценаріїв розвитку в короткостроковій перспективі.

2.4.1. Кількісні та поведінкові метрики

Таблиця 2.1

Метрики оцінки токенів

Метрика	Опис	Методика збору / розрахунку	Значення для аналізу
Обсяг Торгів (Volume)	Загальна сума SOL, що пройшла через перші 100 транзакцій.	Сума всіх SOL, використаних для купівлі-продажу.	Високий обсяг свідчить про значний початковий інтерес.
Кількість Унікальних Гаманців (Unique Buyers)	Число унікальних адрес, які здійснили хоча б одну купівлю токена.	count(DISTINCT Sender Address) для транзакцій купівлі.	Низька кількість (до 10) може вказувати на високу ймовірність маніпуляцій.
Співвідношення Купівель/Продажів (RBS)	Відношення кількості транзакцій купівлі до кількості транзакцій продажу.	RBS = Buy Volume / Sell Volume	RBS > 1 — домінує купівельний тиск. RBS < 1 — домінує продажний тиск.
Середній Час Між Транзакціями	Середній час у мілісекундах між послідовними транзакціями токена.	Середнє арифметичне всіх різниць часу сусудніх транзакцій	Дуже низьке значення може бути індикатором бот-активності.
Концентрація Власності (Top 10)	Відсоток загальної пропозиції токена, що знаходиться у 10 найбільших власників.	Users Balance / Total supply	Висока концентрація (> 70%) несе ризик Rug Pull.

2.4.2. Розробка алгоритму інтегральної оцінки

Для оцінки кожного токену необхідно звести зібрані метрики в єдиний кількісний показник — Score.

Кожна метрика Mi повинна бути приведена до єдиного діапазону [0, 1] за допомогою функції нормалізації:

$$\text{Норм}(M_i) = \frac{M_i - M_{min}}{M_{max} - M_{min}} \tag{2.1}$$

Де Mmin та Mmax — емпірично визначені мінімальні та максимальні значення метрики для токенів у мережі Solana.

На основі аналізу предметної області встановлюються вагові коефіцієнти Wi, що відображають важливість кожної метрики. Сума вагових коефіцієнтів має дорівнювати одиниці.

Оцінка токена (Score) розраховується як зважена сума нормалізованих метрик:

$$\text{Score} = \sum_{i=1}^{n} w_i \cdot \text{Норм}(M_i) \tag{2.2}$$

Результат Score переводиться у якісний висновок:

- Score > 0.7 (Високий): високий органічний інтерес. Рекомендовано для подальшого аналізу.
- Score 0.4-0.69 (Середній): помірна активність. Співвідношення ризику та винагороди нейтральне.
- Score < 0.4 (Низький): низька активність або ознаки маніпуляції. Високий ризик.

2.5. Специфіка ринку нових токенів: ризики та виявлення бот-активності

Критичною частиною аналізу предметної області є розуміння спекулятивної природи ринку нових токенів, що вимагає інтеграції механізмів виявлення ризиків та ботів для захисту від так званих маніпуляцій “Rug Pull”.

2.5.1. Аналіз ризиків "Rug Pull" та маніпуляцій

Rug Pull — це маніпулятивна схема, при якій розробники чи ранні власники токена швидко виводять усю ліквідність, знецінюючи актив.

Індикатори ризику Rug Pull (на момент аналізу перших 100 Tx):

1. Надмірна концентрація: якщо перевищує 70%, це означає, що токен перебуває під контролем вузької групи осіб, які можуть легко вивести ліквідність.
2. Дисбаланс RBS: надто низьке співвідношення (багато продажів) за відсутності значної кількості унікальних покупців, що вказує на швидке скидання токенів ранніми покупцями-снайперами.
3. Метадані токена: можливість зміни адреси, що контролює випуск (Mint Authority), що може бути використано для створення необмеженої кількості токенів і знецінення активу (хоча Pump.fun зазвичай робить це автоматично, це є важливим критерієм для інших методів лістингу).

2.5.2. Ідентифікація бот-активності (Sniping Bots)

На ринку Solana домінують автоматизовані торгові системи (снайперські боти), які відстежують появу токенів через GRPC канали, що і наша система, але діють швидше, ніж люди.

Методи виявлення ботів системою:

1. Аналіз затримки: боти здійснюють транзакції з мінімальною затримкою.

2. Аналіз унікальності гаманців: якщо великий обсяг торгів припадає на малу кількість унікальних адрес, це вказує на використання ботами кількох гаманців для однієї стратегії.
3. Аналіз структури транзакцій: боти часто використовують однакові, "круглі" суми для купівлі (наприклад, рівно 1 SOL), що відрізняє їх від органічних, випадкових покупок.
4. Аналіз інтервалів транзакцій одних і тих же кошельків: існують так звані бот-ферми, які складаються з різних кошельків, але керовані одним користувачем блокчейну.

Токени, ідентифіковані як такі, у яких значну частину активності формують боти, не обов’язково є неякісними або безперспективними. Однак їхній Score має бути скоригований у бік зменшення, оскільки зафіксоване зростання часто має штучний характер і є результатом автоматизованих торгових стратегій, а не органічного інтересу реальних користувачів. Така активність може викривлювати справжню картину попиту та створювати ілюзію високої ліквідності або популярності.

Корекція рейтингу дозволяє знизити вагу маніпулятивних або спекулятивних факторів у загальній оцінці та зосередитися на показниках, що більш точно відображають довгострокову цінність проєкту — зокрема на поведінці унікальних учасників, стабільності транзакцій та поступовому, а не різко-спекулятивному зростанні активності.

2.5.3. Еволюція стратегій: від снайпінгу до паттерн-аналізу

Історично, стратегія снайпінгу (Sniping) домінувала на ринках нових криптовалютних активів, включно з ранніми етапами розвитку Pump.fun.

Снайпінг – це дія, спрямована на миттєву купівлю токена одразу після його запуску, часто протягом перших мілісекунд, за найнижчою можливою ціною. Метою було випередити основну масу покупців, скористатися

алгоритмічним зростанням ціни, зумовленим механізмом Bonding Curve, та продати актив на піку першої хвилі попиту.

Проте, зростання інтересу до блокчейну Solana та поява професійних інфраструктурних рішень призвели до експоненційного зростання конкуренції. На сьогоднішній день традиційний снайпінг, який ґрунтується лише на швидкості відправлення транзакції, став недоступним та нерентабельним для більшості користувачів. Професійні арбітражні боти, що працюють безпосередньо у вузлах валідаторів і з оптимізованим доступом до даних (HFT-стратегії), здатні реагувати набагато швидше, ніж будь-яка зовнішня система чи людина.

Ця зміна парадигми вимагає переходу від стратегії чистої швидкості до стратегії аналізу та селективного входу. Якщо швидкість купівлі більше не є перевагою, критично важливим стає якість і структура початкової активності токена.

Нова, ефективна стратегія полягає у виявленні патернів у перших транзакціях, що свідчать про:

- Органічний попит: наявність великої кількості унікальних трейдерів, що вказує на залученість спільноти, а не одного або двох ботів.
- Низький ризик маніпуляції: відсутність надмірної концентрації токенів у руках обмеженої кількості ранніх покупців.
- Стійкий тренд: домінування купівельного обсягу над продажами.

Таким чином, розроблена в рамках даної роботи система моніторингу та скорингу є відповіддю на еволюцію ринку. Вона дозволяє відійти від сліпого снайпінгу та купувати актив лише після аналізу, що значно підвищує ймовірність успішного вибору та знижує ризики.

Ключова перевага цієї системи полягає у зміні інвестиційної парадигми. Вона дозволяє повністю відмовитися від тактики "сліпого снайпінгу" (або несистематичних, емоційно мотивованих угод) на користь ретельного, глибокого та об'єктивного аналізу потенційних активів.

2.6. Огляд та критичний аналіз існуючих інструментів моніторингу

Для підтвердження практичної цінності розробленої системи необхідно провести критичний аналіз існуючих рішень та виявити їхні недоліки, щоб вирішити їх у нашому проєкті.

2.6.1. Класифікація існуючих інструментів

1. Офіційні Блокчейн-Експлорери (Solscan):
    a. функціональність: надають повну інформацію про блок, транзакції та стани всіх акаунтів, що були змінені під часу створення блоку.
    b. недоліки: непридатні для реального часу. Дані мають високу затримку, вимагають ручного пошуку за адресою токена. Не мають функціоналу автоматичного Score або аналізу Pump.fun.
2. Агрегатори даних та DEX-аналітика (DexTools, Birdeye):
    a. функціональність: надають графіки, обсяги, ліквідність для токенів.
    b. недоліки: моніторинг з запізненням. Це робить їх непридатними для аналізу самого старту токена. Вони також агрегують дані, приховуючи критичні деталі перших транзакцій.
3. Комерційні Снайперські Боти та Telegram-канали:
    a. функціональність: надсилають сповіщення про появу нових токенів.
    b. недоліки: дані надаються у неструктурованому вигляді. Немає аналітичної складової (Score), що вимагає від користувача самостійного аналізу в умовах жорсткого дефіциту часу. Часто працюють із комерційною затримкою (запізненням, щоб у першу чергу скористалися їхні власники).

2.6.2. Конкурентні переваги розробленої системи

Розроблена Система моніторингу та аналізу нових токенів має низку переваг, що усувають ключові недоліки існуючих рішень:

1. Пріоритет на Initial Activity: система орієнтована не на лістинг, а на момент ініціації токена на Pump.fun. Це забезпечує максимальну перевагу в часі.
2. Аналітична Автоматизація: інструменти не надають інтегральної оцінки. Наша система автоматично застосовує Scoring Algorithm до перших 100 транзакцій, генеруючи готовий Базовий Висновок.
3. Виявлення Ризиків: система використовує метрики для ідентифікації ризиків маніпуляцій та бот-активності, що є недоступним у стандартних агрегаторах.
4. Економічна Ефективність: розроблена система може бути запущена на власному сервері, що забезпечує незалежність від дорогих комерційних підписок.

Таким чином, детальний аналіз предметної області підтвердив актуальність та практичну цінність розробки, визначив необхідні канали комунікації з блокчейном (WS/GRPC) та сформував математичний апарат (метрики та алгоритм Score) для реалізації.

РОЗДІЛ 3
ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ СИСТЕМИ МОНІТОРИНГУ НОВИХ ТОКЕНІВ

3.1. Проєктування архітектури та формалізація вимог

Проєктування високоефективної системи моніторингу та аналізу нових токенів у мережі Solana є фундаментальною задачею, яка вимагає ретельного визначення архітектурних рішень та детальної формалізації функціональних і нефункціональних вимог. Цей початковий етап набуває критичного значення через специфіку середовища: Solana є високочастотним блокчейном з надзвичайно високою пропускною здатністю (TPS) і швидким часом фіналізації блоків, що диктує жорсткі вимоги до обробки даних у реальному часі. Таким чином, лише продумана архітектура, здатна забезпечити високу швидкодію, мінімальну затримку та виняткову надійність при захопленні та індексації даних із вузлів мережі, дозволить ефективно функціонувати та надавати актуальну інформацію про новостворені активи, уникнувши втрати критичних подій.

3.1.1. Формалізація функціональних та нефункціональних вимог

Вимоги до розроблюваної системи моніторингу та скорингу були сформовані в результаті ретельного та глибокого аналізу предметної області, чітко орієнтуючись на досягнення двох ключових і взаємодоповнюючих цілей, що є критично важливими для конкуренції на високочастотному ринку: мінімізація затримки та висока точність аналізу. Зокрема, мінімізація затримки є необхідною для забезпечення можливості оперативного виявлення нових активів та використання коротких "вікон можливостей", тоді як точність аналізу є запорукою прийняття обґрунтованих рішень, що мінімізують ризики, гарантуючи, що швидкість системи не буде досягнута ціною надійності та об'єктивності скорингової оцінки.

Функціональні вимоги визначають, що саме система має робити для досягнення мети дослідження.

Таблиця 3.1

Функціональні вимоги

Код	Вимога	Деталізація / Обґрунтування
ФВ1	Моніторинг у реальному часі	Встановлення постійного високошвидкісного потокового з'єднання (GRPC) для безперервного відстеження логів Програми Pump.fun.
ФВ2	Ідентифікація та трекінг	Виявлення специфічних інструкцій, що сигналізують про створення нового токена (Mint Account), та ініціалізація окремого процесу збору даних для цього токена.
ФВ3	Обмежений збір транзакцій	Збір та зберігання перших 100 транзакцій для кожного нового токена. Це забезпечує фокус на початковій активності та обмежує обсяг даних для швидкого аналізу.
ФВ4	Розрахунок метрик	Програмний розрахунок ключових показників: Unique Wallets (кількість унікальних трейдерів), RBS Ratio (співвідношення обсягів купівлі/продажу), Average Interval (середній час між транзакціями) та Concentration of Ownership (концентрація Top 10).
ФВ5	Генерація інтегральної оцінки	Застосування алгоритму зваженого скорингу (Score) до нормалізованих метрик для отримання числового показника інвестиційного потенціалу (від 0 до 1).
ФВ6	Звітність та виведення	Формування фінального структурованого звіту (JSON) з адресою токена, усіма метриками та базовим висновком (Високий, Середній, Низький потенціал).

Нефункціональні вимоги визначають загальну якість, стабільність і надійність роботи системи, що є особливо критично важливим у контексті спекулятивного моніторингу, де навіть незначні затримки, помилки в обробці даних або нестабільність з’єднання можуть призвести до втрати важливої інформації та спотворення результатів аналізу.

Таблиця 3.2

Нефункціональні вимоги

Код	Вимога	Деталізація / Обґрунтування
НФВ1	Висока швидкість	Затримка між включенням транзакції у блокчейн та її отриманням системою повинна бути мінімальною, що вимагає використання GRPC-протоколу.
НФВ2	Надійність та відмовостійкість	Система повинна бути стійкою до розривів з'єднання та високих пікових навантажень (великої кількості транзакцій у слоті), що досягається за допомогою асинхронної обробки даних.
НФВ3	Продуктивність	Збір даних, парсинг та фінальний розрахунок Score для одного токена не повинен впливати на обробку інших токенів.
НФВ4	Технологічний стек	Використання Node.js та TypeScript для забезпечення високої продуктивності вводу-виводу (I/O) та надійної типізації даних.
НФВ5	Масштабованість	Архітектура має дозволяти легке розширення функціоналу, наприклад, додавання нових метрик або інтеграцію бази даних.

3.1.2. Обґрунтування вибору технологічного стеку та архітектури

Для розробки системи було обрано Node.js у поєднанні з TypeScript.

- Node.js: забезпечує високу ефективність для завдань, орієнтованих на ввід-вивід (I/O-bound tasks), таких як робота з мережевими потоками (GRPC Stream) та обробка великої кількості вхідних даних без затримок. Асинхронна, подієво-орієнтована модель Node.js ідеально підходить для безперервного моніторингу в режимі реального часу.
- TypeScript: надбудова над JavaScript, яка надає статичну типізацію. Це критично важливо при роботі зі складними та багатокомпонентними блокчейн-даними (структури транзакцій, лог-події, RPC-відповіді), мінімізуючи помилки, пов'язані з типом даних, та забезпечуючи високу якість коду, що відповідає вимозі НФВ4.
- Ключові бібліотеки:
    - @triton-one/yellowstone-grpc: використовується для встановлення високошвидкісного GRPC-з'єднання (НФВ1).
    - @solana/web3.js: використовується для виконання традиційних RPC-запитів, що неможливо зробити лише через потоковий GRPC.

Система реалізована як монолітний асинхронний застосунок з чіткою модульною декомпозицією. Це забезпечує високу швидкість взаємодії між компонентами та відповідає вимозі ФВ3.

1. Модуль Моніторингу (Subscription Module): відповідає за виконання ФВ1 та ФВ2. Його основним завданням є підтримання постійного GRPC-з'єднання та первинна фільтрація вхідного потоку даних.
2. Модуль Збору та Аналізу (Analysis Module): відповідає за ФВ3, ФВ4 та ФВ5. Обробляє зібрані 100 транзакцій, парсить їх, розраховує всі метрики та застосовує Scoring-алгоритм.
3. Модуль Звітності (Reporting Module): відповідає за ФВ6. Форматує фінальний Score та метрики у структурований звіт для виведення.

3.1.3. Модульна декомпозиція архітектури системи

Для забезпечення високої швидкості обробки потоку даних та підтримки вимог НФВ2 (Надійність) і НФВ3 (Продуктивність), система розроблена на основі модульної подієво-орієнтованої архітектури (Event-Driven Architecture).

Система функціонує як єдиний асинхронний застосунок, де події (нові транзакції) послідовно передаються між трьома основними незалежними модулями.

Процес роботи системи є послідовним конвеєром обробки даних, що включає збір, аналіз та звітність.

Система складається з трьох основних логічних модулів, кожен з яких відповідає за певний етап обробки та виконання конкретних функціональних вимог.

3.1.3.1. Модуль Моніторингу

Цей модуль є фронтальним інтерфейсом системи, відповідальним за комунікацію з блокчейном Solana та первинну фільтрацію даних (ФВ1, ФВ2).

- Основне завдання: Встановлення та підтримка постійного двостороннього GRPC-з'єднання з вузлом.
- Функціонал:
    - ініціалізація GRPC: встановлення високошвидкісного потокового з'єднання.
    - підписка: надсилання запиту на підписку на події програми Pump.fun.
    - ідентифікація та фільтрація: безперервна фільтрація вхідного потоку логів для виявлення події створення нового токена.

- ініціалізація Трекера: при виявленні нового токена, модуль створює трекер, який починає збирати перші 100 транзакцій для цього токена.

3.1.3.2. Модуль Аналізу та Скорингу

Цей модуль є логічним ядром системи, відповідальним за обробку зібраних даних та розрахунок інтегральної оцінки (ФВ3, ФВ4, ФВ5). Він активується після того, як Модуль Моніторингу передасть повний набір з 100 транзакцій.

- Основне завдання: перетворення сирих транзакційних даних на кількісні метрики та застосування Scoring-алгоритму.
- Функціонал:
  - парсинг: декодування логічних транзакцій, визначення ролі трейдера (купівля/продаж) та обсягів SOL/токена.
  - розрахунок метрик: обчислення Unique Traders, RBS Ratio та Average Interval.
  - скоринг: нормалізація метрик та розрахунок фінального Score через зважену суму.

3.1.3.3. Модуль Звітності

Цей модуль відповідає за фінальну стадію обробки та взаємодію з користувачем (ФВ6).

- Основне завдання: генерація структурованого, зрозумілого висновку на основі числового Score.
- Функціонал:
  - конвертація висновку: перетворення числового значення Score (наприклад, 0.82) у якісний висновок («Високий Потенціал», «Низький Потенціал»).

- форматування: складання фінального звіту (JSON-об'єкт або консольна таблиця), що містить адресу токена, всі розраховані метрики та фінальний висновок.
- журналювання: реєстрація події аналізу для подальшого аудиту.

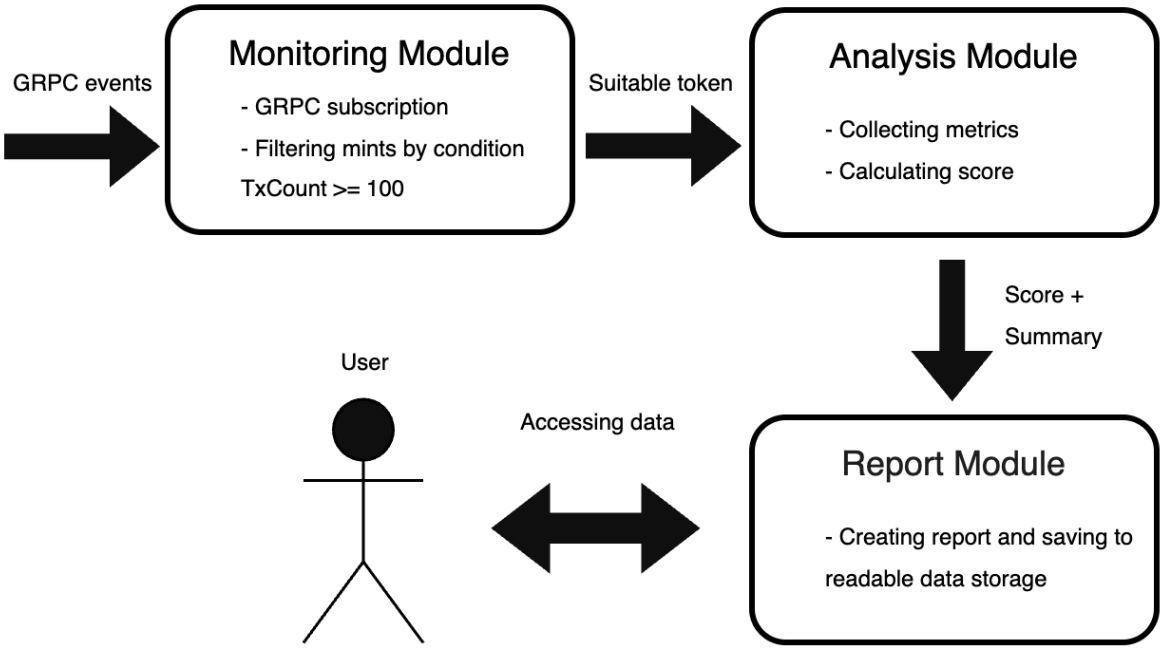

Рисунок 3.1. Схема модулів системи моніторингу

3.2. Розробка модуля моніторингу та збору даних у реальному часі

Модуль моніторингу є абсолютно критичним елементом розроблюваної системи, оскільки він безпосередньо відповідає за забезпечення виконання ключових нефункціональних вимог, а саме: НФВ1 (мінімальна затримка) та НФВ3 (продуктивність). Швидкість, з якою цей модуль здатен виявити новий токен одразу після його розгортання та відстежити перші транзакції, має пряму кореляцію з точністю аналізу його початкової активності; адже чим раніше зібрані дані, тим вища якість інформації для скорингової моделі, що є основою для прийняття своєчасних та високоточних інвестиційних рішень.

3.2.1. Принципи встановлення високошвидкісного з'єднання з блокчейном

Для забезпечення конкурентної переваги на ринку спекулятивного моніторингу, традиційні методи отримання даних було визнано недостатніми:

1. Стандартний RPC (Remote Procedure Call): використовується для синхронних запитів (pull-механізм). Однак, RPC має високу затримку та є непридатним для безперервного потокового моніторингу подій у реальному часі.
2. Стандартний WebSocket (WS): використовується для потокових підписок (push-механізм), але створює значне мережеве навантаження, особливо в умовах високої активності мережі Solana.

Для задоволення вимоги НФВ1 було обрано протокол GRPC, реалізований через спеціалізовані ноди Geyser/Triton One Yellowstone.

Переваги GRPC:

- Бінарна серіалізація: використання Protocol Buffers замість JSON забезпечує значно менший об’єм даних, що передаються, та швидшу десеріалізацію, мінімізуючи накладні витрати (overhead).
- HTTP/2: GRPC використовує протокол HTTP/2, який підтримує мультиплексування, дозволяючи виконувати кілька асинхронних запитів/потоків через одне з'єднання.
- Мінімальна затримка (Ultra-Low Latency): GRPC-сервіси, такі як Geyser, дозволяють системі отримати інформацію про транзакцію з мінімальною затримкою.

Використання спеціалізованої бібліотеки @triton-one/yellowstone-grpc є ключовим архітектурним рішенням для забезпечення вимоги ФВ1, оскільки вона надає програмну можливість для ініціалізації двостороннього потоку даних безпосередньо з вузлами Solana.

3.2.2. Програмне забезпечення GRPC-підписки на логі програми Pump.fun

Підписка на потік лог-даних здійснюється через спеціальну функцію, яка ініціалізує GRPC-клієнт та надсилає запит на підписку.

```
import Client, {
   CommitmentLevel,
   SubscribeRequest,
   SubscribeUpdate,
   SubscribeUpdateTransaction,
} from "@triton-one/yellowstone-grpc";
import { ClientDuplexStream } from '@grpc/grpc-js';

async function subscribeToPFTransactions(handleData: (data:
SubscribeUpdate) => void) {
    const client = new Client(GRPC_ENDPOINT, undefined, {});
    const stream = await client.subscribe();
    const request = createSubscribeRequest();

    try {
        await sendSubscribeRequest(stream, request);
        console.log('Geyser connection established - watching
new mints.');
        await handleStreamEvents(stream, handleData);
    } catch (error) {
        console.error('Error in subscription process:',
error);
        stream.end();

        process.exit(1);
    }
}
```

Лістинг 3.1. Код для з’єднання по GRPC

Сам запит на підписку формується у вигляді структурованого об’єкта, який визначає, які дані система буде відстежувати та отримувати в реальному часі.

```
{
        accounts: {},
        slots: {},
```

```
        transactions: {
            pumpfun: {
                accountInclude: FILTER_CONFIG.programIds,
                accountExclude: [],
                accountRequired: []
            }
        },
        transactionsStatus: {},
        entry: {},
        blocks: {},
        blocksMeta: {},
        commitment: COMMITMENT,
        accountsDataSlice: [],
        ping: undefined,
    }
```

Лістинг 3.2. Вигляд запиту для з’єднання

У полі programIds об’єкта FILTER_CONFIGS зберігається адрес смарт контракту PumpFun, через який здійснюється вся взаємодія (створення токену, інiювання куплі/продажу) - 6EF8rrecthR5Dkzon8Nwu78hRvfCKubJ14M5uBEwF6P.

3.2.3. Алгоритм первинної фільтрації та збору транзакцій

Після встановлення високошвидкісного GRPC-з'єднання (3.2.2), Модуль Моніторингу переходить у режим безперервної обробки даних, реалізуючи алгоритм трекінгу нових токенів (ФВ2, ФВ3).

Для точної ідентифікації події створення нового токена на Pump.fun використовується аналіз дискримінаторів інструкцій (Instruction Discriminators).

- Дискримінатори інструкцій: це перші 8 байт (або 4 байти, залежно від методу кодування) поля data інструкції, які однозначно ідентифікують викликаний метод у смарт-контракті (Програмі PF).
- Версії створення токена: було ідентифіковано дві версії або методи створення токена, які використовуються платформою:
    - Перша версія створення токена (Discriminator V1).

- Друга, оновлена версія створення токена (Discriminator V2).

- Логіка фільтрації: кожна вхідна транзакція аналізується. Системою перевіряється, чи відповідає дискримінатор інструкції транзакції одному з відомих дискримінаторів створення токена. У разі збігу, це є підтвердженням появи нового токена, і система ініціює його відстеження.

Після ідентифікації нового токена, система активує механізм збору перших 100 транзакцій.

- Умова завершення збору: збір даних для конкретного токена має дві умови завершення:
  - ліміт транзакцій (Success Condition): досягнення встановленого ліміту - 100 транзакцій.
  - таймаут (Failure Condition): якщо за певний проміжок часу (наприклад, 60 секунд) токен не зміг зібрати 100 транзакцій, це свідчить про відсутність початкового інтересу або провал запуску. В цьому випадку збір даних припиняється, а токен виключається з аналізу.

- Передача управління: як тільки умова N=100 виконана, зібраний масив транзакцій передається до наступного етапу — Модуля Аналізу та Скорингу, що є початком його роботи (3.3).

Таким чином, Модуль Моніторингу не лише ідентифікує появу токена з високою швидкістю, але й керує життєвим циклом збору даних, гарантуючи, що для подальшого аналізу передаються лише ті токени, які продемонстрували мінімальний початковий інтерес.

```
    async function waitForNTransactions(mint: string, n: number, timeout: number) {
        const timeoutDate = Date.now() + timeout;
        let transactions = 0;

        while (transactions < n && Date.now() < timeoutDate) {
            const signatures = await connection.getSignaturesForAddress(new PublicKey(mint), { limit: 1000 });
```

```
            if (signatures.filter(x => !x.err).length >= n) {
                return true;
            }

            await sleep(1000);
        }

        return false;
    }
```

Лістинг 3.3. Код функції для очікування набору 100 транзакцій

Для підтвердження кількості транзакцій та забезпечення умови таймауту було реалізовано функцію waitForNTransactions (лістинг 3.3). Цей метод використовує стандартний RPC-запит до вузла (getSignaturesForAddress) для перевірки кількості фіналізованих транзакцій, асоційованих з адресою токена.

Загальний код модуля моніторингу знаходиться у додатку А.

3.3. Реалізація модуля аналізу та розрахунку метрик

Модуль Аналізу та Скорингу є центральним логічним елементом системи, який відповідає за перетворення сирих транзакційних даних, отриманих від Модуля Моніторингу, на кількісні метрики та інтегральну оцінку Score.

3.3.1. Процес завантаження та парсингу транзакцій

Після підтвердження наявності 100 успішних транзакцій для нового токена, система переходить до наступного етапу — їхнього завантаження та детального аналізу. На цьому етапі кожна транзакція обробляється окремо для отримання максимально повної інформації про дії трейдерів та зміну стану токена. Для отримання детальних даних, таких як баланси акаунтів, метадані транзакцій, використовується стандартний RPC-запит.

Система спочатку отримує підписи всіх успішних транзакцій (successfulSignatures), а потім викликає метод loadTransactions, який здійснює пакетне завантаження повних структур транзакцій, включаючи їхні метадані.

```
    async function analyzeMint(mint: string) {
       const signatures = await
connection.getSignaturesForAddress(new PublicKey(mint), { limit:
1000 });
       const successfulSignatures = signatures.filter(x =>
!x.err);

       const transactions = await
loadTransactions(successfulSignatures, 200);
       ...
```

Лістинг 3.4. Код збору наявних транзакцій

Ключовим етапом процесу аналізу нових токенів є детальний парсинг кожної окремої транзакції, що дозволяє визначити роль трейдера у взаємодії з токеном — чи він здійснює Купівлю, чи Продаж. Для цього відстежується не лише напрямок транзакції, а й точні обсяги обміну, що дає змогу будувати більш точну картину активності на ринку.

Метод ґрунтується на аналізі змін балансів на спеціальному Bonding Curve Account (BC Account) токена. BC Account виступає своєрідним індикатором ліквідності та стану токена: будь-яка операція з токеном, що впливає на баланс цього акаунта, відображає фактичну взаємодію трейдера з кривою ціни токена. Таким чином, система дозволяє не лише відстежувати обсяги та напрями торгівлі, а й оцінювати динаміку ціни, ліквідність та активність учасників ринку в реальному часі.

Застосування цього підходу є критично важливим для побудови алгоритмічних стратегій аналізу токенів, прогнозування трендів та автоматичного визначення потенційних сигналів для трейдерів, інвесторів або систем моніторингу ринку. Крім того, такий детальний аналіз дозволяє

виявляти аномальні або нетипові транзакції, що може свідчити про спроби маніпуляцій або появу нових стратегічних гравців на ринку.

- Логіка визначення ролі: транзакція є купівлею (buy), якщо на BC Account відбулося надходження SOL та виведення токенів. Вона є продажем (sell), якщо на BC Account відбулося надходження токенів та виведення SOL.
- Використання метаданих: використовуються поля postTokenBalances, preTokenBalances, postBalances та preBalances з метаданих транзакції

```
    const parsedTransactions = transactions.map(tx => {
        const signature = tx.transaction.signatures[0];

        const payer =
tx.transaction.message.accountKeys[0].pubkey;

        const bondingCurve = getBondingCurveAddress(mint);
        const bondingCurveAccountIndex =
tx.transaction.message.accountKeys.findIndex(x =>
x.pubkey.toBase58() === bondingCurve.toBase58());

        const tokenPostBalance =
tx.meta.postTokenBalances?.find(x => x.mint === mint && x.owner
=== bondingCurve.toBase58())?.uiTokenAmount.uiAmount;

        const tokenPreBalance = tx.meta.preTokenBalances?.find(x
=> x.mint === mint && x.owner ===
bondingCurve.toBase58())?.uiTokenAmount.uiAmount;

        const tokenBalanceChange = tokenPostBalance -
tokenPreBalance;

        const solPostBalance =
tx.meta.postBalances[bondingCurveAccountIndex];

        const solPreBalance =
tx.meta.preBalances[bondingCurveAccountIndex];

        const solBalanceChange = (solPostBalance - solPreBalance)
/ LAMPORTS_PER_SOL;

        if (Number.isNaN(solBalanceChange) ||
Number.isNaN(tokenBalanceChange)) {
            return undefined;
        }

        let side = '';
```

```
        if (solBalanceChange > 0) {
            side = 'buy';
        } else {
            side = 'sell';
        }

        return {
            signature,
            payer: payer.toBase58(),
            side,
            tokenAmount: Math.abs(tokenBalanceChange),
            solAmount: Math.abs(solBalanceChange),
            time: tx.blockTime,
        };
    })
        .filter(x => x !== undefined);
```

Лістинг 3.5. Код парсингу транзакцій

3.3.2. Розрахунок ключових метрик активності

Після успішного парсингу даних (parsedTransactions), модуль обчислює п'ять ключових метрик, необхідних для алгоритму скорингу.

1. Метрики обсягу та тренду (M1, M3)
    a. чистий обсяг SOL: розраховується як різниця між загальним обсягом SOL, витраченим на купівлі, та обсягом, отриманим від продажів.
    b. співвідношення купівель/продажів (RBS Ratio): ця метрика демонструє поточний тренд попиту. Використовується співвідношення обсягів: RBS = Обсяг купівлі / Обсяг продажу. В разі нульових продажів, показник встановлюється на високе значення 100.
2. Кількість унікальних трейдерів (M2)
    a. кількість унікальних платників (tx.payer) є прямим індикатором широти попиту та органічної активності, оскільки імітувати велику кількість унікальних гаманців є складніше.

3. Часові метрики (M5)
    a. середній інтервал: для визначення цього показника необхідно спочатку відсортувати зібрані транзакції за часом (tx.time) у висхідному порядку. Середній інтервал розраховується як середнє арифметичне різниць часу між послідовними транзакціями. Менший інтервал може свідчити про наявність бот-активності.
4. Розрахунок концентрації власності (M4)
    a. ця метрика оцінює ризик централізації та потенційного Rug Pull.
    b. позиції користувачів: спочатку обчислюються кінцеві позиції (баланси) токенів для кожного унікального трейдера (userPositions) на основі всіх 100 транзакцій.
    c. концентрація Top 10: список власників сортується за спаданням tokenAmount, і обчислюється сума часток токенів, що належать 10 найбільшим власникам.

3.3.3 Впровадження інтегрального алгоритму скорингу

Фінальний етап роботи Модуля Аналізу полягає у перетворенні п'яти розрахованих метрик у єдину, стандартизовану інтегральну оцінку Score (від 0 до 1). Ця оцінка є головним результатом системи, що дозволяє швидко прийняти рішення щодо потенціалу нового токена. Реалізація Scoring-алгоритму вимагає двох ключових кроків: нормалізації метрик та застосування зваженої суми.

3.4. Впровадження інтегрального алгоритму скорингу (Scoring Algorithm)

Фінальний етап роботи Модуля Аналізу полягає у перетворенні п'яти розрахованих метрик у єдину, стандартизовану інтегральну оцінку Score (від 0 до 1). Ця оцінка є головним результатом системи, що дозволяє швидко

прийняти рішення щодо потенціалу нового токена. Реалізація Scoring-алгоритму вимагає двох ключових кроків: нормалізації метрик та застосування зваженої суми.

3.4.1. Обґрунтування та реалізація функції нормалізації метрик

Нормалізація даних є обов'язковою процедурою в багатофакторному аналізі. Вона необхідна для того, щоб привести всі метрики, які вимірюються в різних одиницях, до єдиного безрозмірного діапазону [0, 1]. Це запобігає ситуації, коли метрика з більшою абсолютною величиною (наприклад, Обсяг) домінує над іншими.

Для нормалізації використовується метод лінійного масштабування (Min-Max Scaling) та функція normalize().

```
    function normalize(value: number, min: number, max: number,
inverse: boolean = false): number {
        const clampedValue = Math.max(min, Math.min(max, value));

        let normalized = (clampedValue - min) / (max - min);
        if (inverse) {
            normalized = 1 - normalized;
        }

        return Math.max(0, Math.min(1, normalized));
    }
```

Лістинг 3.6. Метод нормалізації

3.4.2. Визначення вагових коефіцієнтів та розрахунок зваженої суми

Фінальний Score розраховується як зважена сума нормалізованих метрик. Вагові коефіцієнти Wi визначають ступінь важливості кожної метрики для фінального висновку. Наприклад, метрика, що вказує на органічний попит (як RBS Ratio), отримує більшу вагу, ніж показник бот-активності (Average Interval).

На поточному етапі розробки, вагові коефіцієнти були підібрані імперативним шляхом на основі експертної оцінки та попереднього аналізу ринку Pump.fun. Подальше вдосконалення системи передбачає оптимізацію цих ваг за допомогою інших методів.

Таблиця 3.3

Вагові коефіцієнти

Метрика	Вага	Оцінка важливості
Чистий обсяг SOL	0.20	Важливий, але не ключовий (об'єм можна імітувати).
Унікальні трейдери	0.15	Високий показник органічності.
Співвідношення Купівель/Продажів	0.35	Критично важливий показник тренду (спрос/пропозиція).
Концентрація Top 10	0.20	Ризик-фактор.
Середній інтервал	0.10	Допоміжний індикатор бот-активності.
Сума	1.00	

3.4.3. Генерація базового висновку та звітності

Головна функція calculateTokenScore() виконує комплексний аналіз показників токена. Вона спершу здійснює нормалізацію всіх метрик, приводячи їх до єдиної шкали, що дозволяє коректно порівнювати показники різної природи та масштабу. Далі до кожної нормалізованої метрики

застосовуються вагові коефіцієнти, які відображають її відносну важливість для загальної оцінки.

```
    function calculateTokenScore(metrics: ParsedMetrics): {
score: number, conclusion: string } {
        const { sumVolume, uniqueTradersCount, rbsRatio,
top10Concentration, averageInterval } = metrics;
        const { M1_MIN, M1_MAX, M2_MIN, M2_MAX, M3_MIN, M3_MAX,
M4_MIN, M4_MAX, M5_MIN, M5_MAX } = MIN_MAX_VALUES;
        const scoreM1 = normalize(sumVolume, M1_MIN, M1_MAX);
        const scoreM2 = normalize(uniqueTradersCount, M2_MIN,
M2_MAX);
        const scoreM3 = normalize(rbsRatio, M3_MIN, M3_MAX);
        const scoreM4 = normalize(top10Concentration, M4_MIN,
M4_MAX, true);
        const scoreM5 = normalize(averageInterval, M5_MIN,
M5_MAX, true);
        const finalScore = (
            scoreM1 * WEIGHTS.M1_SUM_VOLUME +
            scoreM2 * WEIGHTS.M2_UNIQUE_WALLETS +
            scoreM3 * WEIGHTS.M3_RBS_RATIO +
            scoreM4 * WEIGHTS.M4_TOP10_CONCENTRATION +
            scoreM5 * WEIGHTS.M5_AVG_INTERVAL
        );

        let conclusion: string;

        if (finalScore >= 0.75) {
            conclusion = "✅ HIGH POTENTIAL. Strong organic
demand and low risk of manipulation. Recommended for immediate
analysis.";
        } else if (finalScore >= 0.50) {
            conclusion = "🟡 MEDIUM POTENTIAL. Moderate
activity, balanced risks. Further observation required.";
        } else if (finalScore >= 0.25) {
            conclusion = "🟠 LOW POTENTIAL. Weak demand or signs
of bot activity. High risk.";
        } else {
            conclusion = "❌ HIGH RISK. Strong concentration
and/or dominance of bots. Not recommended.";
        }

        return {
            score: parseFloat(finalScore.toFixed(4)),
            conclusion
        };
    }
```

Лістинг 3.7. Метод розрахунку фінальної оцінки

Модуль Звітності (Reporting Module) приймає фінальний об'єкт з score та conclusion для форматування та виведення результатів користувачеві (ФВ6).

Загальний код модуля аналізу знаходиться у додатку Б.

3.5. Модуль Звітності (Reporting Module)

Модуль Звітності є фінальним етапом у конвеєрі обробки даних (ФВ6) та відповідає за структурування, збереження та представлення результатів аналізу. Він забезпечує виконання нефункціональної вимоги НФВ4 щодо зберігання даних у структурованому вигляді.

3.5.1. Реалізація та механізм збереження результатів

Після того, як Модуль Аналізу розрахував інтегральний Score та згенерував якісний висновок (conclusion), ці дані передаються до функції processReport для їхньої фіксації.

- Призначення: зберігання історичних даних про токени, що були проаналізовані, для подальшої апробації та порівняння.
- Стек: використовується вбудований у Node.js модуль fs/promises для асинхронної роботи з файловою системою.
- Формат зберігання: всі звіти акумулюються в єдиному файлі reports.json.

Модуль є легко маштабованим для збереження даних у БД або відправки нотифікацій у телеграм.

```
    import fs from 'fs/promises';
    export async function processReport(mint: string, score:
number, conclusion: string) {
        const report = {
            mint,
            score,
            conclusion
        };
```

```
        await saveReportToJson(report);
        await sendReportToTelegram(report);
    }


    async function saveReportToJson({
        mint,
        score,
        conclusion
    }: { mint: string, score: number, conclusion: string }) {
            const reports = await fs.readFile('reports.json',
'utf8');

        const reportsJson = JSON.parse(reports);
        reportsJson.push({
            mint,
            score,
            conclusion
        });

                            await       fs.writeFile('reports.json',
JSON.stringify(reportsJson, null, 2));
    }


    async function sendReportToTelegram({
        mint,
        score,
        conclusion
    }: { mint: string, score: number, conclusion: string }) {
        // realization
    }
```

Лістинг 3.8. Модуль звітування

3.6. Оцінка результатів

Цей підрозділ присвячений демонстрації результатів роботи системи, підтверджуючи її здатність виявляти та оцінювати нові активи.

Апробація системи проводилася на реальному потоці даних мережі Solana протягом двох різних інтервалів часу. Сумарно було зафіксовано та проаналізовано близько 100 нових токенів, що досягли ліміту в 100 транзакцій.

Таблиця 3.4

Вагові коефіцієнти

Адреса токена	M2	M3	M4	Score
8c2S...pump	0.58	0.02	0.18	0.23
A8uY...pump	1.00	0.01	0.00	0.25
AhVC...pump	0.88	0.01	1.00	0.70
3v8c...pump	0.69	0.01	1.00	0.76

Аналіз кейсів:

1. Кейс 8c2S... (HIGH RISK): низький Score (0.23) обумовлений дуже низькими показниками безпеки (M4=0.18, що означає високу концентрацію власності) та попиту (M3=0.02).
2. Кейс A8uY... (LOW POTENTIAL): хоча токен має ідеальний показник унікальних трейдерів (M2=1.00), абсолютна відсутність безпеки (M4=0.00) призводить до мінімального фінального Score. Це ілюструє, як ризик-метрика (M4) може обнулити вплив позитивних факторів.
3. Кейс 3v8c... (HIGH POTENTIAL): високий Score (0.76) досягнутий завдяки ідеальному показнику безпеки (M4=1.00, що означає низьку концентрацію), що разом із помірною органічною активністю (M2=0.69) переводить токен у зону високого потенціалу.

3.7. Шляхи подальшого вдосконалення системи

Розроблена система повністю відповідає поставленим функціональним та нефункціональним вимогам, забезпечуючи високошвидкісний моніторинг

та надійний початковий скоринг токенів. Однак, зважаючи на високу динаміку розвитку ринку децентралізованих фінансів (DeFi) та появу нових інструментів, існує низка ключових напрямків для стратегічного розвитку системи моніторінгу, підвищення точності прогнозування та масштабованості архітектури.

3.7.1. Оптимізація вагових коефіцієнтів Scoring-алгоритму через ML-методи

На поточному етапі вагові коефіцієнти метрик були підібрані евристичним (імперативним) шляхом на основі експертних знань та попереднього аналізу ринку. Для вдосконалення системи необхідно впровадити статистичні та Machine Learning (ML) методи.

Подальша робота повинна бути спрямована на статистичний аналіз кореляції між кожною окремою метрикою та фінальним успіхом токена. Успіх токена може бути визначений як бінарний результат (наприклад, чи відбувся перехід на PumpSwap AMM, чи досягнув токен капіталізації X usd).

- Регресійний аналіз: використання регресії дозволить кількісно оцінити, наскільки значущим є внесок кожної метрики в ймовірність успіху токена.
- Автоматизований підбір ваг: отримані коефіцієнти регресії можуть бути безпосередньо використані для автоматичного калібрування вагових коефіцієнтів, замінюючи їхнє ручне налаштування.

Для максимізації точності прогнозування слід використовувати алгоритми класифікації.

- Алгоритми: розглядається застосування регресії (для оцінки ймовірності успіху) та більш потужних методів. Ці моделі дозволять не лише визначити ваги, а й виявити нелінійні взаємозв'язки між метриками, які неможливо врахувати у простій зваженій сумі.
- Етап навчання: для ML-аналізу необхідний великий, чистий історичний датасет, що робить необхідною зміну архітектури зберігання даних.

3.7.2. Розширення функціоналу та архітектурна оптимізація

Для забезпечення надійності, масштабованості та підтримки необхідно провести оновлення архітектури системи.

Поточне використання файлової системи JSON для зберігання звітів є достатнім лише для тестового середовища. Для ефективного зберігання та швидкого доступу до великих обсягів історичних даних, необхідних для навчання ML-моделей та проведення бектестингу, потрібна повноцінна база даних.

Розглядається використання PostgreSQL (як надійної реляційної СУБД) або MongoDB (як гнучкої NoSQL-бази даних), залежно від вимог до схеми та обсягів неструктурованих транзакційних даних.

Критичною зміною у життєвому циклі токена є Міграція Ліквідності (Liquidity Migration) — перехід торгів із Bonding Curve на стандартний PumpSwap AMM. Ця подія різко змінює ринкову динаміку та структуру комісій.

Система повинна бути розширена для автоматичного відстеження цієї події та додавання відповідної позначки до звіту. Це дозволить оцінити поведінку токена не тільки на етапі Pump.fun, а й після переходу на відкритий DEX-ринок.

Для забезпечення максимальної практичної цінності системи необхідна розробка шару взаємодії з користувачем.

Інтеграція зі сторонніми сервісами сповіщень (Telegram або Discord) дозволить миттєво інформувати користувача про токени, які отримали високий Score, що є критичним для своєчасного інвестиційного рішення.

Розробка простого веб-інтерфейсу для візуалізації історичних даних, динаміки скорингу та графічного представлення ключових метрик.

Таким чином, подальший розвиток системи забезпечить перехід від ефективного інструменту моніторингу до комплексної, інтелектуальної аналітичної платформи.

3.7.3. Впровадження нових метрик

На ринку мем-токенів, особливо на платформі Pump.fun, де ціна значною мірою залежить від хайпу та психології натовпу, активність спільноти є ключовим, а часто й провідним, драйвером ціни. Таким чином, кількісний аналіз фінансових метрик має бути доповнений якісним показником настроїв спільноти.

Сентимент-аналіз (Sentiment Analysis) дозволяє перетворити неструктуровані текстові дані із соціальних медіа у кількісний показник. Позитивний сентимент (наприклад, зростання кількості позитивних згадок токена) є сильним прогностичним фактором для припливу нових покупців та продовження висхідного руху Bonding Curve.

Впровадження Sentiment-аналізу в конвеєр обробки даних вимагає послідовного виконання таких етапів:

1. Збір даних із зовнішніх джерел:
    a. Соціальні мережі: Основні джерела – Twitter (X) API та Telegram. Необхідно налаштувати збір повідомлень за ключовими словами (назвою або хештегом токена), використовуючи їхні офіційні API.
    b. Часові рамки: Збір даних має бути обмежений критичним періодом – першими кількома годинами життя токена, щоб сентимент був максимально релевантним початковому етапу.
2. Попередня обробка тексту:
    a. Очищення: Видалення нерелевантних символів, посилань, емодзі (або їхня токенізація, якщо вони несуть смислове навантаження).
    b. Токенізація та нормалізація: Розбиття тексту на слова (токени), приведення слів до нормальної форми та видалення стоп-слів (артиклів, прийменників), які не впливають на тональність.
3. Моделювання та оцінка тональності:
    a. Вибір моделі: На початковому етапі можна використовувати лексичний підхід – зіставлення слів із готовими словниками, де

кожному слову присвоєно оцінку тональності (позитивна/негативна).

b. Machine Learning: Для більшої точності необхідне застосування моделей, навчених на великих корпусах крипто-тексту.

4. Інтеграція в Scoring-алгоритм: Фінальний результат сентимент-аналізу нормалізується до діапазону [0, 1] і включається як додатковий ваговий коефіцієнт.

Впровадження сентимент-аналізу дозволить системі перейти від суто кількісної оцінки транзакцій до гібридного аналізу, що значно підвищить її прогностичну цінність.

ВИСНОВКИ

У процесі виконання дипломної роботи було успішно досягнуто поставленої мети, що полягала у розробці та впровадженні високоефективної системи моніторингу та кількісної оцінки інвестиційного потенціалу нових токенів, які запускаються на децентралізованій платформі Pump.fun у мережі Solana.

Розроблена система має модульну подієво-орієнтовану архітектуру на базі Node.js/TypeScript, що складається з Модуля Моніторингу, Модуля Аналізу та Модуля Звітності. Така декомпозиція забезпечила високу швидкість обробки даних та відповідає вимогам до продуктивності та надійності.

Впроваджено логіку парсингу транзакцій, яка ґрунтується на аналізі зміни балансів на Bonding Curve Account. Це дозволило точно ідентифікувати роль кожного трейдера (Купівля/Продаж) та обсяги торгівлі, що є основою для всіх подальших розрахунків.

Було формалізовано п'ять ключових метрик, що оцінюють органічність попиту, ризик централізації та інтенсивність торгівлі. На їхній основі розроблено інтегральний алгоритм скорингу, що використовує нормалізацію та зважену суму для виведення єдиної кількісної оцінки. Це дозволяє автоматично класифікувати токени як «Високий», «Середній» або «Низький» потенціал.

Система була успішно апробована на реальних даних, підтвердивши свою ефективність у виявленні токенів із сильним органічним попитом та низьким ризиком маніпуляцій. Одержані результати можуть бути використані як інструмент автоматизованого прийняття рішень у сфері децентралізованих фінансів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Solana Cookbook: офіційна документація — Режим доступу: https://solana.com/ru/developers/cookbook
2. Blockchain Basics: стаття на coingeek — Режим доступу: https://coingeek.com/blockchain101/blockchain-basics-key-things-to-know-as-a-beginner/
3. Blockchain Platform Comparison: стаття на blaize.tech — Режим доступу: https://blaize.tech/blog/blockchain-platform-comparison-a-comprehensive-analysis/
4. Механізми консенсусу: стаття на hacken.io — Режим доступу: https://hacken.io/discover/consensus-mechanisms/
5. Основи смарт-контрактів: стаття на quicknode.com — Режим доступу: https://www.quicknode.com/guides/ethereum-development/smart-contracts/an-overview-of-how-smart-contracts-work-on-ethereum
6. Solana RPC methods: Solana Docs — Режим доступу: https://solana.com/ru/docs/rpc
7. Solana WS methods: Solana Docs — Режим доступу: https://solana.com/ru/docs/rpc/websocket
8. Fungible Tokens: стаття на Trust Wallet — Режим доступу: https://trustwallet.com/ru/blog/nft/fungible-vs-non-fungible-tokens-explained
9. GRPC Basics — Режим доступу: https://www.helius.dev/docs/grpc
10. PumpFun Math — Режим доступу: https://medium.com/@buildwithbhavya/the-math-behind-pump-fun-b58fdb30ed77

ДОДАТКИ

ДОДАТОК А

Код модуля моніторингу

```
import Client, {
   CommitmentLevel,
   SubscribeRequest,
   SubscribeUpdate,
   SubscribeUpdateTransaction,
} from "@triton-one/yellowstone-grpc";
import { ClientDuplexStream } from '@grpc/grpc-js';
import bs58 from 'bs58';
import { Connection, PublicKey } from "@solana/web3.js";
import { GRPC_ENDPOINT, RPC_ENDPOINT } from "./config";


interface CompiledInstruction {
   programIdIndex: number;
   accounts: Uint8Array;
   data: Uint8Array;
}

interface Message {
   header: MessageHeader | undefined;
   accountKeys: Uint8Array[];
   recentBlockhash: Uint8Array;
   instructions: CompiledInstruction[];
   versioned: boolean;
   addressTableLookups: MessageAddressTableLookup[];
}

interface MessageHeader {
   numRequiredSignatures: number;
   numReadonlySignedAccounts: number;
   numReadonlyUnsignedAccounts: number;
}

interface MessageAddressTableLookup {
   accountKey: Uint8Array;
   writableIndexes: Uint8Array;
   readonlyIndexes: Uint8Array;
}

interface FormattedTransactionData {
   signature: string;
```

```
    slot: string;
    [accountName: string]: string;
}


const connection = new Connection(RPC_ENDPOINT);

const PUMP_FUN_PROGRAM_ID =
'6EF8rrecthR5Dkzon8Nwu78hRvfCKubJ14M5uBEwF6P';
const PUMP_FUN_CREATE_IX_DISCRIMINATOR = Buffer.from([24, 30,
200, 40, 5, 28, 7, 119]);
const PUMP_FUN_CREATE_V2_IX_DISCRIMINATOR = Buffer.from([214,
144, 76, 236, 95, 139, 49, 180]);

const COMMITMENT = CommitmentLevel.CONFIRMED;

const FILTER_CONFIG = {
    programIds: [PUMP_FUN_PROGRAM_ID],
    instructionDiscriminators: [PUMP_FUN_CREATE_IX_DISCRIMINATOR,
PUMP_FUN_CREATE_V2_IX_DISCRIMINATOR]
};


const ACCOUNTS_TO_INCLUDE = {
    pumpfun: [{
        name: "mint",
        index: 0
    }]
};


async function sleep(ms: number) {
    return new Promise(resolve => setTimeout(resolve, ms));
}


export async function waitForNTransactions(mint: string, n:
number, timeout: number) {
    const timeoutDate = Date.now() + timeout;
    let transactions = 0;

    while (transactions < n && Date.now() < timeoutDate) {
        const signatures = await
connection.getSignaturesForAddress(new PublicKey(mint), { limit:
1000 });
        if (signatures.filter(x => !x.err).length >= n) {
```

```
            return true;
        }

        await sleep(10000);
    }

    return false;
}


export function formatData(message: Message, signature: string,
slot: string): FormattedTransactionData | undefined {
    const matchingInstruction =
message.instructions.find(matchesInstructionDiscriminator);

    if (!matchingInstruction) {
        return undefined;
    }

    const accountKeys = message.accountKeys;

    const includedAccounts =
ACCOUNTS_TO_INCLUDE["pumpfun"].reduce<Record<string,
string>>((acc, { name, index }) => {
        const accountIndex = matchingInstruction.accounts[index];
        const publicKey = accountKeys[accountIndex];
        acc[name] = new PublicKey(publicKey).toBase58();
        return acc;
    }, {});

    return {
        signature,
        slot,
        ...includedAccounts
    };
}


export function convertSignature(signature: Uint8Array): {
base58: string } {
    return { base58: bs58.encode(Buffer.from(signature)) };
}


export function matchesInstructionDiscriminator(ix:
CompiledInstruction): boolean {
```

```
    return ix?.data &&
FILTER_CONFIG.instructionDiscriminators.some(discriminator =>
        Buffer.from(discriminator).equals(ix.data.slice(0, 8))
    );
}


export function isSubscribeUpdateTransaction(data:
SubscribeUpdate): data is SubscribeUpdate & { transaction:
SubscribeUpdateTransaction } {
    return (
        'transaction' in data &&
        typeof data.transaction === 'object' &&
        data.transaction !== null &&
        'slot' in data.transaction &&
        'transaction' in data.transaction
    );
}


function createSubscribeRequest(): SubscribeRequest {
    return {
        accounts: {},
        slots: {},
        transactions: {
            pumpfun: {
                accountInclude: FILTER_CONFIG.programIds,
                accountExclude: [],
                accountRequired: []
            }
        },
        transactionsStatus: {},
        entry: {},
        blocks: {},
        blocksMeta: {},
        commitment: COMMITMENT,
        accountsDataSlice: [],
        ping: undefined,
    };
}


function sendSubscribeRequest(
    stream: ClientDuplexStream<SubscribeRequest,
SubscribeUpdate>,
    request: SubscribeRequest
```

```
): Promise<void> {
    return new Promise<void>((resolve, reject) => {
        stream.write(request, (err: Error | null) => {
            if (err) {
                reject(err);
            } else {
                resolve();
            }
        });
    });
}


function handleStreamEvents(
    stream: ClientDuplexStream<SubscribeRequest,
SubscribeUpdate>,
    handleData: (data: SubscribeUpdate) => void
): Promise<void> {
    return new Promise<void>((resolve, reject) => {
        stream.on('data', handleData);
        stream.on("error", (error: Error) => {
            console.error('Stream error:', error);
            reject(error);
            stream.end();
        });
        stream.on("end", () => {
            console.log('Stream ended');
            resolve();
        });
        stream.on("close", () => {
            console.log('Stream closed');
            resolve();
        });
    });
}


export async function subscribeToPFTransactions(handleData:
(data: SubscribeUpdate) => void) {
    const client = new Client(GRPC_ENDPOINT, undefined, {});
    const stream = await client.subscribe();
    const request = createSubscribeRequest();

    try {
        await sendSubscribeRequest(stream, request);
```

```
        console.log('Geyser connection established - watching new
mints.');
        await handleStreamEvents(stream, handleData);
    } catch (error) {
        console.error('Error in subscription process:', error);
        stream.end();

        process.exit(1);
    }
}
```

ДОДАТОК Б

Код модуля аналізу

```
import { Connection, LAMPORTS_PER_SOL,
ParsedTransactionWithMeta, PublicKey } from "@solana/web3.js";
import { RPC_ENDPOINT } from "./config";


interface ParsedMetrics {
   sumVolume: number;
   uniqueTradersCount: number;
   rbsRatio: number;
   top10Concentration: number;
   averageInterval: number;
}

const WEIGHTS = {
   M1_SUM_VOLUME: 0.2,
   M2_UNIQUE_WALLETS: 0.15,
   M3_RBS_RATIO: 0.35,
   M4_TOP10_CONCENTRATION: 0.2,
   M5_AVG_INTERVAL: 0.1,
};

const MIN_MAX_VALUES = {
   M1_MIN: 0, M1_MAX: 10000,
   M2_MIN: 1, M2_MAX: 100,
   M3_MIN: 0, M3_MAX: 100,
   M4_MIN: 0, M4_MAX: 1,
   M5_MIN: 0, M5_MAX: 200,
};


const connection = new Connection(RPC_ENDPOINT);


async function sleep(ms: number) {
   return new Promise(resolve => setTimeout(resolve, ms));
}


export async function analyzeMint(mint: string) {
```

```
   console.log("=====================================💊 Found
mint with enough
transactions!=====================================");

   const signatures = await
connection.getSignaturesForAddress(new PublicKey(mint), { limit:
1000 });
   const successfulSignatures = signatures.filter(x => !x.err);

   const transactions = await
loadTransactions(successfulSignatures, 200);
   const parsedTransactions = transactions.map(tx => {
       const signature = tx.transaction.signatures[0];
       const payer =
tx.transaction.message.accountKeys[0].pubkey;

       const bondingCurve = getBondingCurveAddress(mint);
       const bondingCurveAccountIndex =
tx.transaction.message.accountKeys.findIndex(x =>
x.pubkey.toBase58() === bondingCurve.toBase58());

       const tokenPostBalance =
tx.meta.postTokenBalances?.find(x => x.mint === mint && x.owner
=== bondingCurve.toBase58())?.uiTokenAmount.uiAmount;
       const tokenPreBalance = tx.meta.preTokenBalances?.find(x
=> x.mint === mint && x.owner ===
bondingCurve.toBase58())?.uiTokenAmount.uiAmount;

       const tokenBalanceChange = tokenPostBalance -
tokenPreBalance;

       const solPostBalance =
tx.meta.postBalances[bondingCurveAccountIndex];
       const solPreBalance =
tx.meta.preBalances[bondingCurveAccountIndex];

       const solBalanceChange = (solPostBalance - solPreBalance)
/ LAMPORTS_PER_SOL;

       if (Number.isNaN(solBalanceChange) ||
Number.isNaN(tokenBalanceChange)) {
           return undefined;
       }

       let side = '';
```

```
        if (solBalanceChange > 0) {
            side = 'buy';
        } else {
            side = 'sell';
        }

        return {
            signature,
            payer: payer.toBase58(),
            side,
            tokenAmount: Math.abs(tokenBalanceChange),
            solAmount: Math.abs(solBalanceChange),
            time: tx.blockTime,
        };
    })
        .filter(x => x !== undefined);

    const sumVolume = parsedTransactions.reduce((acc, tx) => acc
+ tx.solAmount * (tx.side === 'buy' ? 1 : -1), 0);
    const uniqueTraders = [...new Set(parsedTransactions.map(tx
=> tx.payer))];

    const buysVolume = parsedTransactions.filter(tx => tx.side
=== 'buy').reduce((acc, tx) => acc + tx.solAmount, 0);
    const sellsVolume = parsedTransactions.filter(tx => tx.side
=== 'sell').reduce((acc, tx) => acc + tx.solAmount, 0);

    const rbsRatio = sellsVolume > 0 ? buysVolume / sellsVolume :
100;

    const tradesIntervals = [];
    for (let i = 0; i < parsedTransactions.length - 1; i++) {
        const currentTx = parsedTransactions[i];
        const nextTx = parsedTransactions[i + 1];
        const interval = currentTx.time - nextTx.time;
        tradesIntervals.push(interval);
    }

    const averageInterval = average(tradesIntervals) || 0;

    const userPositions = {};

    for (const tx of parsedTransactions) {
        if (!userPositions[tx.payer]) {
            userPositions[tx.payer] = {
                wallet: tx.payer,
```

```
                tokenAmount: 0,
                balanceTotalSupplyRatio: 0
            };
        }

        if (tx.side === 'buy') {
            userPositions[tx.payer].tokenAmount +=
tx.tokenAmount;
        } else {
            userPositions[tx.payer].tokenAmount -=
tx.tokenAmount;
        }
    }

    const totalSupply = parsedTransactions.reduce((acc, tx) =>
acc + tx.tokenAmount * (tx.side === 'buy' ? 1 : -1), 0);

    const sortedHolders = Object.values(userPositions)
        .sort((a: any, b: any) => b.tokenAmount - a.tokenAmount);

    const top10Concentration = sortedHolders
        .slice(0, 10)
        .reduce((acc, holder: any) => acc + holder.tokenAmount,
0) as number / totalSupply;

    const metrics: ParsedMetrics = {
        sumVolume,
        uniqueTradersCount: uniqueTraders.length,
        rbsRatio,
        top10Concentration,
        averageInterval
    };

    const score = calculateTokenScore(metrics);
    return score;
}


function normalize(value: number, min: number, max: number,
inverse: boolean = false): number {
    const clampedValue = Math.max(min, Math.min(max, value));

    let normalized = (clampedValue - min) / (max - min);

    if (inverse) {
        normalized = 1 - normalized;
```

```
    }

    return Math.max(0, Math.min(1, normalized));
}


function calculateTokenScore(metrics: ParsedMetrics): { score:
number, scoreM1: number, scoreM2: number, scoreM3: number,
scoreM4: number, scoreM5: number, conclusion: string } {
    const { sumVolume, uniqueTradersCount, rbsRatio,
top10Concentration, averageInterval } = metrics;
    const { M1_MIN, M1_MAX, M2_MIN, M2_MAX, M3_MIN, M3_MAX,
M4_MIN, M4_MAX, M5_MIN, M5_MAX } = MIN_MAX_VALUES;

    const scoreM1 = normalize(sumVolume, M1_MIN, M1_MAX);
    const scoreM2 = normalize(uniqueTradersCount, M2_MIN,
M2_MAX);
    const scoreM3 = normalize(rbsRatio, M3_MIN, M3_MAX);
    const scoreM4 = normalize(top10Concentration, M4_MIN, M4_MAX,
true);
    const scoreM5 = normalize(averageInterval, M5_MIN, M5_MAX,
true);

    const finalScore = (
        scoreM1 * WEIGHTS.M1_SUM_VOLUME +
        scoreM2 * WEIGHTS.M2_UNIQUE_WALLETS +
        scoreM3 * WEIGHTS.M3_RBS_RATIO +
        scoreM4 * WEIGHTS.M4_TOP10_CONCENTRATION +
        scoreM5 * WEIGHTS.M5_AVG_INTERVAL
    );

    let conclusion: string;

    if (finalScore >= 0.75) {
        conclusion = "✅ HIGH POTENTIAL. Strong organic demand
and low risk of manipulation. Recommended for immediate
analysis.";
    } else if (finalScore >= 0.50) {
        conclusion = "🟡 MEDIUM POTENTIAL. Moderate activity,
balanced risks. Further observation required.";
    } else if (finalScore >= 0.25) {
        conclusion = "🟠 LOW POTENTIAL. Weak demand or signs of
bot activity. High risk.";
    } else {
        conclusion = "❌ HIGH RISK. Strong concentration and/or
dominance of bots. Not recommended.";
```

```
    }

    return {
        score: parseFloat(finalScore.toFixed(4)),
        scoreM1,
        scoreM2,
        scoreM3,
        scoreM4,
        scoreM5,
        conclusion,
    };
}


function average(arr: number[]) {
    return arr.reduce((acc, tx) => acc + tx, 0) / arr.length;
}


function getBondingCurveAddress(mint: string) {
    const [bondingCurve] =
PublicKey.findProgramAddressSync([Buffer.from("bonding-curve"),
new PublicKey(mint).toBytes()], new
PublicKey('6EF8rrecthR5Dkzon8Nwu78hRvfCKubJ14M5uBEwF6P'));
    return bondingCurve;
}


async function loadTransactions(signatures: any[], limit = 1000)
{
    const result: ParsedTransactionWithMeta[] = [];

    for (const signature of signatures) {
        const transaction = await
connection.getParsedTransaction(signature.signature, {
            commitment: 'confirmed',
            maxSupportedTransactionVersion: 0
        });

        result.push(transaction);

        if (result.length >= limit) {
            break;
        }

        await sleep(Math.random() * 120);
```

```
    }

    return result;
}
```

ДОДАТОК В

Код модуля звітування

```
import fs from 'fs/promises';

interface Report {
   score: number;
   scoreM1: number;
   scoreM2: number;
   scoreM3: number;
   scoreM4: number;
   scoreM5: number;
   conclusion: string;
}

export async function processReport(mint: string, report:
Report) {
   await saveReportToJson(mint, report);
   await sendReportToTelegram(mint, report);
}

async function saveReportToJson(mint: string, report: Report) {
   const reports = await fs.readFile('reports.json', 'utf8');

   const reportsJson = JSON.parse(reports);
   reportsJson.push({
       mint,
       report,
   });

   await fs.writeFile('reports.json',
JSON.stringify(reportsJson, null, 2));
}

async function sendReportToTelegram(
   mint: string,
   report: Report
) {
   // realization
}
```