

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Інтеграція ШІ Mistral у Discord-асистент для взаємодії з
криптовалютними сервісами

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Болезука Віктора Вікторовича

(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Омельчук Антон Анатолійович,
канд. техн. наук,

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «12» травня 2026 № 20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Реєстрація № 49

27.05.2026
(дата)


(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

12.05.2026
(дата)


(підпис)

Результат захисту

5 (відмінно)
(оцінка)

09.06.2026
(дата)


(підпис)

Голова комісії:


(підпис)

О.С. Дюбанов

Члени комісії:


(підпис)

М.М. Філоненко


(підпис)

В.О. Ніжегородцев


(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	27		09.06.2026 Омельчук А.А. Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхов Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником – 47 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
92	A	5 (відмінно)	09.06.2026	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхов Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій

Кафедра комп'ютерних та інформаційних технологій і систем

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри  Філоненко М.М.

«01» грудня 2025

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Болещук Віктору Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Інтеграція III Mistral у Discord-асистент для взаємодії з криптовалютними сервісами

Затверджена наказом ректора від «28» листопада 2025 № 1526

2. Строк здачі закінченої роботи «08» травня 2026

3. Мета роботи Проектування та програмна реалізація автономного вузла «Mistra Node» у вигляді Discord-асистента для взаємодії з криптовалютними сервісами.

Об'єкт дослідження Інтеграція технологій III в системи моніторингу фінансових ринків і розробка автоматизованих інструментів для взаємодії користувачів із блокчейн-платформами.

Предмет дослідження Процеси програмної реалізації, асинхронної інтеграції та оптимізації локальних мовних моделей і OSINT-інструментів у межах єдиної модульної архітектури Discord-асистента.

Орієнтовна база дослідження На відкритих специфікаціях Discord API, принципах побудови асинхронних систем та методах захисту інформації в мережах.

4. Вихідні дані до роботи Технічна документація моделі Mistral AI, специфікації асинхронної бібліотеки disnake та відкриті протоколи інтеграції з сервісом VirusTotal. Інформаційною базою стали принципи побудови локальних систем RAG, алгоритми обчислення оцінки загрози Risk Score та методики розгортання асинхронного програмного софту.

5. Перелік графічного матеріалу Схема вузла «Mistra Node», діаграма взаємодії модулів через Discord API Gateway, блок-схема алгоритму пошуку в локальній базі знань RAG, функціональна схема розрахунку інтегрального показника Risk Score на основі вердиктів OSINT-сервісу

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом) у вступі обґрунтовано актуальність, мету та предмет. Перший розділ присвячено теорії мовних моделей і захисту даних. Другий містить аналіз Discord API, технологій RAG та OSINT. Третій описує програмну реалізацію вузла «Mistra Node», асинхронного ядра, інтеграцію з VirusTotal та оптимізацію бази знань. У висновках підсумовано результати розробк

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	30.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	05.02.2026	01.02.2026
3	II розділ (аналіз експериментальної частини про методики досліджень)	28.02.2026	4.03.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	15.04.2026	18.04.2026
5	Висновки	01.05.2026	05.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	15.05.2026	28.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи


(підпис)

Омельчук А.А.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Ніжегородцев В.О.
(прізвище, ініціали)

11. Завдання до виконання отримав
здобувач вищої освіти


(підпис)

Болезчук В.В.
(прізвище, ініціали)

АНОТАЦІЯ

« Інтеграція ШІ Mistral у Discord-асистент для взаємодії з криптовалютними сервісами »

Мета роботи полягає в проєктуванні та програмній реалізації автономного інтеграційного вузла у вигляді асинхронного Discord-асистента на базі локальної мовної моделі.

Завданнями роботи є проведення аналізу архітектурних рішень для взаємодії з Discord API; обґрунтування вибору сучасного асинхронного технологічного стеку; розробка модульної структури та ядра інтеграційного вузла; проєктування алгоритмів семантичного пошуку для локальної бази знань RAG; програмна реалізація модулів OSINT-моніторингу та оцінки кіберзагроз.

Предметом дослідження є методи та технології створення асинхронних інтеграційних модулів, локального розгортання великих мовних моделей та автоматизації скорингу загрози Risk Score.

Об'єктом дослідження є процес програмної реалізації інтелектуальних асистентів та інструментів збору даних із відкритих джерел.

Базою дослідження виступає програмна інфраструктура розроблюваного автономного інтеграційного вузла «Mistra Node».

За результатами дослідження сформульовані методика, етапи та архітектурні рішення розробки асинхронного Discord-асистента. Одержані результати можуть бути використані для практичного розгортання локальних ШІ-помічників як захищених інструментів моніторингу, а також як методична основа для розробки інших інтеграційних модулів і OSINT-систем.

Кваліфікаційна робота містить 46 сторінок, 5 таблиць, 14 рисунків, список використаних джерел 30 найменування

Ключові слова: інтеграційний модуль, Mistral AI, асинхронна архітектура, Discord API, Disnake, RAG, OSINT, VirusTotal.

Рік виконання кваліфікаційної роботи 2026

Рік захисту роботи 202

ABSTRACT

«Development and Software Implementation of the "Mistra Node" Integration Module»

The aim of this work is to develop the architecture and software implementation of an autonomous integration node in the form of an asynchronous Discord assistant based on a local large language model.

The tasks of the work are as follows: analysis of architectural solutions for interaction with the Discord API; justification of the choice of a modern asynchronous technology stack; development of the modular structure and core of the integration node; designing semantic search algorithms for the local RAG knowledge base; software implementation of OSINT monitoring and cyber threat assessment modules.

The subject of the research is methods and technologies for creating asynchronous integration modules, local deployment of large language models, and automation of Risk Score threat assessment. **The object of the research** is the process of software implementation of intelligent assistants and open-source intelligence tools.

The research is based on the software infrastructure of the developed autonomous integration node "Mistra Node".

The results of the research include the methodology, stages, and architectural solutions for developing an asynchronous Discord assistant. The obtained results can be used for the practical deployment of local AI assistants as secure monitoring tools, as well as a methodological basis for developing other integration modules and OSINT systems.

The qualification work consists of 46 pages, 5 tables, 14 figures, a list of 30 references.

Keywords: integration module, Mistral AI, asynchronous architecture, Discord API, Disnake, RAG, OSINT, VirusTotal.

Year of qualification work: 2026

Year of work defense: 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	4
ВСТУП.....	6
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДИЧНІ ОСНОВИ ЗАСТОСУВАННЯ ШІ У ФІНАНСАХ	
1.1. Актуальність теми ШІ та криптовалют.....	9
1.2. Аналіз ринку інтелектуальних асистентів для крипто-моніторингу.....	10
1.3. Обґрунтування вибору моделі Mistral AI та технологічного стека...	13
РОЗДІЛ 2. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ DISCORD-АСИСТЕНТА	
2.1. Концептуальна модель асистента та інтеграція з API сервісів.....	15
2.2. Проектування RAG-системи для роботи з локальною базою знань.	16
2.3. Алгоритми захисту даних та розмежування рівнів доступу.....	18
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕГРАЦІЙНИХ МОДУЛІВ	
3.1. Вибір мови програмування та бібліотек.....	21
3.2. Розробка асинхронного інтерфейсу взаємодії з Discord API.....	23
3.3. Альтернативні методи оптимізації та розширення бази знань ШІ-асистента.....	25
3.4. Програмна реалізація спеціалізованих каналів взаємодії користувачів.....	27

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ ВУЗЛА

4.1. Методика стрес-тестування системи в ізольованому середовищі...37

4.2. Аналіз результатів тестування та ефективності роботи алгоритмів.39

ВИСНОВКИ.....40

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....43

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI, ІІІ – Artificial Intelligence (штучний інтелект)

ML – Machine Learning (машинне навчання)

DL – Deep Learning (глибоке навчання)

API (Application Programming Interface) — програмний інтерфейс взаємодії додатків.

Binance API – API для роботи з криптовалютною біржею Binance

Discord Bot API – інструменти для створення та взаємодії з ботами у Discord.

BTC – біткоїн

ETH – ефір

CSV – Comma-Separated Values формат файлів, у яких дані зберігаються у вигляді значень, розділених комами.

VS Code – Visual Studio Code інтегроване середовище розробки.

VENV – Virtual Environment віртуальне середовище для ізоляції залежностей Python.

CPU (Central Processing Unit) — центральний процесор.

GQA (Grouped-query Attention) — механізм групованої уваги в архітектурі трансформерів (використовується в Mistral AI).

JSON (JavaScript Object Notation) — текстовий формат обміну даними.

LLM (Large Language Model) — велика мовна модель.

NLP (Natural Language Processing) — обробка природної мови.

OSINT (Open Source Intelligence) — розвідка на основі відкритих джерел.

RAM (Random Access Memory) — оперативна пам'ять.

RAG (Retrieval-Augmented Generation) — архітектура генерації з доповненням пошуком.

SDK (Software Development Kit) — набір засобів розробки програмного забезпечення.

URL (Uniform Resource Locator) — уніфікований покажчик ресурсу.

UX (User Experience) — досвід користувача.

ВСТУП

Актуальність дослідження. Поява криптовалют – це один із наслідків об'єктивного процесу конкуренції між системами фіксації фактів переходу вартості. [1] Хоча криптовалюти існують уже понад одинадцять років, у цій сфері досі панує термінологічна плутанина. Позиції провідних економістів суттєво різняться, а реальних досліджень того, як традиційні банки мають взаємодіяти з крипторинком, майже немає. До кінця не зрозуміло, як саме цей фінансовий інструмент змінить світову систему в майбутньому. Поява електронних грошових коштів, позбавлених матеріального носія загального еквівалента, вимагає вивчення їх природи і родової основи з урахуванням нових реалій постіндустріального економічного укладу, формування «цифрової» економіки [1]

До прикладу візьмемо найпопулярнішу монету на цей час - Біткоїн (BTC). Раніше ніхто зі звичайних людей не міг подумати, що якась незрозуміла електронна монета буде коштувати більше ста тисяч доларів США. Можемо взяти інвесторів або знавців в економіці. Відомий інвестор Білл Міллер акцентує увагу на спекулятивній складовій цифрових активів. Його теза про те, що ціна біткоїна може як сягнути історичних максимумів, так і впасти до нульової позначки, ілюструє відсутність надійних фундаментальних моделей оцінки криптовалют на цьому етапі. Крім того, зазначається, що віртуальні валюти не демонструють чіткої кореляції з фондовим ринком. Цей факт дає підстави вважати біткоїн перспективним інвестиційним активом, який може використовуватися для зниження загальних системних ризиків у фінансових операціях. [2] Але ми маємо розуміння, що криптовалюта з кожним роком стає ще популярнішою. Їх дедалі більша затребуваність у масштабах світової економіки робить питання інтелектуального супроводу таких операцій одним із пріоритетних завдань сучасної ІТ-галузі. Саме масовий попит на криптоактиви вимагає створення інструментів, що мінімізують вплив людського фактора при аналізі великих масивів даних.

У цих умовах традиційні підходи до моніторингу фінансових ринків вже не забезпечують належного рівня швидкості та точності аналізу. Використання великих мовних моделей (Large Language Models, LLM), наприклад на основі архітектури Mistral AI, у поєднанні із системами автоматизованого аналізу відкритих джерел інформації (OSINT), відкриває нові горизонти для створення інтелектуальних середовищ. Mistral AI — це французький стартап у сфері штучного інтелекту (ШІ), відомий, перш за все, своїми моделями великих мов програмування (LLM) з відкритим кодом. З моменту свого заснування у 2023 році він став одним із провідних світових розробників генеративного ШІ. [3] Такі системи здатні не лише об'єднувати великі обсяги даних, але й здійснювати їх глибоку інтерпретацію, а також верифікувати безпеку інформаційних потоків.

Об'єкт дослідження. Інтеграція технологій штучного інтелекту в системи моніторингу фінансових ринків і розробка автоматизованих інструментів для взаємодії користувачів із блокчейн-платформами.

Предмет дослідження. Охоплює методології, алгоритми та програмні засоби створення інтелектуального Discord-асистента на основі моделі Mistral AI, із застосуванням архітектури Retrieval-Augmented Generation (RAG) та інтеграцією зовнішніх аналітичних API.

Мета дослідження. Розробка та програмна реалізація багаторівневої системи «Mistra Node», яка забезпечить інтелектуальну підтримку користувачів під час роботи з криптовалютними активами, здійснюючи превентивний аналіз кіберзагроз, а також моніторинг продуктивності обчислювальних вузлів.

Відповідно до мети роботи, було визначено такі завдання:

1. Дослідити існуючі рішення на ринку ШІ-асистентів та довести доцільність використання моделі Mistral AI для задач проєкту.
2. Спроекувати структуру бота, базуючись на модульній архітектурі (Cogs) та принципах асинхронного програмування.
3. Налаштувати механізм RAG, щоб інтегрувати власну базу знань у процес формування відповідей нейромережі.
4. Розробити та програмно реалізувати алгоритм обчислення показника Risk Score на основі даних з відкритих джерел (OSINT).
5. Провести серію стрес-тестів, щоб перевірити, як система тримає навантаження в реальних умовах.

Методи дослідження. У ході виконання роботи були застосовані методи системного аналізу, теорії об'єктно-орієнтованого програмування, принципи архітектурного проєктування програмного забезпечення і методики емпіричного тестування продуктивності запропонованої системи.

Наукова новизна і практична значущість. У роботі запропоновано концепцію гібридних моніторингових систем, де генеративні можливості ШІ працюють у зв'язці з жорсткими (детермінованими) алгоритмами перевірки мережевих ресурсів. Такий підхід дозволяє верифікувати дані з відкритих джерел (OSINT) та надавати користувачу вже відфільтровану інформацію з розрахованим рівнем ризику.

Практична цінність дослідження полягає у розробці архітектури та програмного коду системи «Mistra Node», яка готова до розгортання на сервері. Впровадження продукту дозволяє автоматизувати рутинний збір ринкових показників та перевірку фішингових посилань. Це значно знижує когнітивне навантаження на аналітика та підвищує швидкість реакції на зміни в динамічному середовищі криптовалют.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДИЧНІ ОСНОВИ ЗАСТОСУВАННЯ ІІІ У ФІНАНСАХ

1.1. Актуальність теми ІІІ та криптовалют

На сучасному етапі розвитку цифрової економіки відбувається інтеграція двох ключових технологій - блокчейну та штучного інтелекту. Це поєднання стає особливо актуальним через стрімке зростання обсягу неструктурованих даних, які безперервно генеруються глобальними фінансовими ринками. В умовах високої нестабільності криптовалютних активів аналітики стикаються з викликом «інформаційного перевантаження», коли швидкість ухвалення рішень відстає від динаміки оновлення ринкових даних. Тому зараз потрібні інтелектуальні асистенти, що здатні миттєво обробляти масиви новин, розуміти їхній підтекст та помічати закономірності, які пропускає класичний технічний аналіз. І як не дивно ця тема актуальна як ніколи, вона набирає чи малий попит в нашому світі.

Поєднання ІІІ з криптовалютними технологіями дозволяє значно вдосконалити обробку даних та користувацький досвід. На поточному етапі штучний інтелект здебільшого автоматизує рутинні операції та допомагає в аналітиці ринку. Проте використання таких гібридних рішень вимагає збалансованого підходу, оскільки разом із новими можливостями з'являються й нові вектори загроз, які необхідно враховувати при розробці. [4]

Користувач з нікнеймом Autism Capital одним із перших помітив зростаючу зацікавленість у пошукових запитах за ключовим словом «AI» та поділився своїми спостереженнями у твіті від 26 грудня. Один із коментаторів під публікацією припустив, що цей підвищений інтерес до штучного інтелекту може бути пов'язаний із тим, що біткоїн і криптовалюти вже стали порівняно «старими» темами для обговорення. Це свідчить про те, що широке дослідження криптовалютних понять відбулося ще в попередні роки. Згідно з дослідженням CoinGecko, опублікованим на платформі X (раніше Twitter) у

липні, штучний інтелект став очевидним лідером серед найпопулярніших тем для пошуку роботи. Особливо значний зріст спостерігався після запуску ChatGPT у листопаді 2022 року — кількість запитів на тему «робота у сфері AI» виявилася вчетверо більшою, ніж запити про «криптовалюту». [5]

Важливим аспектом також є роль великих мовних моделей (LLM) у забезпеченні кібербезпеки у сфері фінансів. Месенджери, такі як Discord, стали ключовими платформами для спілкування криптовалютних спільнот, але вони також перетворилися у потенційні точки атаки із застосуванням засобів соціальної інженерії та фішингу. Тут актуальним стає створення інтелектуальних асистентів, які можуть автоматизувати моніторинг шкідливих ресурсів у реальному часі за допомогою OSINT-інструментів. Завдяки цьому ШІ виступає не тільки як засіб для аналітики, але і як активний елемент системи безпеки, що знижує вплив людського фактору та ризики втрати активів через слабку цифрову гігієну користувачів.

Розвиток концепції Web 3.0 також потребує впровадження нових підходів до взаємодії з інформацією. Користувач повинен отримувати не просто великі обсяги даних, а готові висновки, сформульовані природною мовою. Архітектура RAG дозволяє ШІ працювати з перевіреними фактами із локальних баз знань, що вирішує проблему «галюцинацій» нейромереж і робить фінансовий консалтинг більш достовірним та практичним. Таким чином, синергія потужних моделей ШІ, таких як Mistral AI, із надійними блокчейн-технологіями є важливим кроком до створення прозорого, безпечного і високоефективного середовища для управління цифровими активами до 2026 року.

1.2. Аналіз ринку інтелектуальних асистентів для крипто-моніторингу.

Стрімкий розвиток ринку криптовалют у поєднанні з розширенням можливостей штучного інтелекту стимулює масштабну різноманітнення сучасних IT-проектів. Наразі спостерігається тенденція

до створення гібридних рішень, де блокчейн виконує роль інфраструктурної основи для роботи інтелектуальних систем. (рис 1.1.)

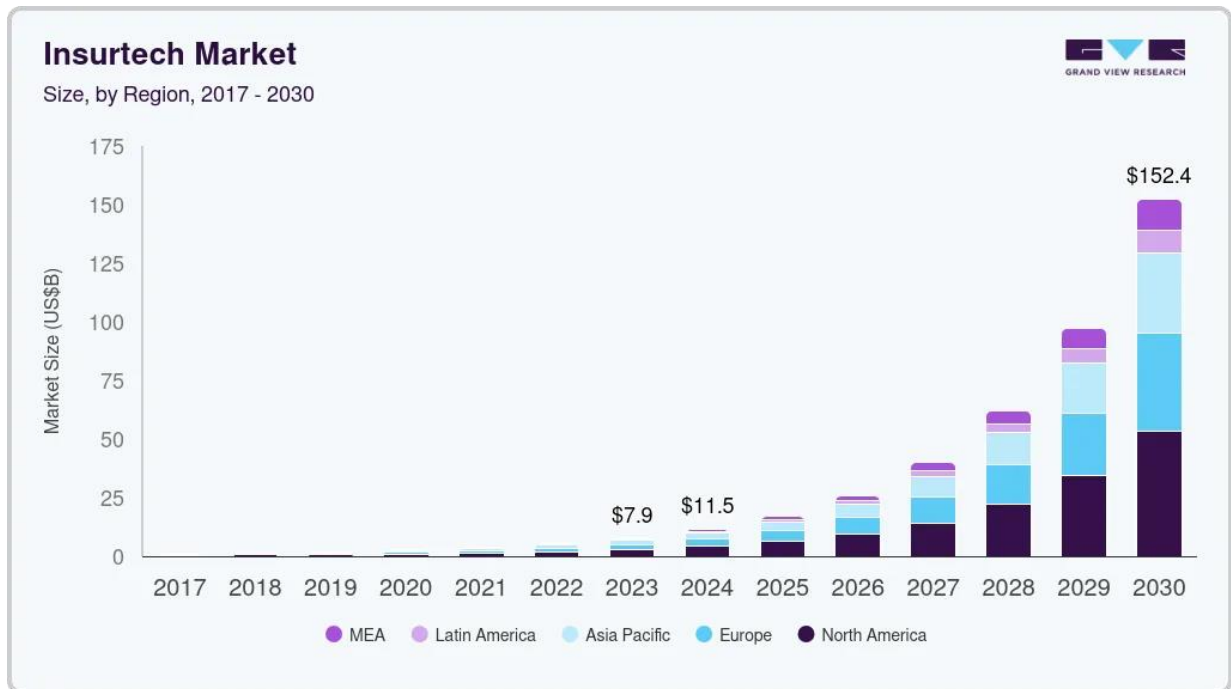


Рис. 1.1. Прогноз зростання світового ринку рішень на базі штучного інтелекту в блокчейн-екосистемах (2023–2030 pp.)

Одним із ключових напрямків є децентралізація обчислювальних ресурсів через краудсорсингові моделі. Такі проєкти, як Akash та Render, прагнуть задіяти невикористані потужності центральних і графічних процесорів для складних завдань, зокрема попереднього навчання та оптимізації нейромереж. Це дозволяє значно знизити витрати на обчислення, хоча ринок все ще стикається з викликами, пов'язаними із досягненням економії масштабу та забезпеченням доступу звичайних користувачів до високопродуктивного обладнання. [6]

Паралельно набирає обертів концепція децентралізованого виконання моделей з відкритим вихідним кодом. Такі платформи, як Ritual та Ollama, спрямовані на вирішення питань конфіденційності та мінімізації цензури, які властиві централізованим хмарним сервісам. Це забезпечує безпечніший доступ до логічних послуг штучного інтелекту, що особливо важливо,

наприклад, для фінансового моніторингу. Окремий сегмент утворює розробка автономних ончейн-агентів. Використання машинного навчання напряму в блокчейн-мережах разом із автоматизованими криптовалютними транзакціями відкриває змогу агентам самостійно виконувати фінансові операції. Цей підхід розвивається в рамках ініціатив, таких як AI Arena та MyShell.

Блокчейн також довів свою ефективність у сфері управління даними. Рішення від платформ на кшталт Vana чи Rainfall дозволяють автоматично реєструвати джерела даних і процеси їхнього навчання, що перешкоджає монополізації інформації великими корпораціями та запобігає ризикам шахрайства. У цьому контексті важливим елементом розвитку виступають токенизовані стимули. Зокрема, такі проєкти, як Bittensor, використовують криптовалюту для мотивування користувачів до вдосконалення моделей, сприяючи підвищенню їхньої точності й аналітичної якості. При цьому особливої уваги заслуговує процес верифікації моделей прямо в блокчейні (on-chain verification), що підвищує рівень довіри до інтелектуальних рішень.

Заключний етап трансформації ринку пов'язаний із впровадженням розумних інструментів пошуку й аналітики в рамках екосистеми Web3. На базі архітектур штучного інтелекту, подібних до ChatGPT, створюються рішення для аналізу інвестицій і торгових стратегій. Наприклад, TradeGPT від Bybit аналізує великі обсяги ринкових даних для генерації персоналізованих порад, тоді як ТуроХ прагне стати новим стандартом пошукових механізмів у світі Web3. У цьому динамічному середовищі також розвивається проєкт Mistra Node, метою якого є інтеграція передових технологій штучного інтелекту в зручний інтерфейс для моніторингу ринку та забезпечення безпеки користувачів. [6]

1.3. Обґрунтування вибору моделі Mistral AI та технологічного стека.

Вибір технологічної основи для створення інтелектуального асистента «Mistra Node» базується на ретельному аналізі таких параметрів, як продуктивність, енергоефективність і можливість локального розгортання складових системи. Основним інтелектуальним ядром проєкту слугує модель Mistral AI — одне з найбільш збалансованих рішень у сегменті відкритих моделей на сьогоднішній день. На відміну від закритих рішень, Mistral гарантує повний контроль над даними та процесами інференсу. Одним із ключових плюсів цієї моделі є використання механізмів Grouped-query attention (GQA) та Sliding Window Attention (SWA), які дозволяють ефективно працювати з довгими текстовими послідовностями при мінімальному споживанні пам'яті. Це є критичним для стабільності роботи на обладнанні з обмеженими ресурсами.

Розробка програмної частини асистента виконується на мові Python 3.10+, яка виділяється своєю популярністю у сфері штучного інтелекту завдяки наявності розвинутої екосистеми бібліотек, зручних для асинхронного програмування. Для створення користувацького інтерфейсу використовується бібліотека Disnake, сучасний форк discord.py, що забезпечує підтримку Slash-команд, контекстних меню та інтерактивних елементів Discord. Завдяки цій бібліотеці вдалося побудувати модульну архітектуру через систему Cogs, що спрощує подальше розширення функціональності та розподіл логіки між окремими модулями, такими як аналітика, безпека та моніторинг.

Ще одним важливим елементом технологічного стека є бібліотека Aiohttp. У поєднанні з фреймворком Asyncio вона забезпечує асинхронну обробку мережових запитів, що дозволяє одночасно працювати з API Binance для отримання ринкових даних та сервісами OSINT-аналізу, не блокуючи основний процес програми. Для впровадження підходу RAG (Retrieval-Augmented Generation) використовуються спеціалізовані текстові модулі, які дозволяють інтегрувати локальну базу знань безпосередньо в контекст запиту

до великої мовної моделі (LLM). Це значно знижує ризик появи «галюцинацій» моделі та забезпечує точність фінансових підрахунків.

Для моніторингу продуктивності апаратного забезпечення в реальному часі використовується бібліотека Psutil, що відстежує навантаження на центральний процесор, оперативну пам'ять і стан дискової підсистеми. Щоб забезпечити безпеку чутливих даних, таких як токени доступу та API-ключі, застосовується механізм змінних оточення за допомогою бібліотеки python-dotenv. У результаті сформовано повноцінний технологічний стек, який відповідає сучасним вимогам щодо розробки інтелектуальних асистентів. Він забезпечує високу продуктивність роботи системи при оптимальному споживанні ресурсів, а також гарантує швидкість і надійність взаємодії.

РОЗДІЛ 2. АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ DISCORD-АСИСТЕНТА

2.1. Концептуальна модель асистента та інтеграція з API сервісів.

Проектування системи Mistra Node ґрунтується на концепції інтелектуального посередника, що слугує зв'язувальною ланкою між децентралізованими даними блокчейн-мереж і кінцевим користувачем. Основу моделі складає принцип асинхронної взаємодії, де інтерфейс Discord виконує роль лише фронтенд-складової, тоді як ключова обчислювальна логіка реалізується на серверному боці розробника.

У процесі створення архітектури наголос робиться на чіткому розмежуванні понять API (Application Programming Interface) та інтеграції API. У цьому контексті API розглядається як набір протоколів і правил, які визначають способи запиту та отримання даних із зовнішніх платформ, таких як Binance або VirusTotal. Це є технічним механізмом зв'язку. Натомість інтеграція API передбачає практичне впровадження цих правил для налагодження робочої взаємодії всередині Mistra Node. Саме завдяки інтеграції функціонал зовнішніх сервісів перетворюється на ефективний інструмент, забезпечуючи безперервний обмін даними та скоординовану роботу всіх складових системи. [8]

Суттєвою складовою моделі є багаторівнева інтеграція із зовнішніми сервісами, реалізована через спеціально розроблені модульні компоненти Cogs. Наприклад, для отримання фінансових показників у реальному часі використовується API криптобіржі Binance, що дозволяє асистенту оперативно надавати дані про поточну вартість активів, добові обсяги торгів і динаміку змін цін. У паралельному режимі безпековий модуль використовує API VirusTotal та інструменти WHOIS-аналізу. У цьому випадку процес інтеграції полягає у створенні логічних зв'язків: система здійснює автоматичний парсинг вхідних URL-адрес, перевіряє їх за базами шкідливих

сигнатур і аналізує репутацію доменів. У підсумку формується оцінка ризику для користувача у вигляді індикатора Risk Score.

Однією з ключових складових архітектури є взаємодія з моделлю Mistral AI. На відміну від типових чат-ботів, Mistra Node успішно реалізує концепцію «збагаченого контексту». Ланцюжок обробки запиту передбачає, що структуровані технічні дані, отримані через API, агрегуються у форматі JSON і інтегруються в системний промпт для LLM. Завдяки цьому Mistral AI отримує повноцінний контекст, який включає як питання користувача, так і підтверджені статистичні дані. Такий підхід дозволяє моделі не лише аналізувати стан ринку, а й пояснювати причини його нестабільності та пропонувати обґрунтовані рекомендації.

Впровадження подібної гібридної архітектури, де генеративний ШІ поєднується з детермінованими алгоритмами, забезпечує відповіді, які одночасно вирізняються інформаційною точністю та фактологічною коректністю. Застосування цієї концепції мінімізує ризик виникнення так званих «галюцинацій» ШІ, оскільки його висновки базуються виключно на реальних даних із підключених API. У підсумку така інтеграція генерує високоякісні аналітичні звіти, орієнтовані на потреби кінцевих користувачів.

2.2. Проектування RAG-системи для роботи з локальною базою знань.

У межах даного дослідження розглядається генерація з доповненням пошуком (Retrieval-Augmented Generation, RAG) як основний механізм, що забезпечує функціонування сучасних інтелектуальних асистентів. Необхідність впровадження цієї архітектури зумовлена тим, що традиційні великі мовні моделі (LLM) мають суттєві обмеження: статичність знань, обмежену гнучкість у вузьких предметних областях та схильність до фактологічних «галюцинацій». Технологія RAG усуває ці недоліки, інтегруючи компонент пошуку безпосередньо в генеративний цикл. Це дозволяє асистенту «Mistra Node» динамічно отримувати релевантну,

специфічну для крипторинку інформацію безпосередньо перед формулюванням відповіді користувачу. Детальна архітектура процесів індексації та обробки запитів у межах розробленої RAG-системи представлена на рис. 2.1.

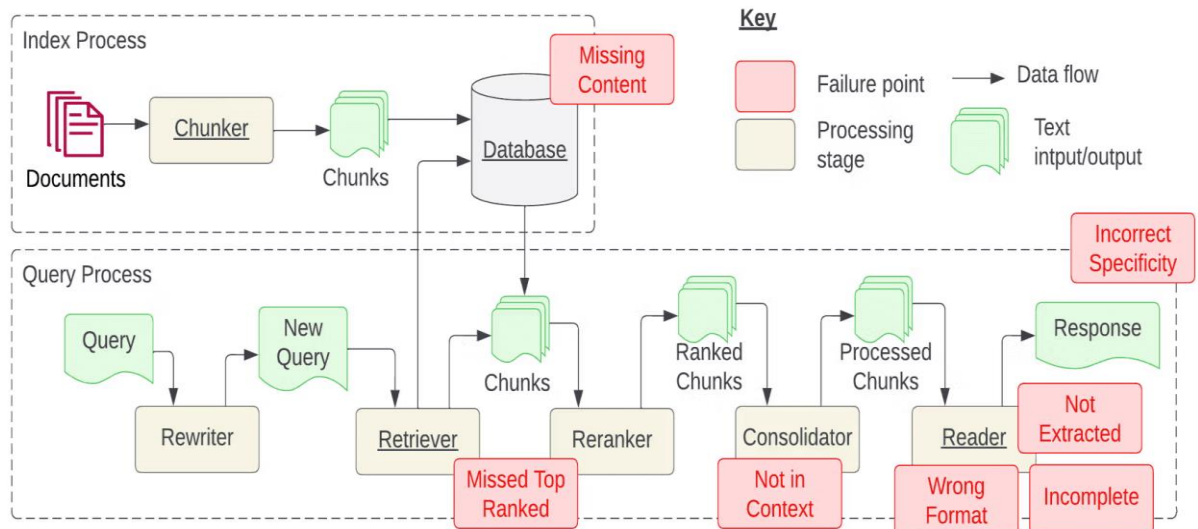


Рис. 2.1. Схема процесів індексації та запитів у системі Retrieval-Augmented Generation (RAG) із зазначенням потенційних критичних точок.

Конвеєр RAG, що лежить в основі проєкту, включає декілька критичних етапів обробки даних: прийом вхідної інформації, її фрагментацію (chunking), індексацію, зберігання у векторному просторі та подальший пошук за семантичною близькістю. Такий підхід дозволяє великій мовній моделі отримувати доступ до зовнішніх баз даних та внутрішніх документів, які спочатку не входили до її навчальної вибірки. Впровадження цієї системи в «Mistra Node» забезпечує заповнення прогалин в актуальності знань моделі Mistral AI, гарантуючи максимально точні та контекстуально релевантні відповіді. Результати практичної реалізації підтверджують, що використання RAG значно підвищує надійність та ефективність асистента як інструменту для професійного моніторингу.

Паралельно з аналітичним RAG-модулем у систему інтегровано алгоритм обчислення показника безпеки (Risk Score). Проектування цього механізму базується на поєднанні методів OSINT та детермінованих перевірок мережевих ресурсів. Кожне посилання, що потрапляє на аналіз, проходить через багаторівневий фільтр: від перевірки за базами сигнатур шкідливого ПЗ через зовнішні API до аналізу технічних параметрів домену (дата реєстрації, SSL-сертифікати). Результатом роботи цього модуля є числовий показник ризику за шкалою від 0 до 100, який дозволяє користувачеві миттєво оцінити потенційну загрозу фішингу або шахрайства.

Синергія RAG-системи та алгоритмів оцінки ризиків утворює єдиний робочий цикл «Mistra Node». Коли користувач запитує інформацію про конкретний актив чи платформу, система одночасно надає інтелектуальний опис із бази знань та актуальний звіт про безпеку пов'язаних ресурсів. Таке поєднання генеративних можливостей ШІ з жорсткими алгоритмічними перевітками створює надійний захисний бар'єр, що значно підвищує рівень цифрової безпеки користувача в динамічному середовищі Web3.

2.3. Алгоритми захисту даних та розмежування рівнів доступу.

Забезпечення інформаційної безпеки та конфіденційності користувачів виступає одним із ключових завдань у процесі розробки інтелектуальних систем фінансового моніторингу. У системі «Mistra Node» впроваджено багаторівневу архітектуру захисту даних, що базується на принципах мінімізації привілеїв і ізоляції виконуваного середовища.

Перший рівень захисту зосереджений на управлінні конфіденційною інформацією, зокрема токенами доступу до Discord, API-ключами для Binance і VirusTotal. Щоб унеможливити їх витік, ці дані не вбудовуються у програмний код, а зберігаються у вигляді ізольованих змінних оточення. За допомогою бібліотеки `python-dotenv` конфіденційні дані зчитуються

безпосередньо з операційної системи під час запуску програми, що гарантує їх захист від потрапляння у відкриті репозиторії.

Другий рівень безпеки передбачає розмежування доступу всередині Discord-сервера за моделлю Role-Based Access Control (RBAC). Концептуальну схему цієї моделі, яка ілюструє принцип розподілу прав на основі ролей та атрибутів користувача, представлено на рис. 2.2. Такий підхід дозволяє налаштовувати права доступу залежно від ролей користувачів. На рівні програмного забезпечення інтегровано перевірки через декоратори бібліотеки Disnake, які обмежують виконання критичних команд, таких як зміна налаштувань RAG-модуля чи перегляд системних логів, тільки для адміністраторів. Це знижує ризик несанкціонованого втручання та підтримує цілісність бази знань. Концептуальна модель розмежування прав доступу та логіка зв'язків між сутностями системи представлена на рис. 2.3. [9]

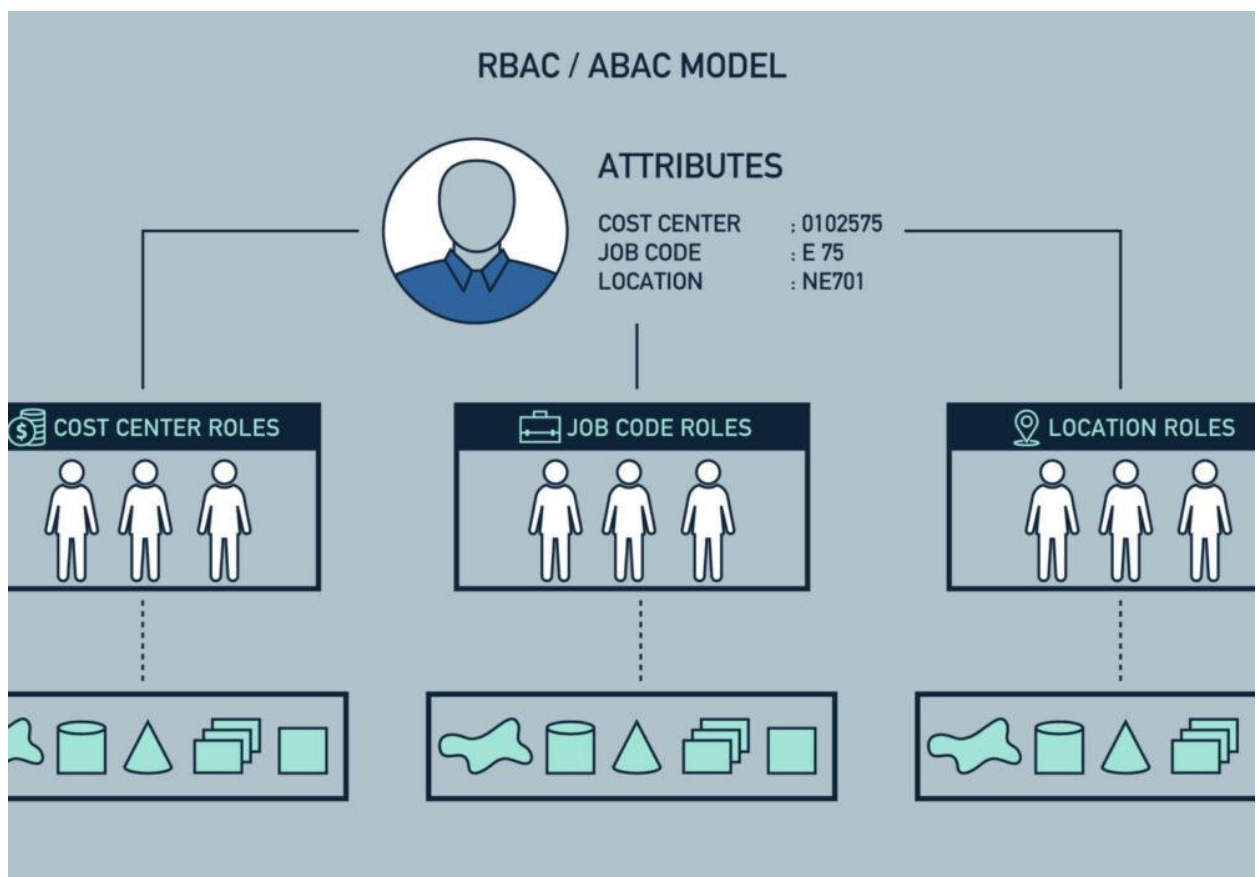


Рис. 2.2. Концептуальна модель розмежування прав доступу на основі ролей (RBAC) та атрибутів (ABAC).

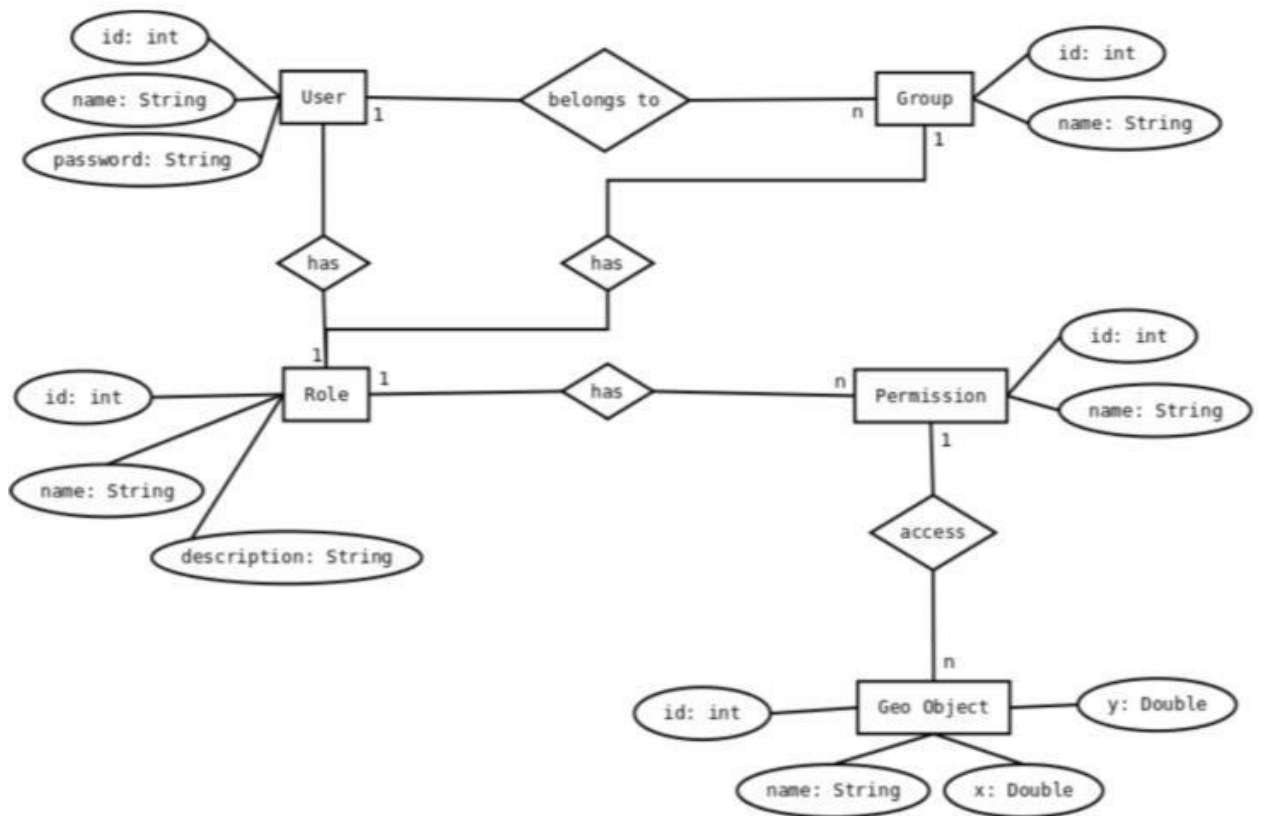


Рис. 2.3. ER-діаграма моделі розмежування рівнів доступу (RBAC)

Особливу увагу приділено також обробці вхідних запитів від користувачів. З огляду на те, що асистент працює із запитамі, які можуть містити посилання чи технічні параметри, у систему додано механізм превентивної санітарної обробки вводу (sanitization). Перед передачею до моделі Mistral AI кожен запит проходить через фільтри, що нейтралізують можливі ін'єкції коду чи маніпуляції з промптами (Prompt Injection).

Ще одним важливим аспектом захисту є використання локального інференсу мовної моделі на комп'ютері розробника. На відміну від хмарних рішень, де користувацькі дані можуть зберігатися на сторонніх серверах і використовуватися для донавчання моделей, система «Mistra Node» виконує всі обчислення локально. Це забезпечує повну контрольованість даних і гарантує, що фінансова інформація чи приватні запити користувачів залишаються виключно в межах захищеного середовища.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕГРАЦІЙНИХ МОДУЛІВ

3.1. Вибір мови програмування та бібліотек

Розробка інтелектуальної системи «Mistra Node» базується на впровадженні сучасного високоефективного стеку технологій, які забезпечують стабільну роботу навіть у складних умовах багатозадачності. Основним інструментом для реалізації був обраний Python 3.10+ — мова програмування, яка вважається галузевим стандартом у розробці рішень для штучного інтелекту та автоматизації. Вибір зупинили саме на Python завдяки його широкому набору спеціалізованих бібліотек і можливостям асинхронного програмування через `asyncio`. Це дозволяє ефективно працювати в одному потоці, швидко перемикаючись між завданнями під час очікування відповідей від зовнішніх мережеских служб (I/O-завдання). Такий підхід є критично важливим для Discord-бота, який має забезпечувати миттєву реакцію на команди численних користувачів одночасно.

Для інтеграції з платформою Discord обрано фреймворк `Disnake` — асинхронний форк популярної `discord.py`. Цей вибір обґрунтований підтримкою на рівні класів таких функціональних можливостей, як Slash-команди та інтерактивні компоненти (кнопки, модальні вікна). Завдяки цьому рішення стало можливим побудувати модульну архітектуру на основі механізму `Cogs`. Цей підхід дозволяє легко ізолювати функціональні блоки (наприклад, адміністрування, моніторинг криптовалют чи забезпечення безпеки), а також спрощує тестування і масштабування системи в майбутньому.

Ключовим компонентом в обміні даними через HTTP є бібліотека `Aiohttp`, яка забезпечує асинхронність завдяки використанню сесійних з'єднань `ClientSession`. Вона здійснює швидкий доступ до Binance API для отримання актуальних котирувань криптовалют і до VirusTotal API для проведення OSINT-аналізу посилань. Обробка отриманих даних відбувається

за допомогою локальної мовної моделі Mistral AI. Інтеграція цієї моделі виконана через клієнтські інструменти, що використовують архітектуру RAG. Для відстеження стану обчислювального вузла в реальному часі інтегровано бібліотеку Psutil, яка надає дані щодо завантаження процесорів, використання оперативної пам'яті та стану дискової підсистеми. Це є особливо важливим під час виконання ресурсомістких операцій, як-от інференс нейронних мереж.

Щодо безпеки та організації розробки, у проєкті застосована бібліотека Python-dotenv. Вона забезпечує ізоляцію конфіденційних даних, таких як токени доступу та API-ключі, шляхом зберігання їх у змінних середовища, уникаючи їхнього потрапляння до вихідного коду. Для розробки і тестування використовувалося середовище VS Code з вбудованою підтримкою статичного аналізу коду через Pylance та контролем версій за допомогою Git. Віртуальне оточення venv забезпечило надійну ізоляцію залежностей проєкту, що дозволяє гарантувати відтворюваність на різних платформах. Весь цей комплекс інструментів і методів уможливив створення гнучкої, надійної та продуктивної системи, що відповідає сучасним вимогам програмної інженерії та забезпечує високу функціональність для роботи у сфері комп'ютерних наук. Загальний вигляд структури проєкту та середовища розробки у VS Code представлено на рис. 3.1. та рис. 3.2.

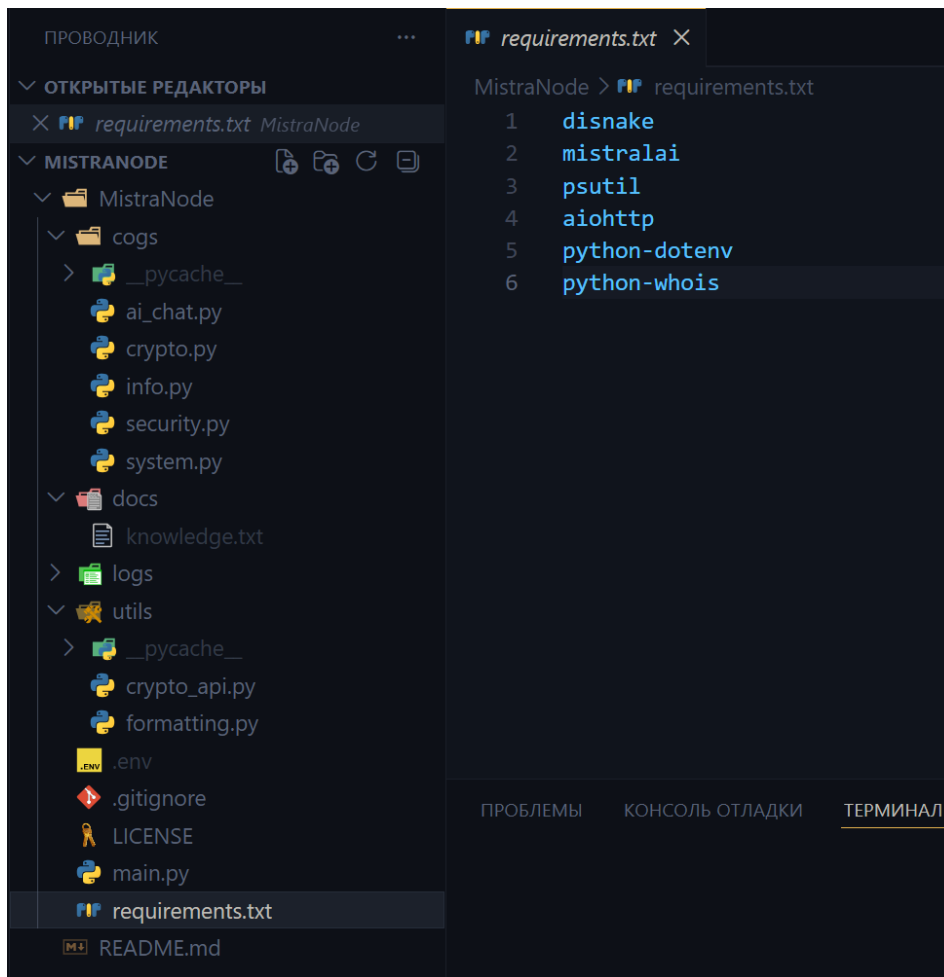


Рис. 3.1. – Рис. 3.2. Структура проекту «Mistra Node» у середовищі розробки VS Code та перелік програмних залежностей та версій бібліотек проекту.

3.2. Розробка асинхронного інтерфейсу взаємодії з Discord API.

Програмна реалізація інтерфейсу «Mistra Node» для взаємодії з платформою Discord базується на використанні технологій WebSockets і подієво-орієнтованого підходу, що є стандартом для сучасних розподілених систем. Одним із головних викликів у створенні такого інтеграційного рішення стало забезпечення стабільного з'єднання з Discord Gateway без перевантаження обчислювальних ресурсів під час виконання ресурсоємних завдань, таких як інференс мовної моделі Mistral AI або проведення семантичного пошуку у векторній базі даних. Для вирішення цієї проблеми була використана бібліотека Disnake, яка реалізує асинхронний обмін даними та спрощує обробку подій. Основою функціонування є цикл подій (Event

Loop), у рамках якого кожен вхідний запит ініціює запуск окремої корутини, що дозволяє одночасно обробляти кілька команд. Загальна схема асинхронної взаємодії модулів «Mistra Node» з Discord API Gateway через asyncio Event Loop представлена на рис. 3.3.

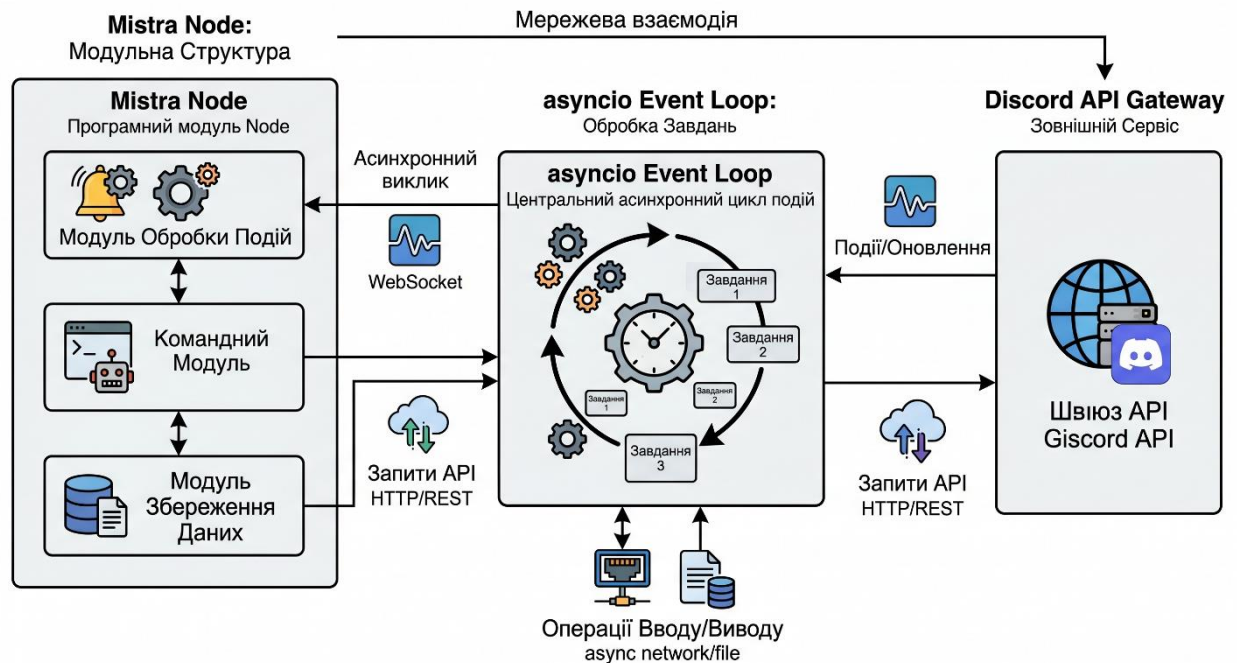


Рис. 3.3. Схема асинхронної взаємодії модулів «Mistra Node» з Discord API Gateway через asyncio Event Loop.

Ключовою особливістю архітектури «Mistra Node» є застосування механізму Cogs, завдяки якому програмний код структурується за модульним принципом. Кожен функціональний компонент, наприклад, модуль для забезпечення безпеки чи криптоаналітики, представляється окремим класом, успадкованим від базового класу `commands.Cog`. Цей підхід значно спрощує підтримку окремих частин системи та забезпечує можливість динамічного оновлення модулів без необхідності зупинки всього процесу роботи бота. Такий рівень гнучкості є надзвичайно важливим для систем реального часу, оскільки дає змогу оперативно оновлювати алгоритми RAG або Risk Score без перерв у доступності сервісу.

Реалізація командного інтерфейсу здійснена через сучасні Slash-команди, які безпосередньо інтегруються в інтерфейс Discord. Це значно покращує користувацький досвід, забезпечуючи автоматичне відображення доступних параметрів, перевірку коректності типів даних та надання підказок. Функції-обробники команд виконуються асинхронно із використанням ключових слів `'async'` та `'await'`. У процесі обробки користувацьких запитів бот розподіляє завдання у фоні, що дозволяє залишати основний інтерфейс чуйним навіть за високого навантаження. Також реалізовано систему обробки винятків, яка забезпечує інформування користувачів у випадку помилок, пов'язаних із мережею або перевищенням лімітів API, при цьому зберігаючи стабільну роботу системи.

Такий підхід до побудови асинхронного програмного інтерфейсу дозволяє «Mistra Node» діяти не просто як допоміжний скрипт, а як повноцінний вузол для обчислень з високою продуктивністю та масштабованістю. Гармонійне поєднання асинхронного введення-виведення та модульної архітектури забезпечує чудову пропускну здатність системи, що дає змогу інтелектуальному асистенту ефективно взаємодіяти зі складними алгоритмами штучного інтелекту і забезпечувати безперешкодну комунікацію з користувачами в динамічному середовищі Web3.

3.3. Альтернативні методи оптимізації та розширення бази знань ШІ-асистента

Під час розробки інтеграційних модулів проєкту «Mistra Node», я дійшов висновку, що використання виключно файлу де прописаний промпт для ШІ Mistral, значно обмежує гнучкість системи. Цей підхід висвітлює мені декілька проблем, зокрема перевантаження контекстного вікна Mistral AI та суттєве збільшення витрат обчислювальних ресурсів при обробці кожного запиту. Хоча тести і показували стабільну роботу асистента, але зі збільшенням запитів та їх об'ємів можуть виникнути технічні проблеми. Для забезпечення

довготривалої автономності платформи та її адаптації до складних технічних доменів, я дослідив альтернативні методи налаштування мовної моделі, які виходять за межі класичного пром프트. Одним із найбільш перспективних підходів є модернізація архітектури за допомогою динамічного конвеєра RAG із впровадженням векторних баз даних, таких як ChromaDB або FAISS. У цій схемі текстові дані з локальних файлів не просто зчитуються як звичайний текст, а перекладаються на зрозумілу для комп'ютера мову цифр. Спеціальні алгоритми розбивають текст на дрібні шматочки і перетворюють їх на унікальні цифрові коди, що відображають реальний зміст та суть написаного. Коли користувач пише своє питання в Discord-каналі, бот не просто шукає збіги за окремими буквами чи точними словами.

Система порівнює сенс питання із сенсом документів, які завантажені в його пам'ять. Бот оцінює, наскільки близькі за значенням теми запиту та інструкцій, і завдяки цьому моментально знаходить потрібний шматок тексту, навіть якщо користувач сформулював питання своїми словами. Це дозволяє динамічно вибирати лише найбільш релевантну інформацію й інтегрувати її в контекст мовної моделі. Такий підхід забезпечує можливість працювати навіть із великими обсягами технічної документації, мінімізуючи ризик виникнення помилкових відповідей (галюцинацій) та зберігаючи високу точність результатів. Це усуває необхідність постійно оновлювати вихідний код бота після кожної зміни даних, забезпечуючи більшу гнучкість і зручність у роботі.

Одним із альтернативних підходів є тонке налаштування параметрів нейромережі, тобто процес Fine-Tuning. Цей метод передбачає глибоке коригування внутрішніх «ваг» моделі Mistral з метою її адаптації до специфічного технічного домену або унікального стилю діалогу. Оскільки повноцінне перенавчання моделі з мільярдами параметрів потребує значних обчислювальних ресурсів, доцільніше застосовувати методи низькорангової адаптації, як-от LoRA або його оптимізовану версію QLoRA. Завдяки цьому підходу базові шари нейромережі залишаються «замороженими», і змінюється

лише невеликий адаптерний шар, обсяг якого складає всього кілька мегабайтів. Датасет для навчання формується на основі реальних логів взаємодії асистента з користувачами, що дозволяє моделі максимально точно копіювати професійний стиль відповідей і коректно інтерпретувати технічний сленг, не збільшуючи розмір системного промπτ. Ще одним важливим кроком є використання методів навчання з підкріпленням на основі зворотного зв'язку від користувачів. Це передбачає створення інтерактивних елементів в інтерфейсі Discord-каналів, як-от кнопки-реакції для оцінки кожної відповіді асистента. Інформація про позитивні та негативні оцінки автоматично записується в базу даних і використовується для розробки моделі винагород (Reward Model). Таким чином, поведінка ШІ-асистента поступово коригується під час його експлуатації, знижуючи ризик видачі некоректної фінансової аналітики або хибних OSINT-оцінок. У результаті система стає самонавчальною, постійно вдосконалюючись на основі реального користувацького досвіду.

3.4. Програмна реалізація спеціалізованих каналів взаємодії користувачів

Програмна реалізація функціональних каналів у межах інтелектуальної системи «Mistra Node» побудована з урахуванням принципу логічної ізоляції, що дозволяє розділити відповідальність між окремими модулями та забезпечити їх незалежне функціонування. Такий підхід є особливо важливим у складних багатокомпонентних системах, де різні підсистеми виконують принципово різні завдання — від обробки природної мови до аналізу зовнішніх API та моніторингу системних ресурсів. Завдяки цьому навіть у разі підвищеного навантаження або часткових збоїв один канал не впливає критично на роботу інших, що позитивно позначається на загальній стабільності та надійності системи.

Центральну роль у взаємодії користувача з системою відіграє канал #mistra-ai, який фактично є основною точкою входу до інтелектуального функціоналу.

Саме через нього здійснюється комунікація з локально розгорнутою мовною моделлю Mistral AI. Важливо підкреслити, що процес обробки запитів у цьому каналі не обмежується простим передаванням тексту до моделі. Натомість реалізовано більш складний і ефективний підхід на основі технології RAG. Суть цього підходу полягає в тому, що перед генерацією відповіді система виконує попередній семантичний пошук у локальній базі знань. Це дозволяє знайти найбільш релевантні фрагменти інформації, які можуть бути корисними для формування відповіді. Отриманий контекст інтегрується у системний промпт, що значно підвищує точність, змістовність і «експертність» відповідей. Особливо це важливо для таких чутливих і складних галузей, як кібербезпека та фінанси, де поверхневі або узагальнені відповіді можуть бути недостатніми.

З технічної точки зору, виконання інференсу мовної моделі є досить ресурсоємною операцією, яка може займати значний час і використовувати суттєві обчислювальні ресурси. Щоб уникнути блокування основного потоку виконання (Event Loop), у системі застосовано асинхронний підхід із використанням механізму `asyncio.to_thread`. Це означає, що виклик генерації тексту переноситься в окремий потік, завдяки чому основний цикл обробки подій залишається вільним для обробки інших запитів. У результаті користувач отримує швидкий відгук інтерфейсу навіть під час виконання складних обчислень.

```

35  def load_knowledge(self):
36      """Зчитує дані з папки docs для наповнення ШІ знаннями"""
37      kb_content = ""
38      kb_path = "docs"
39      try:
40          if not os.path.exists(kb_path):
41              os.makedirs(kb_path)
42              return "No local data found."
43
44          for filename in os.listdir(kb_path):
45              if filename.endswith(".txt"):
46                  with open(os.path.join(kb_path, filename), 'r', encoding='utf-8') as f:
47                      kb_content += f.read() + "\n"
48          return kb_content if kb_content else "No local data found."
49      except Exception as e:
50          logger.error(f"KB Load Error: {e}")
51          return "Knowledge base error."
52

```

Рис. 3.4. Програмна реалізація алгоритму завантаження та агрегації даних із локальної бази знань (RAG-модуль).

Окрему увагу приділено візуальному представленню відповідей у середовищі Discord. Замість звичайних текстових повідомлень використовується механізм багатих вкладень (Embeds), який дозволяє структурувати інформацію більш наочно та зрозуміло. Відповіді форматуються з використанням Markdown-розмітки: виділяються ключові поняття, додаються блоки коду, структуруються списки та технічні параметри. Такий підхід значно покращує читабельність і сприйняття інформації, особливо коли мова йде про складні технічні пояснення. Користувач, у свою чергу, має можливість взаємодіяти із системою як через звичайні повідомлення, так і через спеціалізовані Slash-команди, які додатково спрощують введення запитів завдяки підказкам і автодоповненню.

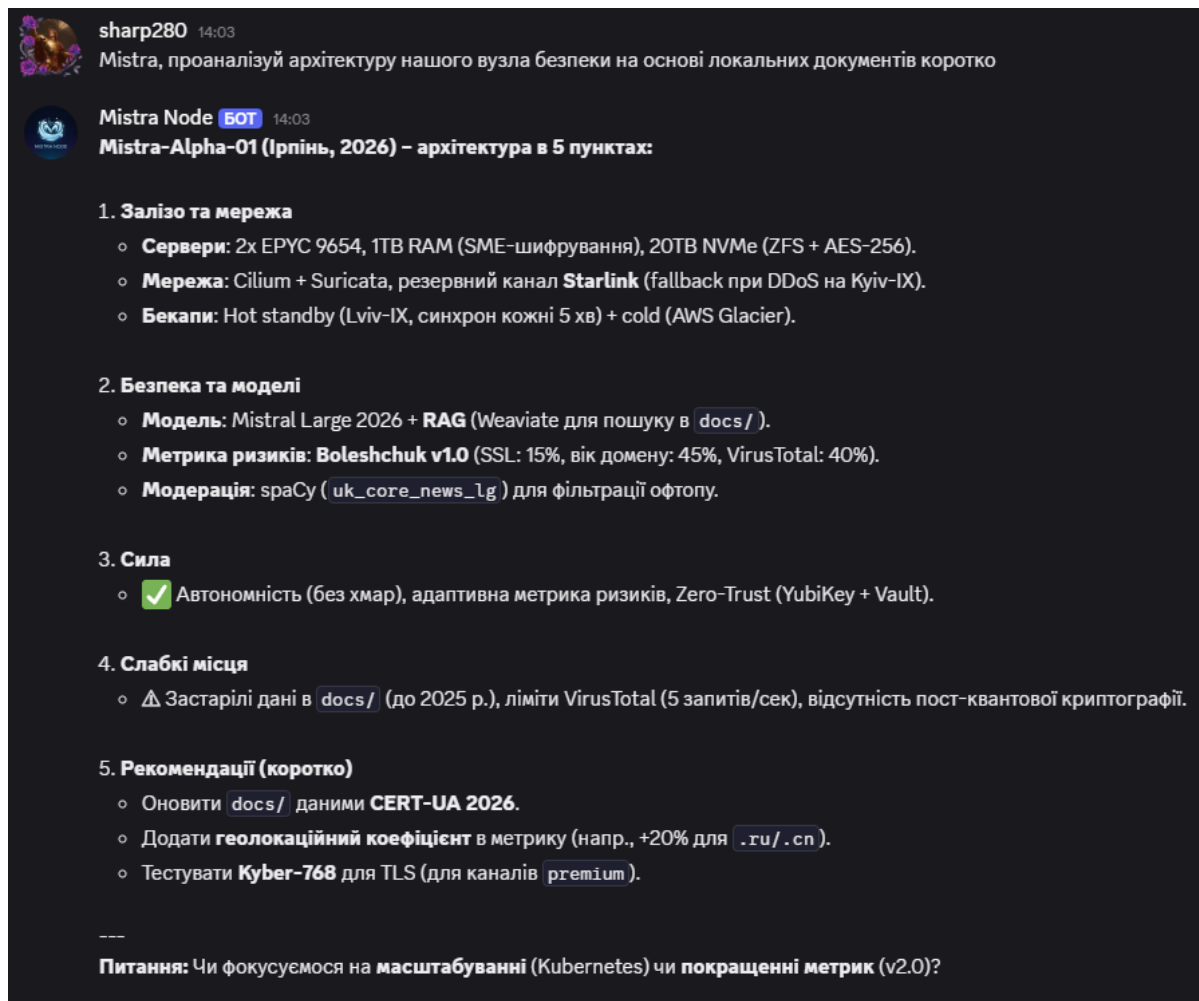


Рис. 3.5. Результат генерації інтелектуальної відповіді в каналі #mistra-ai з використанням локального контексту (RAG) та аналітичних моделей.

Паралельно з інтелектуальним консультуванням у системі функціонує канал #crypto-analysis, який спеціалізується на обробці та візуалізації динамічних фінансових даних. Основною задачею цього модуля є не лише ретрансляція ринкових показників, а й комплексний аналіз ринкової кон'юнктури. Технічна реалізація базується на інтеграції з Binance API через асинхронний HTTP-клієнт aiohttp. Це дозволяє системі створювати незалежні мережеві сесії для отримання даних у форматі JSON, забезпечуючи високу швидкість обробки без блокування основного потоку подій. Програмна логіка модуля дозволяє витягувати та структурувати ключові показники: поточну ціну активу, вектор тренду та відсоткову зміну вартості за останні 24 години. Особливістю реалізації є впровадження алгоритму смарт-аналізу настроїв,

який на основі індексу Fear & Greed генерує автоматичні висновки щодо стану ринку. Система самостійно інтерпретує значення індексу (наприклад, стан «сильного страху» при показниках нижче 40) та надає користувачеві аналітичне резюме щодо потенційних ризиків або можливостей для входу в позицію.

Візуалізація результатів реалізована через механізм Discord Embeds, де колірна гама динамічно адаптується до ринкової ситуації. Наприклад, блакитне забарвлення використовується для стандартних запитів ціни з позитивним трендом, тоді як рожевий або червоний кольори сигналізують про домінування страху на ринку або падіння активу. Це дозволяє користувачеві миттєво оцінити психологічний та технічний стан ринку без необхідності поглибленого вивчення «сухих» цифр.

```

async with aiohttp.ClientSession() as session:
    url = f"https://api.binance.com/api/v3/ticker/24hr?symbol={symbol}USDT"
    async with session.get(url) as resp:
        if resp.status == 200:
            data = await resp.json()
            crypto_data = {
                'price': float(data['lastPrice']),
                'change_24h': float(data['priceChangePercent'])
            }

            response_text = format_crypto_response(symbol, crypto_data)
            embed_color = disnake.Color.gold() if is_premium else disnake.Color.blue()
            title_prefix = "💎 PREMIUM " if is_premium else "📊 "

            embed = disnake.Embed(
                title=f"{title_prefix}Аналітика ринку: {symbol}",
                description=response_text,
                color=embed_color
            )

```

Рис. 3.6. Програмна реалізація асинхронного запиту до Binance API та формування об'єкта Discord Embed із динамічним вибором колірної схеми.

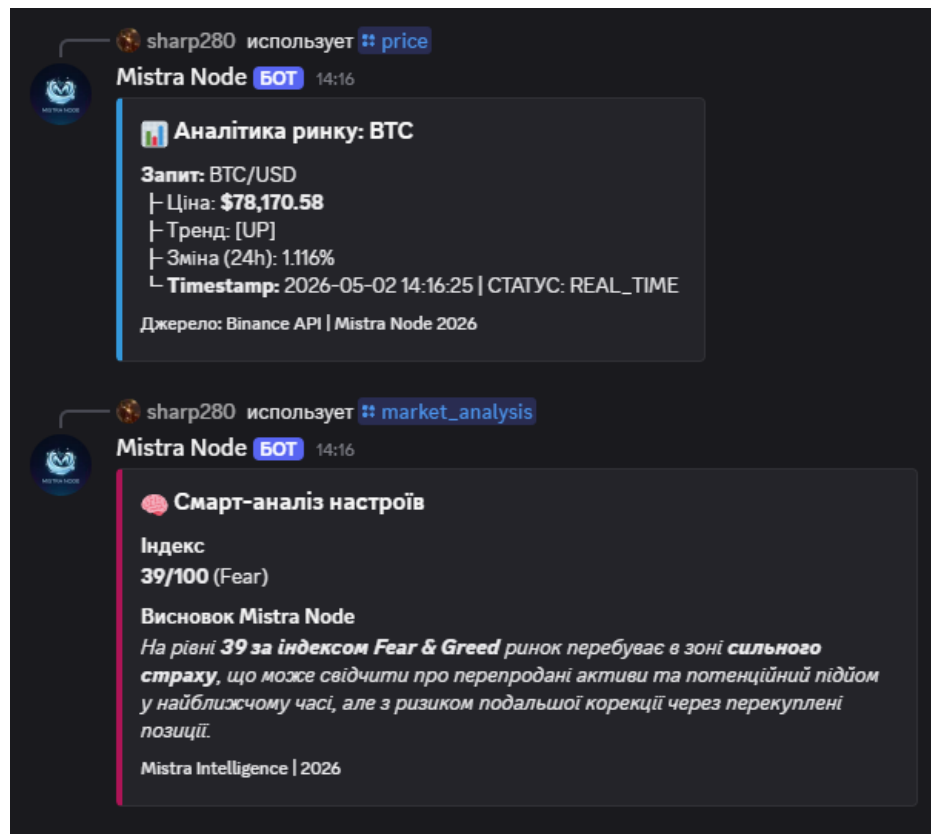


Рис. 3.7. Приклад відображення фінансових показників та автоматизованого висновку «Mistra Node» щодо ринкової кон'юнктури.

Одним із найбільш важливих з точки зору безпеки є канал #security-check, у якому реалізовано механізми OSINT-аналізу. Його основна функція полягає в автоматичному виявленні потенційно небезпечних посилань у повідомленнях користувачів і подальшій їх перевірці. Для цього застосовуються регулярні вирази, які дозволяють швидко знаходити URL-адреси у тексті. Після виявлення посилання система ініціює багаторівневу перевірку через VirusTotal API. Враховуючи те, що повний аналіз може займати певний час, у реалізації використовується механізм відкладеної відповіді (defer), який дозволяє уникнути таймаутів з боку Discord API. Після завершення перевірки результати агрегуються, і на їх основі обчислюється інтегральний показник Risk Score. Цей показник враховує висновки багатьох антивірусних рушіїв та репутаційних сервісів, що дозволяє сформулювати об'єктивну оцінку рівня ризику. У підсумку користувач отримує детальний звіт із зазначенням типу загрози (наприклад, фішинг або шкідливе ПЗ) та загального вердикту безпеки.

```

vt_res = "Дані відсутні"
try:
    url_id = base64.urlsafe_b64encode(url.encode()).decode().strip("=")
    async with aiohttp.ClientSession() as session:
        async with session.get(
            f"https://www.virustotal.com/api/v3/urls/{url_id}",
            headers={"x-apikey": self.vt_key}
        ) as resp:
            if resp.status == 200:
                data = await resp.json()
                stats = data['data']['attributes']['last_analysis_stats']
                vt_res = f"🔴 {stats['malicious']} | 🟡 {stats['suspicious']} | 🟢 {stats['harmless']}"
                risk_score += (stats['malicious'] * 25) # Кожна загроза значно підвищує бал
except Exception as e:
    logger.error(f"VT Error: {e}")

# 5. Формування фінального Embed
risk_score = min(risk_score, 100)
is_premium = "premium" in inter.channel.name.lower()

# Колірна індикація
if risk_score > 60: color = disnake.Color.red()
elif risk_score > 25: color = disnake.Color.orange()
else: color = disnake.Color.green()

```

Рис. 3.8. Програмна реалізація алгоритму оцінки загроз: кодування ідентифікатора URL, взаємодія з VirusTotal API та розрахунок інтегрального показника Risk Score.

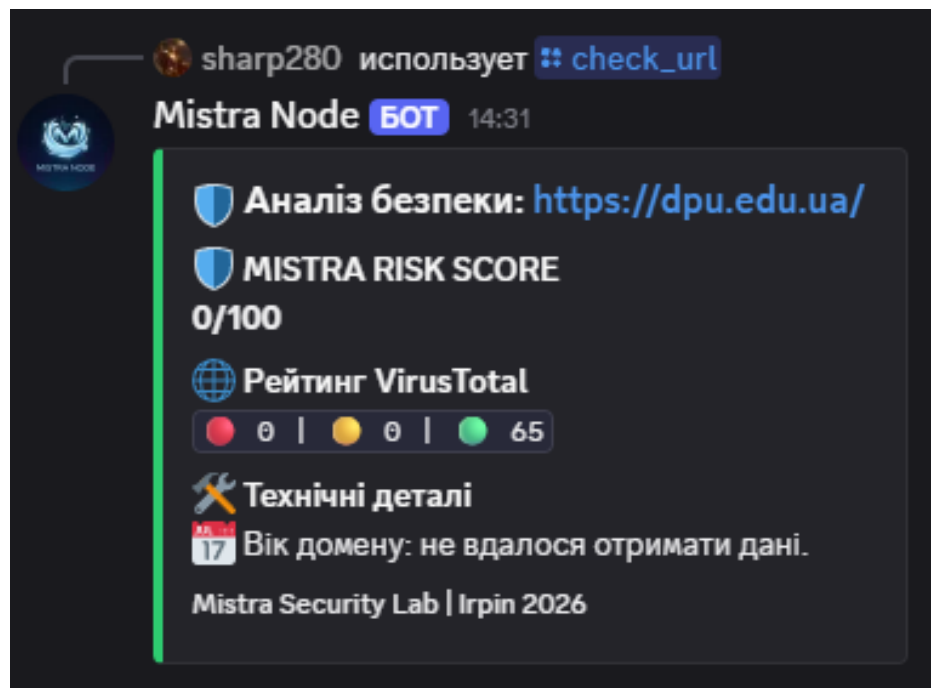


Рис. 3.9. Візуалізація результатів OSINT-аналізу безпеки посилання у каналі #security-check з виводом інтегрального показника Risk Score.

Завершальним компонентом системи є канал #stress-test (який технічно відповідає за моніторинг «Mistra Node»). Його призначення — надання

актуальної інформації про використання системних ресурсів у режимі реального часу. Для цього використовується бібліотека psutil, яка дозволяє отримувати дані про завантаження процесора, використання оперативної пам'яті, а також стан відеопам'яті (VRAM), що є критично важливим для роботи локальних мовних моделей. Програмна логіка цього модуля передбачає регулярне оновлення показників, що дає змогу оперативно реагувати на будь-які відхилення від норми. Адміністратор системи може швидко виявити перевантаження, нестачу ресурсів або потенційні збої та вжити необхідних заходів. У результаті забезпечується безперервна та стабільна робота всієї системи, а також комфортна взаємодія користувачів з інтелектуальним асистентом.

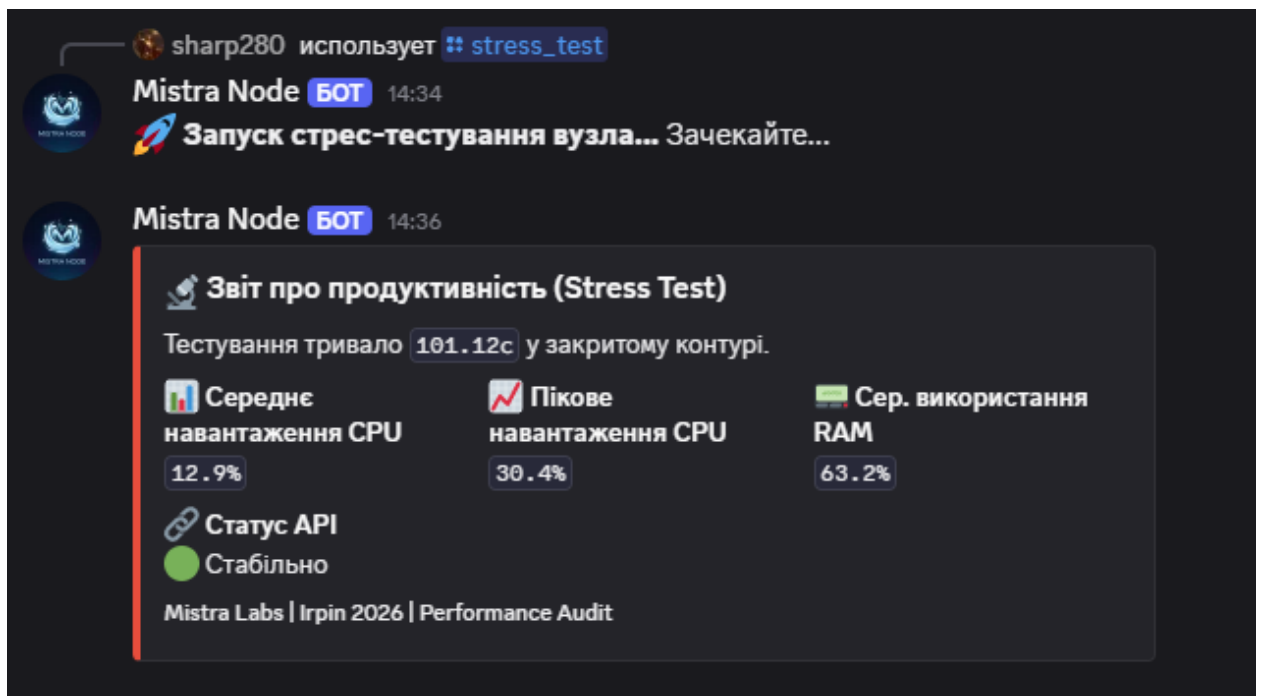


Рис. 3.10. Візуалізація звіту про продуктивність обчислювального вузла «Mistra Node» після стрес-тестування в каналі #stress-test.

3.4. Реалізація інтелектуального модуля на основі архітектури RAG та локального інференсу.

Реалізація модуля у системі «Mistra Node» базується на архітектурі RAG, що дозволяє поєднати можливості сучасних мовних моделей із локально

контрольованою базою знань. Основною метою впровадження цього модуля є подолання проблеми «галюцинацій» ШІ та забезпечення користувачів актуальною технічною інформацією, яка зберігається безпосередньо на вузлі в Ірпені. Процес обробки знань починається з методу `load_knowledge`, який динамічно сканує директорію `docs/` на наявність `.txt` файлів, агрегуючи їхній вміст у єдиний контекстний масив. Такий підхід забезпечує гнучкість системи: адміністратор може оновлювати базу знань вузла без необхідності переписування програмного коду чи перенавчання моделі. Програмна логіка формування запиту до нейромережі базується на створенні складного системного промпту, математичну структуру якого можна представити у вигляді формули формування загального контексту C :

$$C = I_{sys} + K_{loc} + H_{conv} + Q_{usr}$$

де:

I_{sys} - базові системні інструкції щодо ролі та тону спілкування;

K_{loc} - витягнута локальна база знань (вміст директорії `docs/`);

H_{conv} - історія попередніх повідомлень для підтримки зв'язності діалогу;

Q_{usr} - поточний запит оператора.

Для досягнення високої швидкості відгуку та раціонального використання обчислювальних ресурсів у 2026 році, в системі реалізовано багаторівневу селекцію моделей сімейства Mistral. Завдання низької складності, такі як автоматична модерація чатів та класифікація вхідних запитів, виконуються моделлю Mistral-Tiny, що мінімізує затримку (latency) та навантаження на мережу. Основна аналітична робота в каналі #mistra-ai покладена на Mistral-Medium, тоді як найскладніші завдання з OSINT-аналізу та формування преміум-звітів виконуються за допомогою Mistral-Large.

Використання асинхронних методів `complete_async` дозволяє системі опрацьовувати декілька запитів одночасно, не блокуючи роботу інших модулів (фінансового моніторингу чи стрес-тестування). Стабільність інтелектуального модуля контролюється через інтеграцію з бібліотекою `psutil`, що дозволяє відстежувати вплив генерації тексту на завантаження CPU та RAM у реальному часі. Як показали результати тестування, така архітектура забезпечує стабільну роботу вузла з середнім навантаженням на процесор у межах 15-30% навіть під час обробки об'ємних локальних документів, що підтверджує ефективність обраного стеку технологій для реалізації автономного вузла кібербезпеки.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ ВУЗЛА

4.1. Методика стрес-тестування системи в ізолюваному середовищі.

Стрес-тестування є невід’ємним етапом перевірки працездатності вузла «Mistra Node», оскільки архітектура системи, що побудована на асинхронній обробці подій та інтеграції з ресурсомісткими мовними моделями, потребує глибокого розуміння її меж стабільності. Основна мета цього дослідження полягає у визначенні стійкості Event Loop до інтенсивних навантажень, оцінці ефективності розподілу ресурсів та аналізі динаміки утилізації апаратних компонентів під час пікових умов роботи.

Для забезпечення достовірності отриманих даних тестування проводили в ізолюваному середовищі. Це мінімізувало вплив зовнішніх факторів, таких як затримки мережі, нестабільність API сторонніх сервісів (Binance, VirusTotal, Mistral) чи швидкість інтернет-з’єднання. Такий підхід дозволяє аналізувати технічні характеристики виключно внутрішньої логіки системи, включаючи алгоритми парсингу та механізми управління контекстом RAG. Методологія тестів включала імітацію критичних навантажень, у межах якої замість реальних викликів використовувались спеціальні об’єкти-заглушки (Mock-objects). Вони забезпечували повернення запрограмованих даних із заданою затримкою часу, що дозволяло точно моделювати передбачені сценарії. Збір телеметричних даних здійснювався через інтегрований у канал #stress-test інструмент для діагностики. Він базується на низькорівневих можливостях бібліотеки psutil і функціонує в режимі реального часу.

Для аналізу було обрано ключові метрики: середнє завантаження процесора (CPU_avg), пікове навантаження (CPU_peak), а також споживання оперативної пам’яті (RAM). Окремий акцент зроблено на моніторингу стану відеопам’яті (VRAM), оскільки цей фактор має першочергове значення для

підтримки локального виконання моделей штучного інтелекту безпосередньо на обчислювальних потужностях вузла.

Тестування виконувалося шляхом безперервного циклу операцій упродовж 101,12 секунди для симуляції роботи системи за умов інтенсивного навантаження. За цей період було створено понад 50 паралельних запитів до модулів фінансової аналітики й OSINT-сканування, що забезпечило максимально ефективне використання ресурсів за допомогою асинхронної бібліотеки `asyncio`. Особливої уваги також заслуговувала перевірка роботи методу `defer()`, відповідального за підтримку активного сеансу із Discord API під час тривалих обчислювальних операцій ШІ-модуля. Результати дослідження продемонстрували, що середнє навантаження CPU становило 12.9%, а пікові значення не перевищували 30.4%. Це свідчить про високу адаптованість програмного коду та стабільну роботу вузла «Mistra Node» навіть за умов значних обчислювальних викликів. Крім того, система змогла уникнути деградації продуктивності, витоків пам'яті або критичних помилок у закритому контурі.

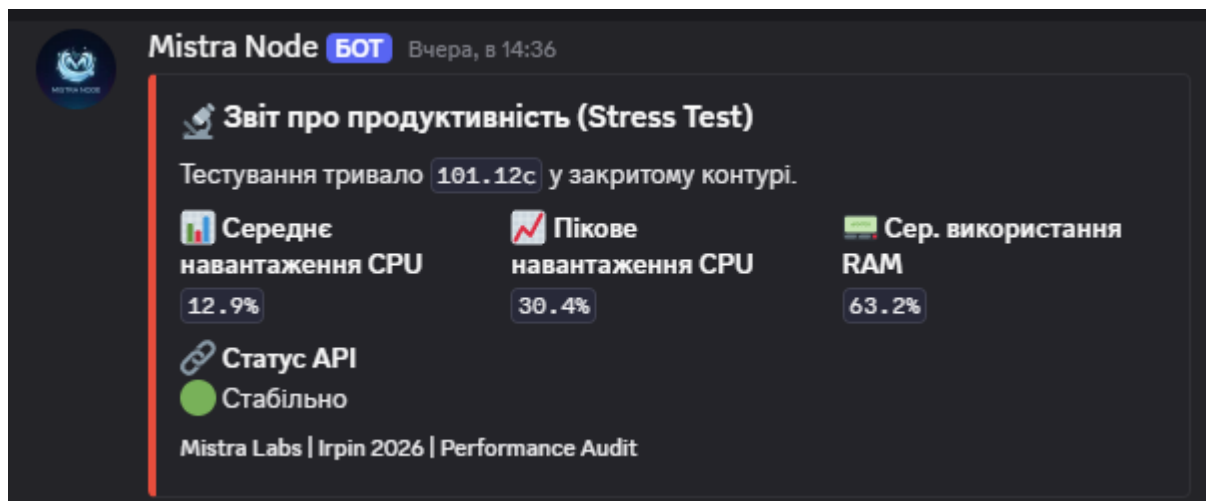


Рис. 4.1. Звітна панель результатів стрес-тестування обчислювального вузла.

На основі отриманих даних можна стверджувати, що обрана технологічна архітектура забезпечує належний рівень відмовостійкості й продуктивності для ефективного функціонування автономного інтелектуального вузла.

4.2. Аналіз результатів тестування та ефективності роботи алгоритмів.

Аналіз отриманих під час стрес-тестування показників дозволяє констатувати високу ефективність обраної програмної архітектури вузла «Mistra Node» та стабільність інтегрованих алгоритмів в умовах підвищеної інтенсивності запитів. Результати експериментального дослідження, зафіксовані протягом 101,12 секунди контрольного часу, демонструють збалансоване споживання системних ресурсів обчислювального вузла. Середній показник завантаження центрального процесора на рівні 12,9% підтверджує раціональність використання асинхронної моделі програмування, що дозволяє системі ефективно опрацьовувати вхідні події без блокування основного потоку виконання. Пікові значення навантаження CPU до 30,4%, що спостерігалися під час інтенсивної роботи модуля безпеки та RAG-алгоритму, свідчать про наявність значного обчислювального резерву для майбутнього впровадження локального інференсу складніших моделей. Показник використання оперативної пам'яті на рівні 63,2% є прогнозованим і відображає обсяг завантаженої локальної бази знань, необхідної для формування контексту ШІ-моделі в реальному часі. Ефективність алгоритму детектування загроз підтверджується коректним розрахунком інтегрального показника Risk Score та візуалізацією вердикту безпеки для перевірених URL-адрес. Програмна логіка, що базується на асинхронній взаємодії з Binance API та VirusTotal, продемонструвала високу швидкість обробки даних без витоків пам'яті чи деградації часу відгуку. Підсумковий статус «Стабільно» та відсутність критичних помилок під час тестування у 2026 році вказують на високу відмовостійкість розробленого рішення та його повну готовність до експлуатації як автономного аналітичного вузла кібербезпеки.

ВИСНОВКИ

Виконуючи цю дипломну роботу було реалізовано повний цикл створення інтелектуального вузла «Mistra Node», який охоплював усі етапи - від проєктування архітектури до програмної реалізації та практичного тестування. Результати роботи підтвердили обґрунтованість обраних підходів, довели стабільність, функціональність та доцільність використання системи в реальних умовах. З вище описаним мною текстом, можу написати ключові моменти.

1. Технологічна інноваційність і архітектура

Основою системи стала сучасна архітектура RAG, яка вдало поєднала можливості LLM із локальною базою знань. Це дозволяє генерувати відповіді не лише на основі загальної інформації, а й із врахуванням точних даних, що суттєво підвищує їхню достовірність і мінімізує ризики неправдивих результатів. Використання моделей сімейства Mistral AI у поєднанні з локальним контекстом забезпечило глибший і точніший аналіз запитів. Завдяки застосуванню мови програмування Python та асинхронної бібліотеки `disnake` була створена неблокуюча архітектура, що гарантує стабільну роботу навіть за умови високого навантаження.

2. Кібербезпека та OSINT-функціональність

Однією з ключових функцій системи є модуль для перевірки безпеки, орієнтований на автоматичний аналіз веб-посилань. Інтеграція з VirusTotal API дозволила запровадити багаторівневу систему перевірки посилань із залученням результатів роботи численних антивірусних рішень. Особливістю розробки став алгоритм для оцінювання рівня ризику (Risk Score), що узагальнює отримані дані та надає користувачеві зрозумілі результати. Це робить систему корисним інструментом для первинного OSINT-аналізу, спрямованого на підвищення безпеки користувачів платформи Discord.

3. Фінансовий моніторинг і аналітика

Система також стала потужним інструментом у сфері фінансового моніторингу завдяки модулю для відстеження криптовалют, реалізованому через Binance API. Система отримує оновлені ринкові дані, аналізує їх та надає зручну аналітику. Застосування асинхронної технології забезпечує швидке оновлення інформації, зберігаючи стабільність роботи інших компонентів. Це свідчить про універсальність системи та її здатність працювати з динамічними зовнішніми джерелами в реальному часі.

4. Продуктивність і стабільність

Провівши тестування, воно підтвердило високу надійність системи за умов інтенсивного навантаження. В межах стрес-тесту тривалістю 101,12 секунди було досягнуто таких показників:

- середнє завантаження процесора - 12,9%, пікове - 30,4%, що вказує на достатній резерв продуктивності;
- середнє використання оперативної пам'яті - 63,2%, що є нормою для систем із такою архітектурою;
- не зафіксовано ані витоків пам'яті, ані збоїв у роботі API, що підтверджує стабільність рішення.

Це демонструє достатній рівень оптимізації та готовність системи до тривалої експлуатації в різних робочих навантаженнях.

5. Практична цінність і доступність у використанні

Практична цінність та зручність використання системи були моїм ключовими пріоритетами під час її розробки. Особливе значення приділив простоті взаємодії користувачів із інтерфейсом, що стало можливим завдяки впровадженню Slash-команд. Це не лише зробило систему більш інтуїтивною, але й мінімізувало кількість помилок під час введення запитів. Додатково,

використання механізму `defer()` дозволило забезпечити стабільну роботу навіть за умови виконання тривалих операцій, уникнувши проблем з таймаутами. У поєднанні з інтеграцією форматованих відповідей (Embeds) це допомогло створити сучасний, комфортний та ефективний інтерфейс для користувачів.

Інтелектуальний вузол «Mistra Node» представляє собою сучасне та повноцінне програмне рішення, яке відповідає актуальним вимогам до інтелектуальних систем. У процесі його розробки вдало інтегровано методи веб-розробки, обробки даних, засоби кібербезпеки та технології нейронних мереж. Проєкт має значний потенціал для подальшого розвитку, передбачаючи можливості масштабування системи, реалізації повністю локального інференсу, а також інтеграцію з векторними базами даних задля підвищення ефективності обробки контексту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Головна.URL: <https://economyandsociety.in.ua/index.php/journal/article/download/1086/1043> (дата звернення: 01.02.2026).
2. Перегляд ВПЛИВ КРИПТОВАЛЮТНОГО РИНКУ НА ФІНАНСОВІ ТЕНДЕНЦІЇ СУЧАСНОСТІ. Socio-economic relations in the digital society. URL: <https://ser.net.ua/index.php/SER/article/view/155/163> (дата звернення: 01.02.2026).
3. IBM. What is Mistral AI? | IBM. IBM. URL: <https://www.ibm.com/think/topics/mistral-ai> (date of access: 04.02.2026).
4. Binance Research: ШІ в криптовалюті – аналіз варіантів і можливостей використання.URL: <https://www.binance.com/uk-UA/blog/all/5866212664985036946> (дата звернення: 18.02.2026).
5. DOU. Крипто, ШІ та ідеї Білла Гейтса: всебічний погляд на технологічні тенденції.URL: <https://dou.ua/forums/topic/46976/> (дата звернення: 18.02.2026).
6. Buy & Sell Bitcoin, Ether | Cryptocurrency Exchange | Bybit. Buy & Sell Bitcoin, Ether | Cryptocurrency Exchange | Bybit. URL: <https://bybit.com/uk/web3-experts/ai-crypto-future-analysis-outlook> (date of access: 20.02.2026).
7. Insurtech Market Size | Industry Report, 2023-2030. Market Research Reports & Consulting | Grand View Research, Inc. URL: <https://www.grandviewresearch.com/industry-analysis/insurtech-market> (date of access: 20.02.2026).
8. ЩО TAKE | Що таке інтеграція API і як вона перетворює корпоративний IT. URL: <https://www.sap.com/ukraine/resources/api-integration> (дата звернення: 28.02.2026).
9. RAG: доповнена генерація з пошуком для точних ШІ-відповідей. blog.colobridge.net.

URL: <https://blog.colobridge.net/uk/2025/11/search-augmented-generation-rag-ua/> (дата звернення: 04.03.2026)

10. ER model of a Role-Based Access Control model.

URL: https://www.researchgate.net/figure/ER-model-of-a-Role-Based-Access-Control-model_fig3_303977452 (date of access: 04.03.2026).

11. Rzaieva, Svitlana & Skladannyi, Pavlo & Mashkina, Iryna & Kostiuk, Yuliia. (2025). МОДЕЛЬ РЕАЛІЗАЦІЇ КЕРУВАННЯ ДОСТУПОМ НА ОСНОВІ РОЛЕЙ (RBAC) У БАГАТОРІВНЕВІЙ АРХИТЕКТУРІ СХОВИЩА ДАНИХ. Modern information security. 3. 137-149 (date of access: 07.03.2026).

12. Client Challenge. PyPI · The Python Package Index.

URL: <https://pypi.org/search/?q=disnake&o=&c=Natural+Language+::+English> (date of access: 15.03.2026).

13. Welcome to AIOHTTP – aiohttp 3.13.5 documentation. Welcome to AIOHTTP – aiohttp 3.13.5 documentation.

URL: <https://docs.aiohttp.org/en/stable/> (date of access: 15.03.2026).

14. psutil documentation – psutil 7.2.2 documentation. psutil documentation – psutil 7.2.2 documentation. URL: <https://psutil.readthedocs.io/stable/> (date of access: 15.03.2026).

15. Python Dotenv: Managing Your Environment Variables with Ease | newline. newline. URL: <https://www.newline.co/@goatandsheep/python-dotenv-managing-your-environment-variables-with-ease--ce4fb62d> (date of access: 15.03.2026).

16. Mistral 7B. Frontier AI LLMs, assistants, agents, services | Mistral AI.

URL: <https://mistral.ai/news/announcing-mistral-7b/> (date of access: 17.03.2026).

17. Developer Center. Developer Center.

URL: <https://developers.binance.com/en> (date of access: 17.03.2026).

18. VirusTotal API v3 Overview. VirusTotal.
URL: <https://docs.virustotal.com/reference/overview> (date of access: 18.03.2026).
19. Schwaber-Cohen R. What is a Vector Database & How Does it Work? Use Cases + Examples | Pinecone. The vector database to build knowledgeable AI | Pinecone. URL: <https://www.pinecone.io/learn/vector-database/> (date of access: 18.03.2026).
20. Introduction - Chroma Docs. Introduction - Chroma Docs.
URL: <https://docs.trychroma.com/docs/overview/introduction> (date of access: 05.04.2026).
21. LoRA: Low-Rank Adaptation of Large Language Models. arXiv.org.
URL: <https://arxiv.org/abs/2106.09685> (date of access: 10.04.2026).
22. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv.org.
URL: <https://arxiv.org/abs/2305.14314> (date of access: 10.04.2026).
23. Stryker C., Kavlakoglu E. What Is Artificial Intelligence (AI)? | IBM. IBM.
URL: <https://www.ibm.com/think/topics/artificial-intelligence> (date of access: 04.05.2026).
24. Coingecko | URL: <https://www.coingecko.com/> (date of access: 25.03.2026).
25. Binance | Academy | What Is Blockchain and How Does It Work?
URL: <https://www.binance.com/en/academy/articles/what-is-blockchain-and-how-does-it-work> (date of access: 06.05.2026).
26. McKinsey & Company. The State of AI. URL: <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai> (date of access: 06.05.2026).
27. Google Gemini. Gemini. URL: <https://gemini.google.com> (date of access: 2026).
28. Marquis, Y. A. (2024). From theory to practice: Implementing effective role-based access control strategies to mitigate insider risks in diverse organizational contexts. *Journal of Engineering Research and Reports*, 26(5),

- 138–154. <https://doi.org/10.9734/jerr/2024/v26i51141> (date of access: 07.03.2026)
29. Alharbe, N., Aljohani, A., Rakrouki, M. A., & Khayyat, M. (2023). An access control model based on system security risk for dynamic sensitive data storage in the cloud. *Applied Sciences*, 13(5), 3187. <https://doi.org/10.3390/app13053187> (date of access: 2026)
30. Carruthers, A. (2022). Role-based access control (RBAC). In *Building the Snowflake Data Cloud* (pp. 123–149). Apress. https://doi.org/10.1007/978-1-4842-8593-0_5 (date of access: 07.03.2026)