МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проєкт

здобувача освіти денної форми навчання
КБ.02.12.000.ДП

МІРЗАЛІЄВА
КИРИЛА СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка віртуального асистента для безпечного перегляду криптовалютної статистики

Проектний матеріал складається з пояснювальної записки на 78 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Дипломник _____ (Мірзалієв К.С.)

Керівник _____ (Скорняков В.С.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «21» червня 2025 р. Протокол ЕК № 2

Оцінка ЕК 4 (добре) / 85 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Мірзалієву Кирилу Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема проєкту *Розробка віртуального асистента для безпечного перегляду криптовалютної статистики*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі закінченого проєкту 16.05.25

3. Вихідні дані до проєкту *1. Використання номера гаманця криптовалюти Ethereum; 2. Основні вимоги до сервісу для взаємодії з біржею; 3. Використання технології блокчейн; 4. Мова програмування Python із застосуванням інтегрованого середовища розробки PyCharm; 5. Реалізація програмного модуля, що забезпечує відстеження транзакцій, баланс гаманця мережі Ethereum, курс криптовалют біржі та зміну мови інтерфейсу*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз технологій для реалізації віртуального асистента; Розробка загальної моделі безпечного перегляду криптовалютної статистики; Вибір технологій розробки; Розробка БСА обробки даних криптовалютної статистики; Розробка модулів застосунку віртуального асистента; Тестування роботи віртуального асистента для безпечного перегляду криптовалютної статистики

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Процес обробки транзакції у мережі Ethereum; Схема інтеграції чат-бота у Telegram; Схема взаємодії системи з джерелами даних криптовалютного ринку; Загальна діаграма роботи асистента; Діаграма основних засобів розробки та бібліотек проекту; БСА прийому нових блоків даних, обробки отриманої інформації, реєстрації адреси цільової мережі криптовалюти, Схема модуля обробки бізнес-логіки, Приклади роботи чат-боту та Деталі транзакції за даними Etherscan

6. Консультанти по проекту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Скорняков В.С.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 11.04.25

Керівник *Скорняков В.С.* _____
(підпис)

Завдання прийняв до виконання *Мірзаліев К.С.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проєктування*	15.05.2025	виконав
2	*Аналіз технічного завдання та пошук літератури*	16.05.2025	виконав
3	*Огляд функціональних елементів для реалізації моделі*	18.05.2025	виконав
4	*Аналіз вимоги до сервісу для взаємодії з біржею*	21.05.2025	виконав
5	*Вибір технічних та програмних засобів розробки*	23.05.2025	виконав
6	*Вибір технологій взаємодії з мережею криптовалюти*	26.05.2025	виконав
7	*Опис способів реалізації інтерфейсу користувача*	02.06.2025	виконав
8	*Опис алгоритму прийому блоків даних*	05.06.2025	виконав
9	*Опис алгоритму обробки даних*	06.06.2025	виконав
10	*Опис алгоритму роботи інтерфейсу*	09.06.2025	виконав
11	*Розробка модулів застосунку віртуального асистента*	10.06.2025	виконав
12	*Випробування застосунку та аналіз результатів*	12.06.2025	виконав
13	*Виконання економічних розрахунків*	13.06.2025	виконав
14	*Розробка питань з охорони праці та техніки безпеки*	13.06.2025	виконав
15	*Підготовка мультимедійної презентації проекту*	16.06.2025	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 12 000. 00 ДП	Дипломний проєкт		
А4			КБ 02. 12 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 12 000. 00 ДП			
Розробив		Мірзалієв К.С.		16.06	Розробка віртуального асистента для безпечного перегляду криптовалютної статистики	Літ.	Аркуш	Аркушів
Перевірив		Скорняков В.С.		16.06		Н Д П	4	78
Н. контр.		Петрашова В.І.		16.06		ВСП "ОТФК ОНТУ" гр.4КБ-02		
Затверд.		Кривченко Ю.В.		16.06				

ЗМІСТ

Вступ 7
1 Основний розділ 7
1.1 Аналіз технологій для реалізації віртуального асистента.....8
1.1.1 Короткий огляд технології блокчейн Ethereum.....8
1.1.2 Аналіз алгоритмів та механізмів безпеки криптовалюти Ефір.....11
1.1.3 Огляд засобів інтеграції чат-ботів у служби обміну повідомленнями.....14
1.2 Розробка загальної моделі безпечного перегляду криптовалютної статистики.....16
1.3 Вибір технологій розробки.....20
1.3.1 Технологія взаємодії з мережею криптовалюти.....20
1.3.2 Спосіб реалізації інтерфейсу користувача.....23
1.3.3 Вибір необхідних засобів розробки та бібліотек.....24
1.4 Розробка БСА обробки даних криптовалютної статистики.....27
1.4.1 Розробка БСА прийому нових блоків даних.....28
1.4.2 Розробка БСА обробки блоків інформації.....30
1.4.3 Розробка БСА реєстрації адреси цільової мережі криптовалюти.....31
1.5 Розробка модулів застосунку віртуального асистента.....34
1.5.1 Модуль інтерфейсу з користувачем.....35
1.5.2 Модуль обробки бізнес-логіки.....37
1.5.3 Модуль управління даними.....41
1.5.4 Реалізація структури проекту застосунку віртуального асистента.....46
1.6 Тестування роботи віртуального асистента для безпечного перегляду криптовалютної статистики.....48
2 Економічний розділ.. 55
2.1 Резюме 55
2.2 Визначення трудомісткості розробки програмного забезпечення.....55

2.3 Розрахунок ціни програмного продукту.....58
3 Розділ охорони праці та техніки безпеки……..…….…….…….60
3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста.....60
3.2 Гігієнічні вимоги до виробничого середовища.....60
3.3 Пожежна безпека.....63
Висновки.....65
Перелік використаних інформаційних джерел.....66
Додаток А. Фрагмент лістингу програмного модулю обробки транзакцій, балансу гаманця та хендлерів..... 67
Додаток Б. Слайди мультимедійної презентації..... 71

ВСТУП

У сучасному світі цифрових технологій криптовалюти набувають все більшої популярності як інструмент здійснення фінансових операцій та зберігання активів. Різке зростання їх використання супроводжується необхідністю забезпечення надійного захисту інформації, що надходить від криптовалютних бірж та інших джерел даних. Одним із викликів, які постають перед розробниками сучасних IT-систем, є створення зручних інструментів взаємодії з інформаційними потоками, що при цьому гарантують високий рівень безпеки даних.

В даному дипломному проекті розглядається концепція створення віртуального асистента для безпечного перегляду криптовалютної статистики. Основна ідея полягає у тому, щоб забезпечити користувачів оперативним доступом до актуальної інформації про криптовалюти через інтеграцію чат-бота у популярному месенджері Telegram.

Чат-бот забезпечує інтерфейс для доступу до даних, що надходять із зовнішніх API, зокрема з біржі CoinGecko та блокчейна Ethereum. Такий підхід дозволить створити зручний інструмент для моніторингу ринкових процесів, водночас суворо дотримуючись вимог щодо захисту конфіденційної інформації. Одним із основних технологічних рішень, обраних для реалізації проекту, є мова програмування Python. Її потужний функціонал, широкий асортимент бібліотек та можливість інтеграції з різними API роблять Python ідеальним вибором для розробки сучасного програмного забезпечення. Особливу увагу приділено інженерним підходам та використанню сучасних методів криптографічного захисту, що забезпечують стійкість системи до зовнішніх атак і несанкціонованого доступу.

Дипломний проект присвячений розробці віртуального асистента, який у поєднанні з технологіями чат-ботів дозволить користувачам безпечно та ефективно отримувати доступ до даних про криптовалюти. За допомогою запропонованого рішення створюється можливість моніторингу актуальних статистичних даних, що є важливою складовою сучасних фінансових сервісів, забезпечуючи при цьому високий рівень захисту інформації та користувацьких даних.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз технологій для реалізації віртуального асистента

Одним із центральних елементів проекту є розробка чат-бота, який інтегрується у популярний месенджер Telegram. Завдяки Telegram Bot API створення інтуїтивно зрозумілого та інтерактивного інтерфейсу стає реальністю, що дозволяє користувачам швидко реагувати на зміни в криптовалютній статистиці. Водночас, використання цього каналу взаємодії відкриває можливості масштабування та адаптації функціональності відповідно до запитів користувачів.

Безпека – один із найважливіших аспектів при розробці віртуального асистента для роботи з фінансовими даними. Для цього аналізуються сучасні методи криптографічного захисту, механізми аутентифікації, а також технології захисту API-запитів. Впровадження таких рішень дозволяє мінімізувати ризики несанкціонованого доступу до даних та забезпечує високий рівень стійкості системи до зовнішніх атак.

1.1.1 Короткий огляд технології блокчейн Ethereum

Ethereum є другою за значимістю блокчейн-платформою після Bitcoin, яка значно розширила функціонал класичного блочейн-механізму за рахунок впровадження смарт-контрактів. Смарт-контракти — це програмні алгоритми, що виконуються автоматично при дотриманні визначених умов, що дозволяє реалізувати децентралізовані додатки (dApps) та забезпечує безпечну обробку фінансових операцій без участі посередників.

Однією з ключових особливостей Ethereum є використання віртуальної машини Ethereum Virtual Machine (EVM), яка дозволяє виконувати програми, написані на спеціалізованих мовах програмування (наприклад, Solidity). Завдяки цьому розробники можуть створювати різноманітні додатки — від токенів і фінансових сервісів до складних організаційних структур, що працюють за принципом децентралізації.

Блокчейн Ethereum характеризується відкритою, публічною мережею, у якій кожна транзакція та виконання смарт-контракту записуються в незмінний та

перевірений журнал. Це забезпечує високий рівень прозорості та стійкість до змін як з боку зловмисників, так і внаслідок технічних збоїв. На практиці ці властивості знаходять широке застосування в рішеннях, які вимагають довіри до обробки і збереження важливої інформації.

У кожному блоці Ethereum зберігається низка важливих даних, серед яких:

– Заголовок (header): містить інформацію про попередній блок (parent hash), часову позначку (timestamp), значення nonce, ліміти газу та інші показники;
– Список транзакцій: містить дані про окремі операції користувачів, включаючи передачу ETH, виклики смарт-контрактів тощо;
– Структура стану: для забезпечення цілісності та відновлюваності даних використовується Merkle Patricia Tree (дерево Меркла), яке дозволяє ефективно зберігати стан мережі.

На рис.1.1 наведено умовну блок-схему, що ілюструє основну структуру блоку Ethereum. У табл.1.1 показано порівняння ключових параметрів Ethereum.

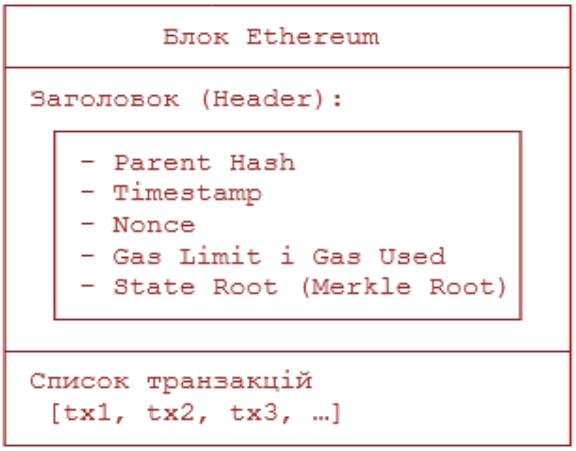


Рисунок 1.1. Основна структура блоку Ethereum

Таблиця 1.1. Порівняння ключових параметрів Ethereum

Показник	**Ethereum**
Тип блокчейну	Публічний, децентралізований
Основна валюта	Ether (ETH)
Консенсус	Первинно – Proof of Work (PoW); у процесі перехід до Proof of Stake (PoS)
Середній час блоку	~13 секунд (залежить від навантаження мережі)
Хеш-функція	Keccak-256
Смарт-контракти	Підтримка мов Solidity, Vyper та інших
Механізм оплати транзакцій	Газова модель: вартість транзакції обчислюється за формулою (див. нижче)

Для опису деяких ключових процесів у мережі Ethereum можна виділити наступні формули:

1. Обчислення вартості транзакції:

$$\text{Transaction Cost (ETH)} = \text{Gas Price (ETH/unit)} \times \text{Gas Used (units)} \quad (1.1)$$

Ця формула дозволяє визначити, скільки ETH буде сплачено за виконання певної транзакції, враховуючи встановлену в мережі ціну за газ та кількість використаного газу;

2. Хешування даних за допомогою алгоритму Keccak-256:

$$H(x) = \text{Keccak-256}(x) \quad (1.2)$$

Функція H(x) використовується для обчислення хешу даних, забезпечуючи цифрову підпис та цілісність інформації в блоках.

Процесу обробки транзакції у мережі Ethereum проілюстровано на рис. 1.2.

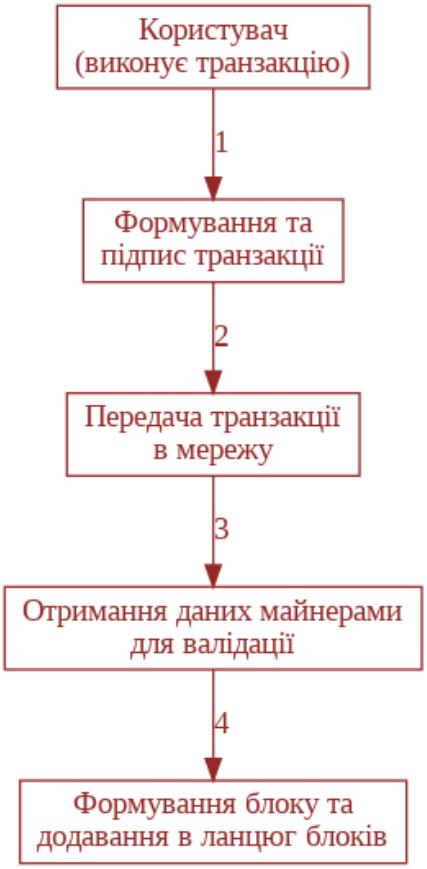


Рисунок 1.2. Процес обробки транзакції у мережі Ethereum

Мережа Ethereum забезпечує високий рівень децентралізації завдяки відкритій архітектурі, де кожен вузол має повну копію ланцюга блоків. Смарт-контракти дозволяють реалізовувати складні бізнес-логіки у вигляді автономних

програм, що виконуються після досягнення певних умов, що значно розширює можливості традиційних фінансових систем. Модель оплати газу слугує засобом запобігання безконтрольного використання мережевих ресурсів, оскільки кожна операція має свою вартість, що розраховується за вищезгаданою формулою.

Ethereum виступає як потужна платформа для розробки сучасних децентралізованих додатків. Його технологічна архітектура забезпечує гнучкість, масштабованість та високий рівень безпеки, що робить його ідеальним кандидатом для інтеграції у різноманітні фінансові сервіси, зокрема для створення віртуальних асистентів, які надають безпечний доступ до криптовалютної статистики.

1.1.2 Аналіз алгоритмів та механізмів безпеки криптовалюти Ефір

Ethereum забезпечує високий рівень безпеки завдяки інтеграції ряду криптографічних алгоритмів та структурних механізмів, що гарантують цілісність даних і автентичність транзакцій. У цьому підрозділі розглядаються основні алгоритми та механізми безпеки, які знаходяться в основі мережі Ethereum, а також їх застосування для захисту криптовалюти Ефір.

Однією з основних гарантів безпеки в Ethereum є використання алгоритму Keccak-256 для обчислення хешів. Цей алгоритм генерує 256-бітовий унікальний хеш для будь-яких вхідних даних, що забезпечує однонаправленість функції (тобто, зворотного відновлення вихідних даних неможливо) та високий рівень стійкості до колізій. Хешування за допомогою Keccak-256 є критично важливим для гарантування незмінності інформації, що зберігається у кожному блоці мережі.

Для перевірки автентичності та цілісності транзакцій Ethereum використовує цифрові підписи за допомогою ECDSA (Elliptic Curve Digital Signature Algorithm) з кривою secp256k1. Цей алгоритм дозволяє кожному власнику приватного ключа підписувати транзакції, гарантуючи тим самим, що лише власник може ініціювати операції зі своїми активами. Основний процес підписання включає такі етапи:

1. Обчислення хешу транзакційних даних: $z = H(m)$ де (m) – дані транзакції, а (H(m)) – результат хешування (Keccak-256);

2. Генерація випадкового числа (k) та обчислення (r): $r = (k \cdot G)_x \backslash modn$ де (G) – базова точка кривої, (n) – порядок групи, а $(k \cdot G)_x$ позначає x-компоненту

результату множення;

3. Обчислення значення (s): $s = k^{-1}(z + r \cdot d) \backslash mod n$ де (d) – приватний ключ користувача.

Після цих розрахунків утворюється цифровий підпис, що складається з трійки (*r, s, v*) (де v використовується для відновлення публічного ключа). Цей процес гарантує, що будь-яка зміна транзакційних даних призведе до невідповідності підпису, що унеможливлює підробку операцій.

На рис.1.3 подано діаграму, яка ілюструє процес цифрового підпису транзакції.

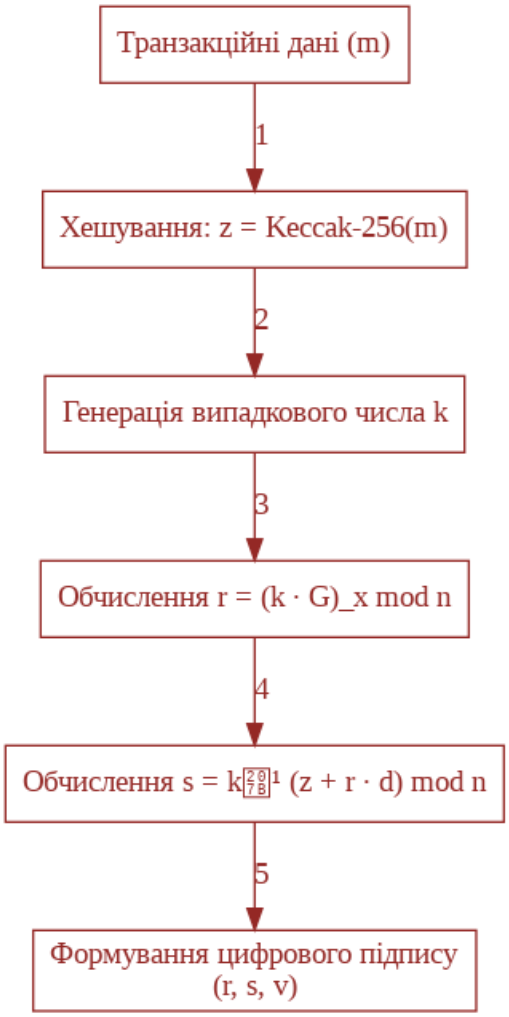


Рисунок 1.3. Діаграма процесу цифрового підпису транзакції

Для забезпечення цілісності та швидкої верифікації стану мережі Ethereum використовується спеціальний тип дерева – Merkle Patricia Tree. Ця структура об'єднує переваги класичного дерева Меркла та патріції-дерева, що дозволяє ефективно зберігати асоціативні дані та забезпечує наступні переваги:

- Ефективна валідація: дає змогу за допомогою невеликих доказів (Merkle proofs) підтвердити наявність або відсутність певної транзакції чи облікових даних;
- Незмінність даних: будь-яка спроба змінити дані призводить до зміни кореневого хешу, що вказує на компрометацію інформації.

У табл. 1.2 наведені основні характеристики цієї структури.

Таблиця 1.2. Основні характеристики структури Merkle Patricia Tree

Параметр	**Опис**
Тип даних	Асоціативний масив (ключ - значення)
Метод верифікації	Використання хешів для побудови кореневого значення (Merkle root)
Призначення	Забезпечення цілісності та швидкого доступу до стану мережі

Безпека Ethereum базується не лише на застосуванні криптографічних алгоритмів, але й на механізмах консенсусу, що забезпечують достовірність транзакцій у мережі. Первісно Ethereum використовував Proof-of-Work (PoW), який вимагав виконання складних обчислювальних завдань для додавання нового блоку. Цей підхід дозволяв захищатися від атак, таких як подвійні витрати, за рахунок високих обчислювальних витрат потенційних зловмисників.

З переходом до Proof-of-Stake (PoS) (у контексті Ethereum 2.0) підвищується не лише ефективність, але і рівень безпеки через:

- Економічні стимули: валідатори ризикують втратити свої активи у разі недобросовісної поведінки;
- Децентралізацію процесу консенсусу: розподіл відповідальності між великою кількістю учасників мережі.

Обидва механізми консенсусу сприяють створенню захищеного середовища, де кожен новий блок ретельно перевіряється і підтверджується великою кількістю незалежних вузлів, що мінімізує ризик компрометації мережі.

Аналіз алгоритмів хешування, цифрових підписів та використання спеціалізованих структур даних, таких як Merkle Patricia Tree, демонструє, що безпека мережі Ethereum забезпечується багаторівневим підходом. Сучасні

механізми консенсусу (PoW та PoS) в комплексі з криптографічними методами гарантують:

- Надійність обробки транзакцій;
- Захист від несанкціонованих змін;
- Високу стійкість системи до зовнішніх атак.

Цей комплекс заходів дозволяє Ethereum бути однією з найнадійніших платформ для розгортання децентралізованих додатків та криптовалютних операцій, що в контексті віртуальних асистентів забезпечує безпечну взаємодію з фінансовими даними.

1.1.3 Огляд засобів інтеграції чат-ботів у служби обміну повідомленнями

Інтеграція чат-ботів у служби обміну повідомленнями відкриває широкі можливості для створення інтерактивних сервісів, що забезпечують оперативний зв’язок з користувачами, автоматизацію рутинних завдань та впровадження кастомізованих бізнес-процесів. Сучасні платформи для месенджерів, такі як Telegram, WhatsApp, Facebook Messenger, Viber тощо, надають розробникам API, за допомогою яких можна побудувати інтерактивних віртуальних асистентів. Ці засоби дозволяють як реалізовувати базові функції обміну повідомленнями, так і інтегрувати зовнішні дані, а також забезпечувати зворотний зв’язок та обробку запитів у режимі реального часу.

У табл.1.3 показане порівняння ключових показників декількох платформ з підтримкою інтеграції чат-ботів. Серед усіх зазначених платформ, саме Telegram виділяється як оптимальне середовище для розробки чат-бота, особливо у випадку створення віртуального асистента для безпечного перегляду криптовалютної статистики. Основні аргументи на користь вибору Telegram включають:

- Відкритий та добре документований Bot API: Telegram надає розробникам можливість легко інтегрувати свій сервіс за допомогою HTTP-запитів. Документація API є розгорнутою, що дозволяє швидко розпочати розробку та адаптувати бот під конкретні потреби;
- Гнучкість архітектурних рішень: Інтеграція може здійснюватися як за

допомогою простого пулінгу, так і використанням webhook-технологій, що дозволяє оптимізувати роботу при високих навантаженнях. Це важливо для проектів, де обробка великих обсягів даних, таких як криптовалютна статистика, повинна проходити з максимальною швидкістю та без затримок;

- Розвинена екосистема бібліотек для Python: Для розробки Telegram-ботів існує безліч зручних бібліотек (наприклад, python-telegram-bot, pyTelegramBotAPI), що спрощують інтеграцію з іншими сервісами, забезпечують легке оброблення повідомлень та використовують сучасні засоби захисту даних. Використання Python, з огляду на його популярність і потужний функціонал, є оптимальним вибором для створення системи, що потребує високої надійності та захисту інформації;
- Безпека та масштабованість: Незважаючи на те, що бот-повідомлення не мають вбудованої end-to-end шифрованості, Telegram забезпечує додаткові засоби захисту через використання токенів аутентифікації та швидку обробку запитів, що дозволяє інтегрувати власні стратегії шифрування й валідації даних. Це важливо для проектів, які викликають роботу з конфіденційними фінансовими даними, такими як криптовалютна статистика.

Таблиця 1.3. Порівняння показників платформ з підтримкою чат-ботів

Платформа	**API Доступність**	**Ключові переваги**	**Обмеження**
Telegram	Повністю відкритий (Telegram Bot API)	Швидка інтеграція, гнучкість у розробці, підтримка як синхронного режиму (пулінг), так і webhook-технології	Бот-повідомлення не мають end-to-end шифрування
WhatsApp	Обмежений, частково комерційний	Висока популярність, масова аудиторія	Суворі обмеження API, складніший процес реєстрації
Facebook Messenger	Відкритий, але зі специфічними вимогами	Інтеграція зі службою Facebook, можливість комплексної розробки бізнес-чатів	Інтеграція може вимагати додаткових витрат на сертифікацію
Viber	Відкритий API	Підтримка інтерактивних функцій, можливості для маркетингових кампаній	Менший ринок користувачів порівняно з Telegram

На рис.1.4 наведена спрощена схема інтеграції чат-бота у Telegram, де показано ключові етапи взаємодії. Ця схема ілюструє, як Telegram Bot API виступає посередником між користувачем та серверною частиною застосунку, забезпечуючи зручну взаємодію, оперативність та можливість масштабування сервісу.

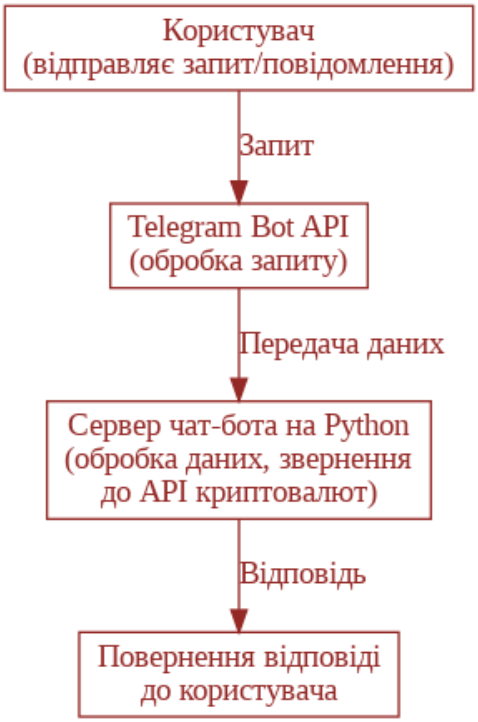


Рисунок 1.4. Схема інтеграції чат-бота у Telegram

Огляд засобів інтеграції чат-ботів у служби обміну повідомленнями показує, що завдяки широким можливостям API, гнучкості реалізації та активній підтримці розробниками, Telegram виступає оптимальним середовищем для створення віртуального асистента. Використання Telegram Bot API в поєднанні з потужними засобами розробки на Python дозволяє забезпечити оперативний, безпечний та масштабований доступ до криптовалютної статистики, що є критично важливим для сучасних фінансових сервісів.

1.2 Розробка загальної моделі безпечного перегляду криптовалютної статистики

Розроблювана система передбачає створення безпечного, зручного та інтуїтивного сервісу, який виступатиме посередником між цільовим користувачем та мережею криптовалюти. Основна ідея моделі полягає в тому, щоб забезпечити

оперативний доступ до актуальної статистики криптовалют, здійснюючи взаємодію з біржами та блокчейнами (наприклад, CoinGecko та Ethereum) через інтегрований інтерфейс, який реалізовано у вигляді Telegram-бота.

Важливою вимогою є забезпечення високого рівня безпеки як на рівні передачі даних, так і зберігання користувацьких даних, що досягається застосуванням сучасних методів криптографічного захисту та перевірених протоколів аутентифікації.

Загальна модель системи включає три ключові функціональні компоненти:

1. Інтерфейс користувача. Забезпечує інтуїтивну взаємодію користувача із сервісом через Telegram-бота, дозволяючи навіть користувачам без спеціальних знань в інформаційних технологіях оперативно отримувати потрібну інформацію. Особливості: Простота використання, адаптація до уподобань користувача, швидкий відгук системи, а також підтримка push-повідомлень щодо нових трансакцій або змін ринкової статистики;
2. Інтерфейс локальної бази даних. Модуль зберігання та обробки користувацьких даних, що включає реєстрації акаунтів (або збереження налаштувань для непрофесійного користування), історію запитів, переваги, а також історичні записи транзакцій чи статистики по обраних адресах. Особливості: Мінімізація об’єму збережених даних, оптимізація запитів до бази (застосування кешування, індексування) та забезпечення конфіденційності через криптографічне шифрування;
3. Інтерфейс роботи з мережею криптовалюти. Забезпечення взаємодії із зовнішніми джерелами даних криптовалютної мережі – отримання поточних курсів валют, даних про стан гаманців, транзакційного потоку та інших важливих показників. Особливості: Використання REST API та бібліотек для роботи з мережею (наприклад, web3.py для Ethereum), що дозволяють підтримувати високу швидкість оновлення інформації та стійкість до перевантажень.

На рис. 1.5. зображено схематичну діаграму взаємодії між основними елементами системи.

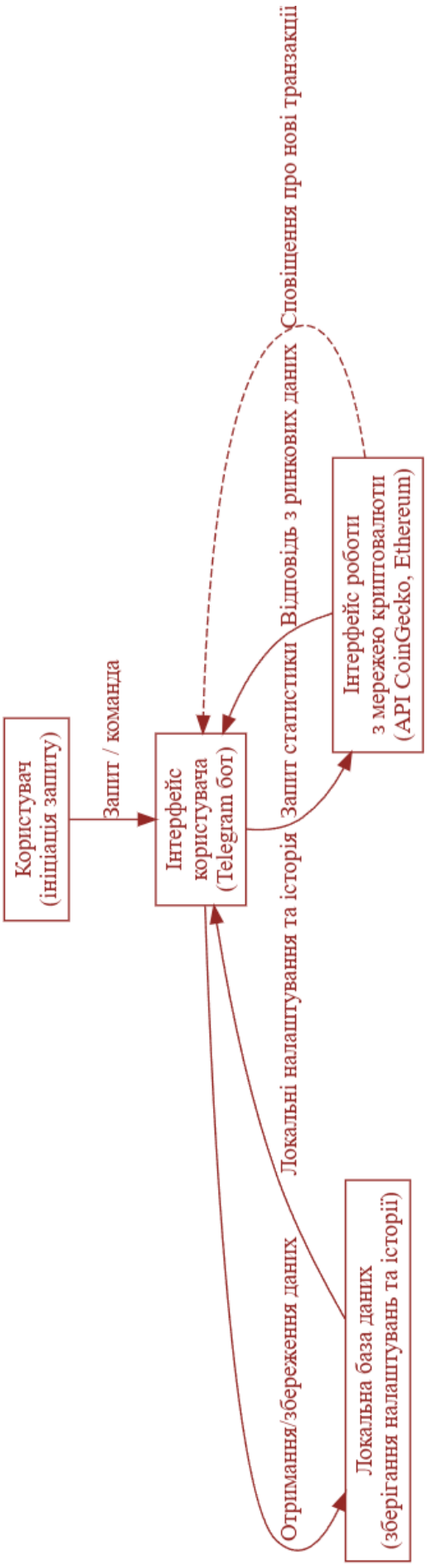


Рисунок 1.5. Схема взаємодії компонентів системи

На схемі взаємодії (рис.1.5) показані такі компоненти:

- Користувач (U): Роль користувача полягає у ініціації запиту через Telegram-бота, який являє собою інтерфейс системи;
- Інтерфейс користувача (UI): Цей компонент обробляє запити користувача, виконує попередню перевірку введених даних та відображає результати у зрозумілому форматі. Він також взаємодіє з локальною базою даних для завантаження користувацьких налаштувань і з мережею криптовалюти для отримання актуальної статистики;
- Локальна база даних (DB): Забезпечує збереження історії запитів, налаштувань та іншої важливої інформації, що дозволяє швидко відновлювати попередні дії користувача та оптимізувати роботу системи;
- Інтерфейс роботи з мережею криптовалюти (CN): Забезпечує безпосередню взаємодію із зовнішніми джерелами даних, де отримується інформація про стан криптовалютного ринку, поточні транзакції, курси та інші показники. Сповіщення про нові дані надходять до інтерфейсу користувача для оперативного оновлення інформації.

Основні функціональні вимоги розроблюваної системи включають:

- Надійність та безпека: Система повинна гарантувати конфіденційність користувацьких даних, захищаючи їх за допомогою криптографічних методів та протоколів шифрування;
- Висока швидкість обробки: Забезпечення конкурентоспроможної швидкості отримання та оновлення даних, що є критично важливим при роботі з фінансовими сервісами;
- Інтуїтивна взаємодія: Розроблений інтерфейс повинен бути зрозумілим навіть для користувачів з мінімальними знаннями у предметній області, що дозволить їм швидко орієнтуватися і експлуатувати сервіс без додаткових навчальних матеріалів;
- Масштабованість та адаптивність: Система повинна бути здатна працювати ефективно як зі збільшенням числа користувачів, так і при зростанні обсягу даних, що забезпечується модульною структурою.

1.3 Вибір технологій розробки

Для успішної реалізації проекту необхідно обрати такі технології, які забезпечать швидкість, надійність, безпеку та масштабованість системи. Обрання відповідних інструментів розробки має велике значення для ефективної інтеграції з різними зовнішніми сервісами та криптовалютними мережами, а також для створення зручного інтерфейсу.

При виборі технологій розробки орієнтувалися на наступні аспекти:

- Продуктивність та надійність: Система повинна швидко обробляти запити, забезпечувати оперативне оновлення даних і мати високий рівень стійкості до перевантажень.

- Безпека: Враховуючи природу роботи з фінансовими даними, важливо забезпечити надійний захист як при передачі інформації, так і при зберіганні особистих даних користувачів.

- Гнучкість інтеграції: Розробка проекту має передбачати взаємодію з зовнішніми API (наприклад, CoinGecko та Ethereum) і підтримку модульної архітектури, що дозволяє легко розширювати функціонал у майбутньому.

- Простота розробки та підтримки: Обрані технології мають бути добре документованими, популярними серед розробників та мати активну спільноту, що сприятиме швидкому пошуку рішень для можливих проблем.

1.3.1 Технологія взаємодії з мережею криптовалюти

Однією з критичних складових проекту є отримання актуальних даних криптовалютного ринку та взаємодія з мережею криптовалюти. Для цього було вирішено застосувати публічні API, які дозволяють отримувати дані за запитами (GET/POST), що мінімізує вимоги до апаратного забезпечення та забезпечує ефективну роботу системи за умови стабільного інтернет-з’єднання.

Для отримання загальної статистики криптовалют, такої як поточні ціни, ринкова капіталізація, обсяги торгів та інші показники, обрано сервіс CoinGecko. CoinGecko є однією з провідних бірж з відкритою базою даних, яка надає дані в реальному часі через свій публічний API. Серед основних переваг цього підходу

варто відзначити:

- Низькі вимоги до ресурсів: Використання CoinGecko API не вимагає від користувача високопродуктивної комп'ютерної системи, оскільки усі дані передаються за конкретним запитом до дистриб'ютора;
- Широкий спектр інформації: Платформа не лише надає актуальні ціни, але й допомагає орієнтуватись у величезній кількості монет завдяки класифікації та різноманітним точкам даних;
- Додаткові навчальні матеріали: CoinGecko пропонує підтримку нових користувачів, що є важливим аспектом для тих, хто вперше знайомиться з криптовалютами.

Для роботи з даними, які стосуються конкретно мережі Ethereum, таких як інформація про стан віртуальних рахунків, історія транзакцій та інші деталі блочного ланцюга, планується інтегрувати API від Etherscan. Основні переваги цього рішення наступні:

- Доступність до детальної інформації: Etherscan є провідною платформою для аналізу блокчейну Ethereum, забезпечуючи розробників можливістю отримання даних про транзакції, баланси та інші ключові показники;
- Гнучкість доступу: Сервіс пропонує як безкоштовні, так і платні (API PRO) плани, що дозволяють обрати оптимальний рівень доступу залежно від потреб проекту;
- Підтримка тестових мереж: Крім основної мережі Ethereum, Etherscan підтримує тестові мережі, такі як Kovan, Rinkeby і Ropsten, що сприяє проведенню тестування та розробки.

Обидва підходи – використання CoinGecko для загальної криптовалютної статистики та Etherscan для даних по мережі Ethereum – забезпечують комплексний доступ до потрібної інформації. Це дозволяє створити інтегровану систему, яка:

- Отримує статистичні дані за запитом, зменшуючи навантаження на апаратні ресурси.
- Забезпечує оперативне реагування на зміни на ринку.
- Гарантує рівноправний доступ до даних завдяки використанню

стандартизованих API.

На рис.1.6 подано схему взаємодії системи з джерелами даних криптовалютного ринку, що ілюструє, як система робить запити до CoinGecko та Etherscan, отримує відповідні дані та на їх основі інформує кінцевого користувача.

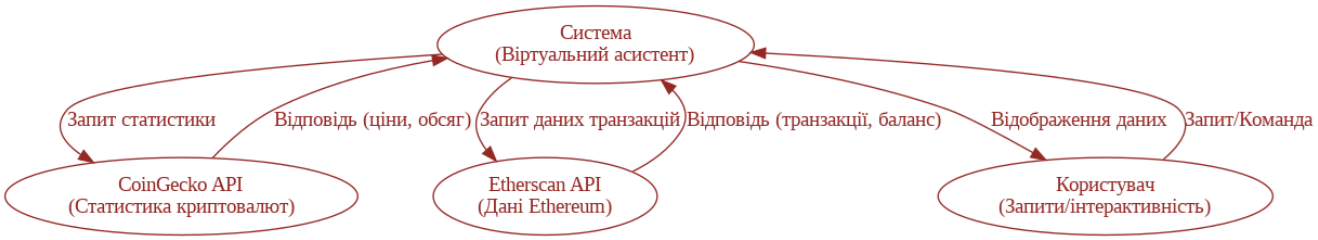


Рисунок 1.6. Схема взаємодії системи з джерелами даних криптовалютного ринку

Вибір технологій для взаємодії з мережею криптовалюти базується на поєднанні можливостей двох потужних інструментів: CoinGecko API для загальної криптовалютної статистики та Etherscan API для отримання даних мережі Ethereum. Такий комплексний підхід дозволяє забезпечити оперативний, безперебійний та безпечний доступ до даних крипторинку, що є основою для роботи віртуального асистента з перегляду криптовалютної статистики.

1.3.2 Спосіб реалізації інтерфейсу користувача

У цьому проекті реалізація користувацького інтерфейсу передбачає використання Telegram Bot API, що дозволяє організувати обмін повідомленнями між користувачем та серверною частиною додатка. Для подальшої розробки обрано наступні інструменти:

- Telegram Bot API: Цей API дозволяє реалізувати функціонал чат-бота для отримання запитів від користувача та відправлення відповідей. API забезпечує набір методів для роботи з повідомленнями, кнопками, inline-розширеннями та іншими елементами, що потрібні для реалізації інтерфейсу без інсталяції додаткового ПЗ клієнтом;
- Бібліотека python-telegram-bot: Використання даної бібліотеки у поєднанні з Python дозволяє швидко інтегрувати функціонал Telegram-бота у систему. Бібліотека підтримує асинхронну обробку запитів, спрощує роботу з webhook та пулінговим режимом, а також містить зручні інструменти для

форматування та структурування відповідей.

Реалізація інтерфейсу базується на наступній модульній схемі:

1. Обробка запитів користувача: Користувач взаємодіє з ботом через месенджер Telegram. Вхідні повідомлення надходять на сервер, де обробляються відповідними модулями;

2. Логіка обробки: Спеціально розроблені обробники (handlers) визначають тип запиту (наприклад, запит статистики криптовалют, запит операцій з рахунком) і викликають відповідні сервіси. Це може включати звернення до API криптовалютних сервісів або бази даних із збереженими налаштуваннями користувача;

3. Формування відповіді: Після обробки запиту генерується стандартизована відповідь, яка може включати текстові повідомлення, форматовану статистику або інтерактивні елементи (наприклад, кнопки для подальших дій);

4. Відправлення відповіді: Бот за допомогою Telegram Bot API відправляє повідомлення з відповіддю користувачу.

На рис.1.7 подано схему взаємодії між основними компонентами інтерфейсу.

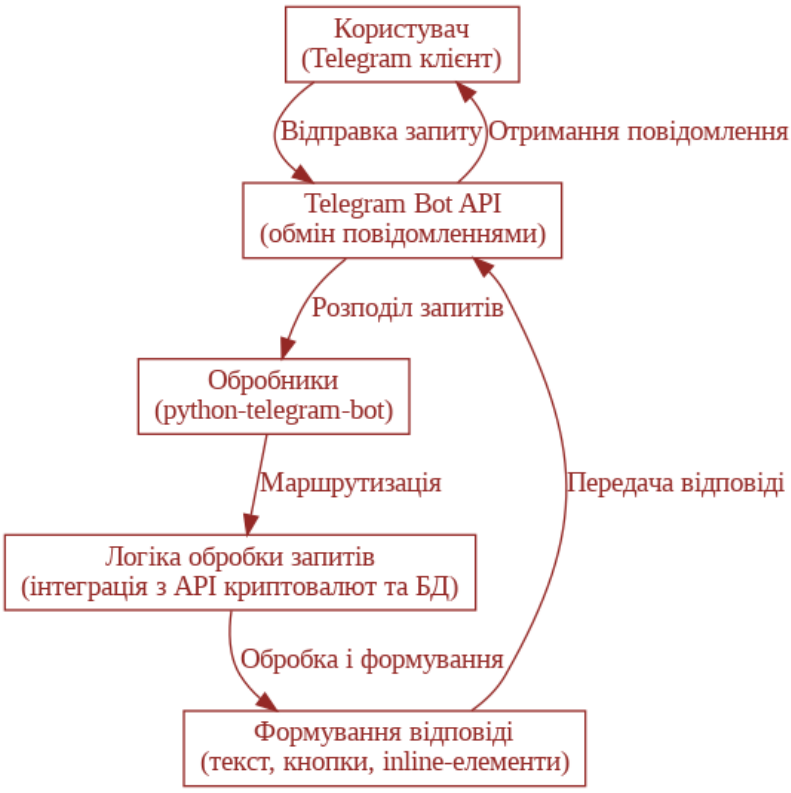


Рисунок 1.7. Схема взаємодії між основними компонентами інтерфейсу

Деталі реалізації інтерфейсу користувача є такими:

- Обробники (handlers): Завдяки використанню бібліотеки python-telegram-bot, кожен тип події (нове повідомлення, команда або callback-запит) обробляється окремим обробником. Це полегшує підтримання чистоти коду та розширює можливості модульного тестування;
- Інтеграція з API: На етапі логіки обробки запитів здійснюється звернення до зовнішніх API (CoinGecko для загальної статистики, Etherscan для даних Ethereum) та до локальної бази даних для завантаження попередніх налаштувань або збереженої історії. Результати обробки агрегуються і перетворюються у формат, придатний для відображення користувачу;
- Формування відповіді: Інтерактивні елементи Telegram (inline-клавіатури) використовуються для покращення взаємодії, що дозволяє користувачу швидко вибирати подальші дії без необхідності вводити додатковий текст вручну.

Вибір засобів для реалізації інтерфейсу користувача базується на простоті інтеграції та можливості масштабування для подальшої розробки. Використання Telegram Bot API у поєднанні з бібліотекою python-telegram-bot дозволяє забезпечити надзвичайно ефективну обробку запитів та відповіді, що є критично важливим для створення системи безпечного перегляду криптовалютної статистики. Підхід дозволяє зосередитися на ключовій логіці додатку, мінімізуючи витрати ресурсів та забезпечуючи стабільну роботу навіть на малопотужних пристроях.

1.3.3 Вибір необхідних засобів розробки та бібліотек

У цьому підрозділі окреслено обґрунтування вибору інструментарію для реалізації проекту, що включає як засоби для роботи з Telegram API, так і бібліотеки для інтеграції з блокчейном Ethereum, а також середовище розробки та операційну систему.

Для роботи з API месенджера Telegram у мові Python доступні кілька сторонніх бібліотек, серед яких найпопулярнішими є:

- pyTelegramBotAPI – забезпечує зручний синхронний стиль програмування,

використовуючи бібліотеку requests для HTTP-запитів.

- aiogram – асинхронний фреймворк, що базується на модулі aiohttp та asyncio. Завдяки асинхронній моделі він дозволяє розробити більш потужний та стійкий до навантажень інтерфейс, а також має вбудовані методи для взаємодії з базою даних.

Враховуючи високі вимоги до продуктивності та стійкості системи, для реалізації користувацького інтерфейсу проєкту обрано бібліотеку aiogram, яка дозволяє обробляти запити в режимі реального часу при великій кількості одночасних з'єднань.

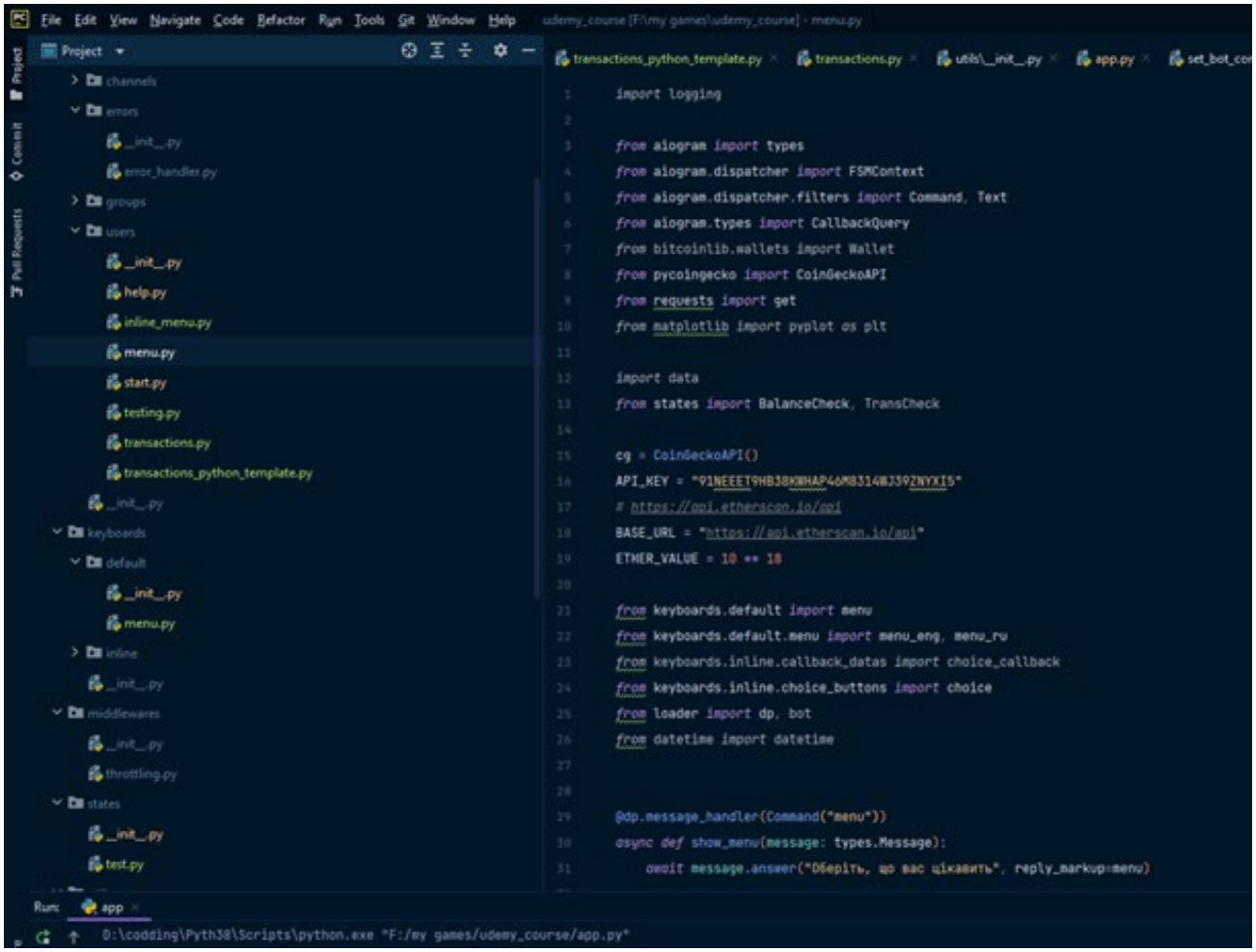


Рисунок 1.8. Розробка проекту у ICP PyCharm

Для розробки проєкту обрано сучасне програмне забезпечення та платформу, що забезпечують ефективність розробки та відлагодження:

- Операційна система: Windows – обрана через стабільність, широке коло підтримуваних програмних засобів та сумісність з сучасними технологіями

розробки;

- Середовище розробки: Для редагування та відлагодження коду обрано PyCharm від компанії JetBrains. Це професійне IDE підтримує роботу з декількома модулями одночасно; інтегроване управління базами даних; налагодження, запуск і тестування програм, а також взаємодію з командним рядком й інтерпретатором мови в одному вікні.

На рис. 1.8. показано інтегроване середовище PyCharm, яке дозволяє розробнику ефективно керувати всіма аспектами проєкту.

З огляду на завдання віртуального асистента, що включає отримання актуальних даних про стан цифрових гаманців, валідацію транзакцій та аналіз історії операцій, для взаємодії з мережею Ethereum обрано бібліотеку web3.py. Ця бібліотека дозволяє:

- Виконувати запити до мережі Ethereum на рівні API;
- Автоматизувати обробку даних і забезпечувати високий рівень безпеки транзакцій;
- Інтегруватись у задану архітектуру проєкту, з'єднуючи серверну частину із зовнішніми джерелами даних.

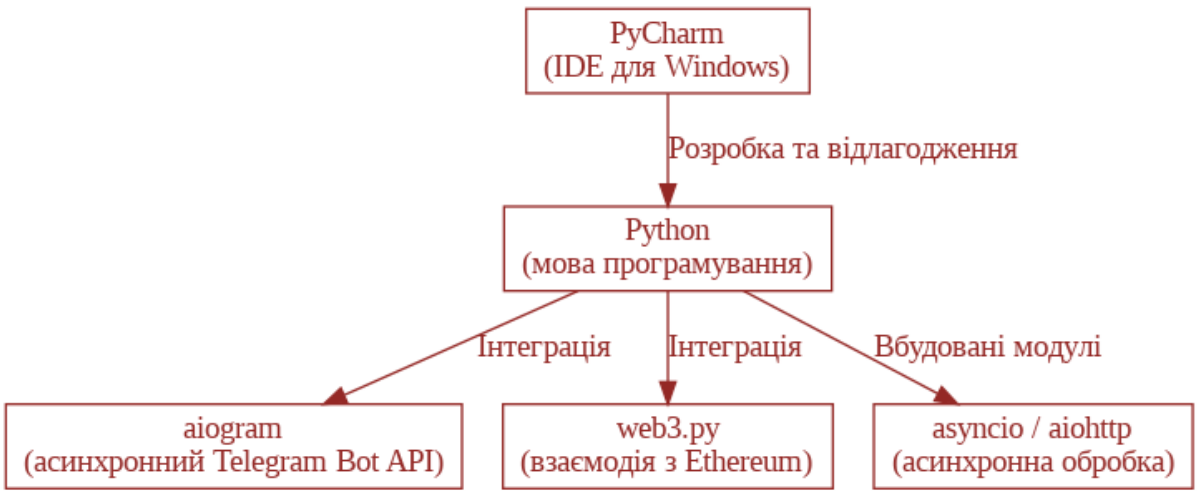


Рисунок 1.9. Діаграма основних засобів розробки та бібліотек проекту

Python обрано як основну мову програмування завдяки її виразності, широкій екосистемі бібліотек та активній спільноті розробників. Для забезпечення ефективної роботи проєкту використовуються:

- Модулі для роботи з REST API;

- Бібліотеки для асинхронного програмування (asyncio, aiohttp), що критично важливі для обробки численних запитів у реальному часі;
- Інтегровані криптографічні рішення, які дозволяють організувати високий рівень безпеки оброблюваних даних.

На рис. 1.9 наведено діаграму, що ілюструє взаємозв'язок основних засобів розробки та бібліотек проекту.

Вибір необхідних засобів розробки та бібліотек ґрунтується на вимогах щодо високої продуктивності, безпеки та стабільності системи. Використання aiogram для інтеграції з Telegram API, web3.py для взаємодії з мережею Ethereum, а також сучасного IDE PyCharm під Windows, у поєднанні з потужними можливостями Python щодо асинхронного програмування й роботи з REST API, дозволяє створити ефективний, масштабований та безпечний продукт для перегляду криптовалютної статистики та управління фінансовими операціями.

1.4 Розробка БСА обробки даних криптовалютної статистики

Блок-схема алгоритму (БСА) обробки даних є ключовою складовою архітектури системи, що реалізує функціонал віртуального асистента для безпечного перегляду криптовалютної статистики. У даному підрозділі розкривається загальна концепція розробки БСА, яка описує послідовність обробки запитів, починаючи з прийому повідомлень від користувача та закінчуючи формуванням відповіді, що містить актуальні дані про крипторинок.

Чат-бот, як система-комунікаційний агент, працює за принципом обробки вхідних текстових повідомлень, використовуючи технології NLP для аналізу та класифікації запитів (рис.1.10). Базова ідея полягає в тому, що обробка запиту ділиться на два основні етапи:

1. Прийом повідомлення. На цьому етапі система отримує текст від користувача через інтерфейс чат-бота. Використовуючи методи попередньої обробки, бот визначає ключові запити та інтенції користувача, що дозволяє правильно спрямувати подальшу обробку;

2. Обробка інформації та формування відповіді. На цьому етапі БСА здійснює звернення до зовнішніх інформаційних джерел (API криптовалютних

бірж, наприклад, CoinGecko або Etherscan) для отримання актуальних даних. Отримані дані агрегуються, аналізуються та форматуються у зрозумілий для користувача вигляд. Результат цього етапу – сформована відповідь, яка надходить назад через той же канал обміну повідомленнями.

Архітектурна модель БСА побудована за принципом модульного поділу завдань: кожен блок алгоритму відповідає за окрему функцію (розпізнавання інтенцій, запит до API, обробку даних, генерацію відповіді). Така структурованість дозволяє не лише спростити розробку й відлагодження, але й забезпечує можливість подальшої оптимізації та масштабування системи.

Рисунок 1.10. Загальна діаграма роботи асистента

1.4.1 Розробка БСА прийому нових блоків даних

У цьому пункті описується розробка блок-схеми алгоритму (БСА) для прийому блоків даних – ключового етапу, що забезпечує взаємодію користувача з мережею за умов мінімального навантаження на систему. Завдання цього

компоненту полягає в тому, щоб якомога ефективніше отримувати необхідну інформацію для подальшої обробки, використовуючи лінійну та ресурсоефективну послідовність операцій.

Основна ідея полягає в тому, що алгоритм має бути організований таким чином, щоб мінімізувати витрати обчислювальних ресурсів та забезпечити своєчасність надходження даних. Для цього було обрано максимально лінійну схему (рис.1.11):

1. Під'єднання до черги брокера. На цьому початковому етапі встановлюється зв'язок із системою обміну повідомленнями (чергою брокера), що дозволяє отримувати дані у вигляді блоків без необхідності виконання додаткових перевірок або складних обчислень;

2. Встановлення зв'язку з API. Налагоджується з'єднання із зовнішнім API, яке є джерелом актуальних даних. Цей етап передбачає перевірку доступності і стабільності з'єднання для гарантування отримання інформації за запитом;

3. Перевірка сигналу про завершення. Система перевіряє, чи досягнуто умови завершення отримання даних. Якщо сигнал дорівнює «Так», процес завершується, і надалі дані не приймаються. Якщо ж сигнал «Ні», алгоритм переходить до наступного етапу;

4. Отримання нового блоку. У разі відсутності сигналу завершення виконується прийом наступного блоку даних із джерела. Цей процес повторюється, поки не буде отримано необхідну інформацію або не спрацює умова завершення;

5. Формування повідомлення. Отримані дані конвертуються у структуроване повідомлення, яке містить ключову інформацію для подальшої обробки іншими компонентами системи;

6. Відправка повідомлення до черги. Сформоване повідомлення надсилається назад до системи через ту саму чергу брокера, що забезпечує подальшу маршрутизацію даних у систему обробки.

Таким чином, розроблена блок-схема алгоритму прийому блоків даних забезпечує лінійну та просту у реалізації послідовність операцій, яка мінімізує навантаження на систему та дозволяє заздалегідь отримати належну інформацію.

Рисунок 1.11. БСА прийому нових блоків даних

1.4.2 Розробка БСА обробки блоків інформації

Другий етап обробки даних, що реалізується у межах системи, є найбільш ресурсоємним і залежить від особливостей використовуваного API (CoinGecko або Etherscan). Цей етап полягає у запиті до зовнішнього джерела даних із заданими користувачем параметрами, після чого відбувається обробка отриманої інформації. Оскільки відповідь від сервісів може включати не лише основні дані, але й додаткову інформацію (наприклад, внутрішні транзакції), алгоритм повинен враховувати можливість проведення додаткових звернень до API.

Щоб мінімізувати кількість запитів до сервісу та зменшити навантаження на обчислювальну машину, у процес інтегровано штучну часову затримку. Використання асинхронного підходу дозволяє даній "затримці" бути локальною для оброблюваного блоку, що не впливає на загальну роботу програми. Таким

чином, підпрограма обробки отриманих блоків виконується поетапно:

1. Запит до API – система надсилає запит до зовнішнього сервісу (CoinGecko/Etherscan) для отримання даних за заданими параметрами;

2. Часова затримка – після отримання відповіді виконується затримка, яка дозволяє зменшити кількість звернень до API та оптимізувати використання ресурсів;

3. Запит внутрішніх транзакцій – за необхідності, за рахунок особливостей API, виконується додаткове звернення для отримання внутрішніх транзакцій, що може подовжувати час обробки;

4. Об’єднання даних – отримані дані з різних запитів, включаючи основні показники та внутрішні транзакції, об’єднуються в один список;

5. Перевірка відстежуваності адрес – отриманий список транзакцій аналізується: перевіряються всі адреси на предмет того, чи відстежуються вони хоча б одним користувачем системи;

6. Надсилання повідомлення – якщо адреса відповідає вимогам (відстежується користувачем), система надсилає сповіщення за допомогою чат-бота;

На рис.1.12 наведено приклад блок-схеми, що ілюструє алгоритмічну логіку обробки блоків інформації.

Використовуючи попередню перевірку на наявність грошових переказів, можна зменшити кількість запитів до API, що, у свою чергу, скорочує час обробки інформації. Після того, як усі транзакції об'єднані в єдиний список, здійснюється перевірка кожної адреси. Якщо адреса відстежується хоча б одним користувачем системи, автоматично генерується та надсилається повідомлення з результатами. Це дозволяє не лише оптимізувати операції за рахунок зменшення навантаження на систему, а й забезпечити своєчасне сповіщення користувачів про нові події в мережі криптовалюти.

1.4.3 Розробка БСА реєстрації адреси цільової мережі криптовалюти

Цей етап присвячено реалізації процесу реєстрації адреси цільової мережі криптовалюти через інтерфейс чат-бота у месенджері Telegram.

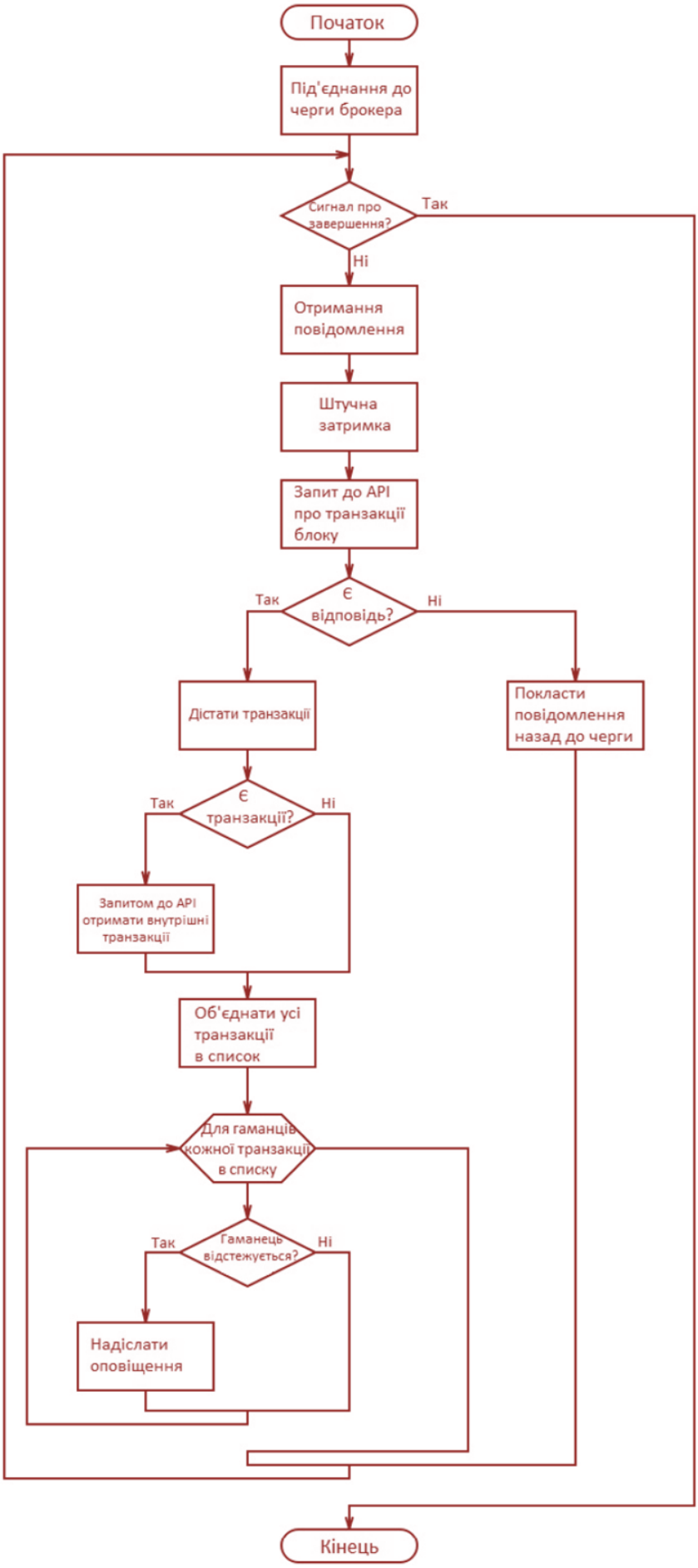


Рисунок 1.12. БСА обробки отриманої інформації

В основі цього процесу лежить чітко визначений сценарій, що представляє собою автомат із власними умовами та правилами переходу між станами, при цьому використовується внутрішня пам’ять для збереження проміжних даних.

Рисунок 1.13. БСА реєстрації адреси цільової мережі криптовалюти

Під час діалогу з користувачем бот послідовно виконує такі основні кроки:

1. Вітання та ініціація діалогу: Бот надсилає привітальне повідомлення та пропонує користувачу розпочати процес реєстрації, пояснюючи, що необхідно вказати адресу цільової мережі криптовалюти для подальшого відстеження;

2. Запит адреси: Бот запитує користувача ввести адресу цільової мережі криптовалюти. Реалізовано автоматичну перевірку коректності введеного формату. Якщо адреса некоректна, бот надсилає повідомлення про помилку з проханням повторити введення;

3. Запит символьного імені: Після успішної перевірки адреси, бот запитує користувача ввести символьне ім’я для відображення інформації або пропонує

автоматично згенерувати його, якщо користувач вирішить пропустити цей крок;

4. Збереження даних: Отримані дані (адреса та символьне ім’я) зберігаються у внутрішній пам’яті додатку для подальшого використання під час взаємодії з користувачем, а також для здійснення сповіщень;

5. Підтвердження реєстрації: На завершення процесу бот надсилає користувачу повідомлення з інформацією про успішну реєстрацію, вказуючи зареєстровану адресу та символьне ім’я.

На рис.1.13 наведено БСА, що ілюструє алгоритмічну логіку реєстрації адреси цільової мережі криптовалюти. Ця БСА відображає послідовний процес реєстрації адреси: починаючи з привітання користувача, через запит і перевірку адреси, до отримання необхідних даних та підтвердження реєстрації. Використання внутрішньої пам’яті для збереження проміжних результатів забезпечує цілісність процесу та дозволяє пізніше масштабувати логіку реєстрації.

1.5 Розробка модулів застосунку віртуального асистента

У даному розділі викладається розробка окремих модулів застосунку віртуального асистента, що забезпечують реалізацію основних функціональних можливостей системи. Архітектура застосунку побудована за принципом модульності, що дозволяє окремо розробляти, тестувати та масштабувати кожен компонент. Основні модулі включають:

- Модуль інтерфейсу з користувачем, який забезпечує діалогову взаємодію за допомогою чат-бота у месенджері Telegram. Цей модуль відповідає за прийом, обробку запитів і відправлення відповідей, а також за збереження проміжних і постійних даних;
- Модуль обробки бізнес-логіки, що займається інтеграцією з зовнішніми API (CoinGecko, Etherscan) та здійснює необхідну агрегацію і аналіз даних;
- Модуль управління даними, який реалізовано через сховище даних на основі Redis із структурою «ключ–значення». Завдяки використанню Redis дані зберігаються у вигляді JSON-рядків, при цьому важливо попередньо задавати типи даних для коректної конвертації.

Для проміжних даних (табл. 1.4) використовуються поля language, wallet,

coin, а для постійних даних – user_id, language (табл. 1.5). Постійні дані зберігаються тривалий час і використовуються для адресації повідомлень та налаштування постійного профілю користувача.

Таблиця 1.4. Проміжні дані (сесійні налаштування)

Назва	Тип	Призначення
language	str	Зміна мови інтерфейсу
wallet	str	Адреса гаманця
coin	str	Вибір криптовалюти

Таблиця 1.5. Основні дані користувача

Назва	Тип	Призначення
user_id	bigint	Унікальний ідентифікатор користувача, який присвоюється при реєстрації
language	varchar	Мова інтерфейсу користувача, що визначає подальшу мовну взаємодію

Розробка модулів застосунку віртуального асистента інтегрує всі ключові компоненти в єдину систему, що оптимізує взаємодію між кінцевим користувачем, зовнішніми сервісами та внутрішнім сховищем даних.

1.5.1 Модуль інтерфейсу з користувачем

Модуль інтерфейсу з користувачем є центральною складовою застосунку, що відповідає за організацію діалогу між користувачем та системою через Telegram-бота. Основні функціональні завдання цього модуля полягають у наступному:

1. Діалогова взаємодія: За допомогою Telegram Bot API модуль приймає вхідні повідомлення від користувача, обробляє їх і формує відповіді згідно з розробленим сценарієм. Логіка діалогу побудована як автомат із визначеними станами та переходами, що забезпечує послідовність дій під час реєстрації, отримання запитів та відправлення сповіщень;

2. Керування сесіями та збереження даних: Для ефективного управління проміжними даними використовується сховище на базі Redis. Тут зберігаються:

- Проміжні дані сесії, такі як вибір мови, адреса гаманця та вибір криптовалюти. Для цих даних використовуються невеликі об’єкти у форматі JSON (табл. 1.4). Дані зберігаються тимчасово та видаляються

після завершення сесії (наприклад, після виходу з меню перегляду історії транзакцій);

- Основні дані користувача, зокрема унікальний ідентифікатор (user_id) та налаштування мови, що зберігаються на тривалий період (табл. 1.5);

3. Адаптація інтерфейсу: Обрана мова інтерфейсу (українська, англійська або російська) впливає на подальшу мовну взаємодію з користувачем. Це забезпечує персоналізацію роботи застосунку: дані, що зчитуються з Redis, відображаються у відповідній мовній локалізації;

4. Інтеграція з бізнес-логікою: Модуль інтерфейсу забезпечує передачу повідомлень до центрального модуля обробки, який виконує запити до зовнішніх API (CoinGecko, Etherscan) та обробляє отримані дані, після чого формує відповідні відповіді, що повертаються користувачу.

На рис.1.14 подано ілюстративну схему модуля інтерфейсу з користувачем.

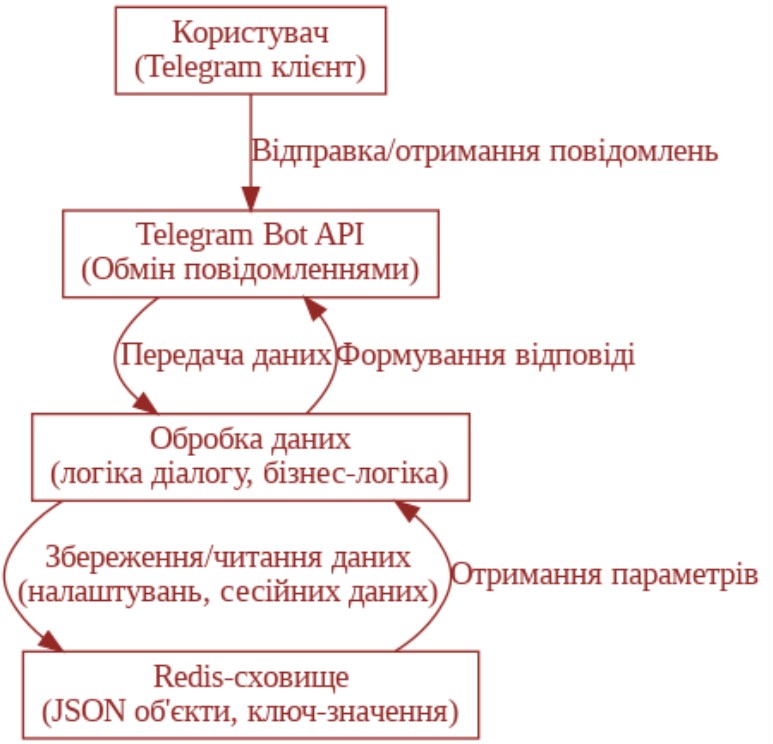


Рисунок 1.14. Схема модуля інтерфейсу з користувачем

Модуль інтерфейсу з користувачем забезпечує безперервний обмін інформацією між користувачем та системою, адаптує взаємодію відповідно до індивідуальних налаштувань (мова, адреса, криптовалюта) та інтегрує дані з Redis-сховища для оптимізації роботи застосунку. Цей підхід дозволяє зменшити

навантаження на систему завдяки використанню простого й ефективного ключ-значення сховища, при цьому забезпечуючи високу швидкість обробки та персоналізовану взаємодію.

1.5.2 Модуль обробки бізнес-логіки

Модуль обробки бізнес-логіки є центральною частиною застосунку віртуального асистента, який відповідає за отримання, агрегацію та аналіз даних, що надходять від зовнішніх API (CoinGecko, Etherscan), а також за їх подальше представлення кінцевому користувачу. Цей модуль реалізовано через низку незалежних програмних компонентів, що об’єднуються в єдину систему, яка працює в асинхронному режимі з використанням Python (asyncio, aiohttp) та Redis для кешування.

Основні функції модуля обробки бізнес-логіки включають:

1. Обробка запитів: На початковому етапі модуль приймає інформацію, отриману з модуля інтерфейсу користувача, і проводить її валідацію. Це включає перевірку коректності введених даних (наприклад, формат адреси цільової мережі, вибір криптовалюти тощо) та визначення характеру запиту

2. Інтеграція з зовнішніми API: Після валідації запиту модуль надсилає запити до зовнішніх сервісів CoinGecko (для отримання загальної статистики криптовалют) та Etherscan (для отримання даних про транзакції та стан гаманця). Для цього використовується асинхронне програмування, що дозволяє знизити затримки та оптимізувати ресурси обчислювальної машини;

3. Агрегація та аналіз отриманих даних: Після отримання відповідей від зовнішніх API модуль виконує агрегацію даних із різних джерел. Цей етап включає об’єднання інформації про транзакції, аналіз часу оновлення даних, розрахунок статистичних показників, а також перевірку внутрішніх транзакцій (при їх наявності). В разі, якщо блок не містить звичайних грошових переказів, запит на внутрішні транзакції може бути пропущено для зниження витрат часу;

4. Оптимізація за допомогою кешування: Для зменшення кількості повторних звернень до зовнішніх API та зниження обчислювальних витрат модуль застосовує механізми кешування за допомогою Redis. Отримані дані зберігаються

на певний час, що дозволяє швидше формувати відповіді при повторних запитах і запобігати блокуванню через надмірне навантаження;

5. Формування відповіді: На фінальному етапі модуль формує структуровану відповідь для модуля інтерфейсу користувача. Відповідь може включати текстові повідомлення, інтерактивні елементи (наприклад, кнопки) або дані для побудови графіків і таблиць. Це забезпечує зручне і зрозуміле представлення отриманої інформації для кінцевого користувача.

На рис.1.15 наведено схему модуля обробки бізнес-логіки, яка ілюструє основні етапи його роботи.

Рисунок 1.15. Схема модуля обробки бізнес-логіки

Модуль обробки бізнес-логіки поділено на такі програмні компоненти:

- API Client. Файл: api_handler.py. Призначення: Реалізація асинхронних

запитів до зовнішніх API (CoinGecko для статистики криптовалют, Etherscan для даних блокчейну). Компонент використовує бібліотеки aiohttp та asyncio для мінімізації затримок. Основні функції: fetch_market_data(), fetch_internal_transactions();

- Data Validator. Файл: data_validator.py. Призначення: Валідація вхідних запитів від модуля інтерфейсу. Перевірка формату адрес, відповідності вибраної криптовалюти, коректності інших параметрів. Основні функції: validate_request(), sanitize_input();
- Cache Manager. Файл: cache_manager.py. Призначення: Зберігання отриманих даних у Redis для зниження кількості повторних запитів до зовнішніх сервісів, оптимізації використання ресурсів і пришвидшення відповіді системи. Основні функції: cache_data(key, value), retrieve_data(key);
- Business Logic Core. Файл: business_logic.py. Призначення: Агрегація та аналіз отриманих даних, обчислення статистичних параметрів (середнє значення, коливання, інтервали оновлення) та формування готової відповіді для модуля інтерфейсу користувача. Основні функції: aggregate_data(), form_response(), process_request(request: dict).

Послідовність роботи модулю:

1. Прийом та валідація запиту: Модуль отримує запит від інтерфейсу (вже попередньо оброблений). Використовується функція validate_request() з модуля data_validator.py для перевірки коректності переданих даних:

```
from data_validator import validate_request
valid, error = validate_request(request)
if not valid:
return {"error": error}
```

2. Запит даних до зовнішніх API: За допомогою fetch_market_data() з модуля api_handler.py надсилаються асинхронні запити до CoinGecko та Etherscan для отримання даних про вибрану криптовалюту та транзакції:

```
from api_handler import fetch_market_data
market_data = await fetch_market_data(request['coin'])
```

3. Кешування результатів: Для оптимізації роботи використовується Redis. Отримані дані зберігаються за ключем, що складається із ідентифікатора криптовалюти, за допомогою функції cache_data() з cache_manager.py:

from cache_manager import cache_data

await cache_data(request['coin'], market_data)

4. Агрегація та аналіз даних: Модуль Business Logic Core обробляє отриману інформацію, об'єднує дані з декількох запитів (включаючи внутрішні транзакції, якщо такі наявні) та проводить розрахунки статистичних показників через функцію aggregate_data():

from business_logic import aggregate_data

aggregated_info = aggregate_data(market_data)

5. Формування відповіді: З отриманих агрегованих даних формується структурована відповідь у форматі JSON за допомогою form_response();

from business_logic import form_response

response = form_response(aggregated_info)

return response

Таблиця 1.6. Основні програмні модулі бізнес-логіки

Назва модуля	**Файл**	**Основні функції**	**Використані бібліотеки**
API Client	api_handler.py	fetch_market_data(), fetch_internal_transactions()	aiohttp, asyncio
Data Validator	data_validator.py	validate_request(), sanitize_input()	Standard Python libs
Cache Manager	cache_manager.py	cache_data(key, value), retrieve_data(key)	aioredis
Business Logic Core	business_logic.py	aggregate_data(), form_response(), process_request()	Standard Python libs

Для запуску модуля обробки бізнес-логіки в режимі тестування використовується командний рядок. Для запуску головного файлу бізнес-логіки:

python business_logic.py --request '{"user_id": 123456789, "coin": "bitcoin", "language": "en"}'

Для тестування асинхронних функцій застосовується фреймворк pytest-

asyncio:

pytest --maxfail=1 --disable-warnings -q

Завдяки чіткому розподілу функціональності між окремими програмними модулями, модуль обробки бізнес-логіки забезпечує ефективну асинхронну взаємодію із зовнішніми API, оптимізацію використання ресурсів за допомогою кешування, а також точну агрегацію та аналіз отриманих даних. Така структура дозволяє значно знизити затримки обробки запитів та підвищити продуктивність віртуального асистента, що критично важливо для роботи з динамічним криптовалютним ринком.

1.5.3 Модуль управління даними

Основною складовою застосунку віртуального асистента є модуль управління даними, який відповідає за довгострокове зберігання та обробку інформації. В основному сховищі зберігаються дані, що підлягають тривалому утриманню, і кількість таких даних може бути досить великою. Тому база даних має бути легкодоступною для розширення та гнучкою щодо внесення змін чи корекцій прямо під час роботи програми.

Для забезпечення стабільної роботи застосунку реалізовано модуль управління даними, що складається з декількох таблиць. Основне сховище організоване таким чином, що дозволяє зберігати всю критичну інформацію у структурованому вигляді і підтримує моментальне відновлення даних у разі, наприклад, очищення оперативної пам’яті або перезавантаження серверу. В даному проєкті база даних містить чотири ключові таблиці (табл.1.7-1.10).

Таблиця 1.7. user

Назва поля	**Тип**	**Призначення**
user_id	bigint(20)	Унікальний ідентифікатор користувача (присвоюється месенджером при реєстрації); первинний ключ
reg_date	timestamp	Дата та час реєстрації користувача
lang	varchar(20)	Мова інтерфейсу користувача (дублікат налаштувань, що зберігається для відновлення після втрати)

Таблиця 1.8. wallet

Назва поля	Тип	Призначення
wallet_id	bigint(20)	Унікальний ідентифікатор адреси (автоматично встановлюється при додаванні запису; первинний ключ)
user_id	bigint(20)	Унікальний ідентифікатор користувача (посилається на первинний ключ таблиці user); разом із текстом адреси утворює унікальний ключ
wallet	varchar(100)	Текстове представлення адреси
name	varchar(30)	Символьне ім’я адреси
reg_date	timestamp	Дата реєстрації адреси

При спробі додати наявну адресу до таблиці, замість створення нового запису відбувається оновлення імені та дати реєстрації.

Таблиця 1.9. history_filter

Назва поля	Тип	Призначення
filter_id	bigint(20)	Унікальний ідентифікатор фільтру (встановлюється автоматично; первинний ключ)
wallet_id	bigint(20)	Унікальний ідентифікатор адреси (посилається на первинний ключ таблиці wallet; утворює з ним унікальний ключ, що запобігає дублюванню фільтрів для однієї адреси)
records_number	tinyint(3)	Кількість записів історії транзакцій для відображення (допустимий діапазон – 1 до 10, за замовчуванням – 5)
show_participant	tinyint(1)	Булевий прапорець, що визначає, чи потрібно відображати від кого або кому здійснюється переказ (за замовчуванням – відображати)

Таблиця user містить дані всіх користувачів застосунку та деякі відомості про них. Таблиця wallet зберігає інформацію про адреси криптовалют, які зареєстровані користувачами. Таблиця history_filter зберігає користувацькі налаштування відображення записів історії транзакцій для певної адреси криптовалютної мережі. Таблиця notifications_filter призначена для зберігання налаштувань оповіщення користувача про нові транзакції за участю відстежуваної адреси. При створенні запису в таблиці wallet відразу генерується відповідний запис у таблиці notifications_filter, що дозволяє користувачу налаштовувати

відправку повідомлень за заданими параметрами. Модуль управління даними реалізовано з урахуванням високої кількості збережуваної інформації та необхідності швидкого масштабування бази даних. Гнучкість сховища забезпечується можливістю внесення змін без переривання роботи застосунку. Наприклад, при зміні вимог до відображення історії транзакцій або налаштування оповіщень базу даних можна доповнювати новими таблицями або додатковими полями, не порушуючи існуючу логіку обробки даних.

Таблиця 1.10. notifications_filter

Назва поля	**Тип**	**Призначення**
filter_id	bigint(20)	Унікальний ідентифікатор фільтру (автоматично встановлюється; первинний ключ)
wallet_id	bigint(20)	Унікальний ідентифікатор адреси (посилається на таблицю wallet; забезпечує унікальність запису для кожної адреси)
min_amount	decimal(20,10)	Мінімальна сума переказу, за якої надсилаються оповіщення (діапазон: від 0.0000000001 до 9999999999.9999999999, за замовчуванням – 0)
tr_type	enum('in','out','inout')	Напрямок транзакції для оповіщення (опція вибору: усі перекази, тільки надходження або лише списання; за замовчуванням – всі транзакції)
off_sound	tinyint(1)	Прапорець, що визначає стан звукового оповіщення ("увімкнути" або "вимкнути")
notify	tinyint(1)	Прапорець увімкнення функції оповіщення, який дозволяє вимкнути оповіщення без видалення адреси (за замовчуванням – включено)

Таким чином, модуль управління даними виступає як центральне сховище тривалих даних користувачів та їх налаштувань, забезпечуючи як оперативний доступ до інформації, так і стабільну роботу системи в умовах постійного оновлення даних із зовнішніх джерел. Це є критично важливим для ефективної взаємодії віртуального асистента з динамічним криптовалютним ринком.

На рис. 1.16 показано схему бази даних, яка використовується в застосунку асистента для зберігання як основних, так і проміжних даних користувачів.

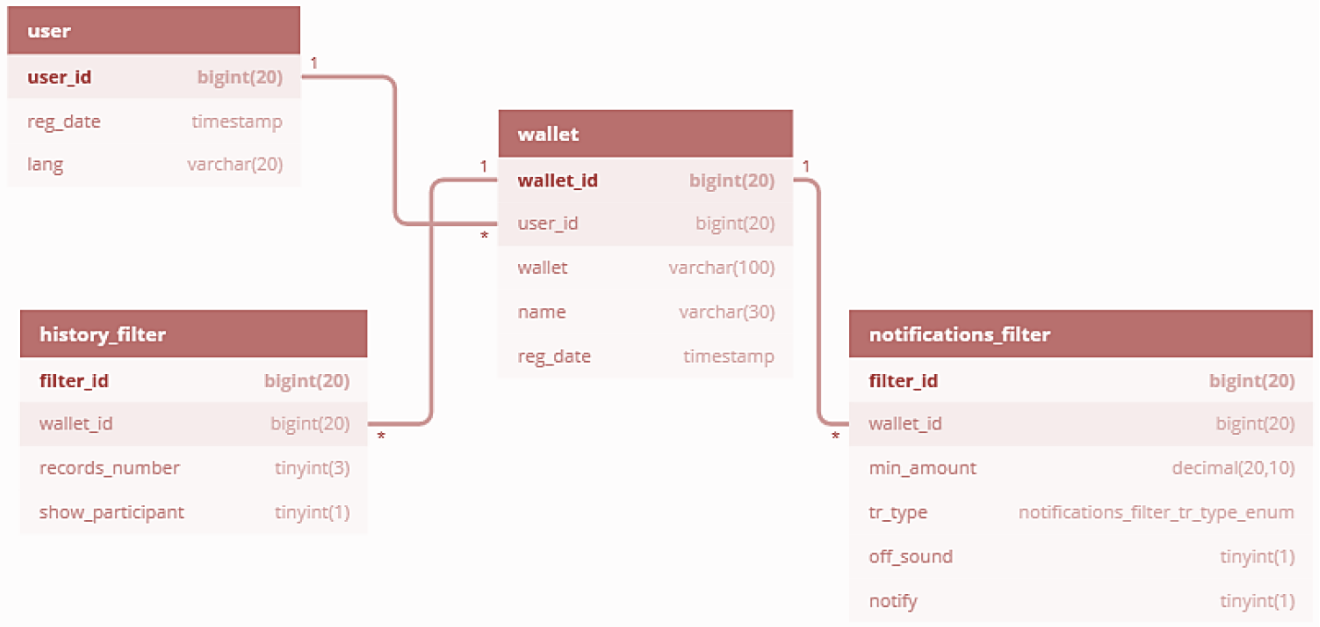


Рисунок 1.16. Схема бази даних управління основними даними застосунку

На схемі рис.1.16 позначено наступні таблиці:

- user: містить основні дані користувача, зокрема унікальний ідентифікатор (user_id), дату реєстрації (reg_date) та мову інтерфейсу (lang). Ця таблиця використовується для адресації повідомлень у месенджері та зберігає дані, які підлягають тривалому зберіганню;
- wallet: зберігає інформацію про адреси гаманців, що зареєстровані користувачами, разом із символьними іменами та датами реєстрації. Таблиця має зв'язок з таблицею user, завдяки якому визначається, який користувач володіє певною адресою;
- history_filter: містить налаштування відображення історії транзакцій для кожної адреси. За допомогою цієї таблиці можна задавати параметри фільтрації (наприклад, кількість записів для відображення та прапорець для показу інформації про інших учасників транзакції);
- notifications_filter: зберігає дані, пов'язані з налаштуванням сповіщень про нові транзакції. Сюди входять мінімальна сума для оповіщення, напрямок транзакції, а також прапорці для звукового та штатного режиму оповіщень.

Зв'язки між таблицями (через первинні та зовнішні ключі) забезпечують цілісність даних і дозволяють легко здійснювати запити для отримання інформації як по окремих користувачах, так і по відповідних налаштуваннях відображення

історії транзакцій і оповіщень.

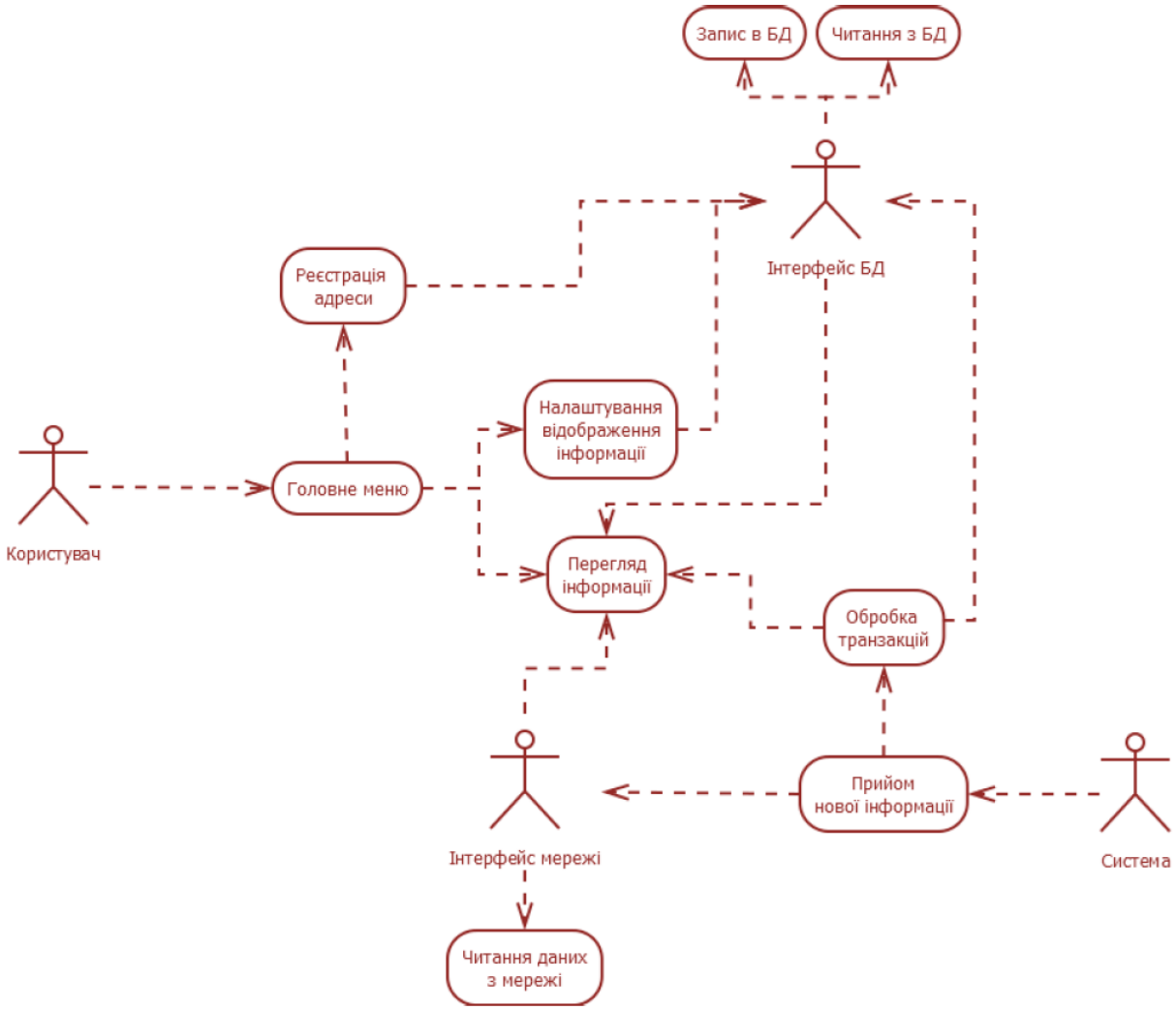


Рисунок 1.17. Схема управління даними застосунку віртуального асистента

На рис.1.17 показано блок-схему, яка ілюструє послідовний процес взаємодії користувача із системою. На схемі відображено етапи роботи застосунку, такі як:

- Реєстрація та ініціація діалогу: Користувач починає взаємодію, переходить до головного меню, де запускається сценарій реєстрації адреси цільової мережі криптовалюти;
- Налаштування відображення інформації: Після введення адреси користувача запитують додаткові параметри, наприклад, символьне ім’я, що використовується для подальшого представлення інформації;
- Обробка запитів та транзакцій: Схема охоплює етапи передачі даних між користувачем, інтерфейсом бота, базою даних (запис та читання) та зовнішнім мережевим інтерфейсом;
- Інтеграція з базою даних: Вказані етапи взаємодії з БД (читання та запис)

дозволяють зберігати як постійні, так і проміжні дані, що забезпечує безперервність роботи застосунку.

1.5.4 Реалізація структури проекту застосунку віртуального асистента

У цьому підрозділі описується організація файлової структури проекту застосунку віртуального асистента, який розроблено на базі Telegram Bot API. Структурна організація проекту має критичне значення для підтримки модульності, розширюваності та зручності у подальшій підтримці системи. Зокрема, з урахуванням способу розподілу відповідальності між виконуваними файлами та частинами користувацького інтерфейсу, структура проекту розбита на логічно відокремлені компоненти.

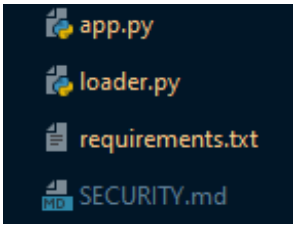


Рисунок 1.16. Структура проекту: основні файли

На рис. 1.16 показано основні виконувані файли, які лежать в основі застосунку:

- app.py – головний файл, що відповідає за запуск Telegram-бота;
- loader.py – файл, який містить основні параметри для роботи бота (наприклад, токен авторизації);
- requirements.txt – файл, що містить перелік залежностей і вимог до програмного забезпечення, необхідних для коректної роботи проекту.

Ці файли забезпечують базову функціональність застосунку, гарантуючи, що всі основні сервіси будуть завантажені і налаштовані перед початком роботи.

На рис. 1.17 представлено детальну організацію директорій, що відповідають за логіку взаємодії з користувачем. Ця структура поділяється на три основні частини:

1. handlers. Директорія handlers та її дочірні піддиректорії містять файли, які обробляють запити користувача. Саме тут реалізовано основну логіку реагування на текстові повідомлення та натискання кнопок. Файли з цієї директорії можуть

імпортувати допоміжний функціонал з інших частин проекту для побудови комплексної відповіді;

2. markups. Директорія markups містить функції та шаблони для формування користувацьких меню, представлених у вигляді кнопок. Завдяки цьому забезпечується інтуїтивно зрозуміла навігація користувача по застосунку;

3. states. Файли у директорії states містять перелік можливих станів, в яких може знаходитися користувач при взаємодії з ботом. Це дозволяє реалізувати керування сесіями та контроль за послідовністю сценаріїв взаємодії.

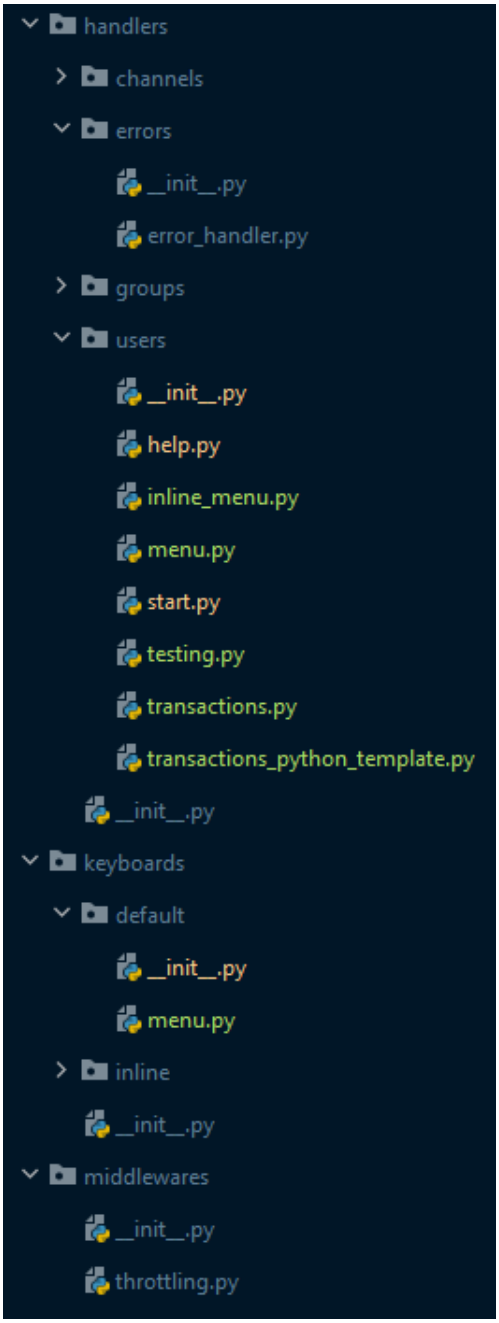


Рисунок 1.17. Структура проекту: файли користувацького інтерфейсу

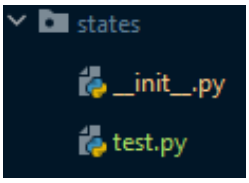


Рисунок 1.18. Структура проекту: файли стану

На рис. 1.18 представлено демонструє приклади файлів, таких як __init__.py та test.py, які визначають ієрархію станів, необхідних для роботи алгоритмів.

Реалізація структури проекту застосунку віртуального асистента чітко розділяє базові файли (app.py, loader.py, requirements.txt) від компонентів, відповідальних за формування користувацького інтерфейсу (директорії handlers, markups, states). Така організація забезпечує високий рівень модульності, спрощує розширення та підтримку застосунку та дозволяє швидко інтегрувати нові функціональні можливості.

1.6 Тестування роботи віртуального асистента для безпечного перегляду криптовалютної статистики

Тестування роботи віртуального асистента проводилося з метою перевірки як функціональних аспектів, так і забезпечення високого рівня безпеки під час обробки та передачі даних. Особливу увагу приділено тому, щоб усі запити користувача, взаємодія з зовнішніми сервісами та внутрішнє зберігання інформації відбувалися у зашифрованому вигляді, а доступ до конфіденційних даних мав лише авторизований користувач. Крім того, тестування включало перевірку цілісності даних, що отримуються з API, та відповідність інформації, отриманої через бот, інформації на офіційних ресурсах Etherscan та CoinGecko.

Основні етапи тестування віртуального асистента для безпечного перегляду криптовалютної статистики:

1. Запуск бота та вітальна взаємодія: При надсиланні стандартної команди /start, що є загальновживаною для всіх віртуальних асистентів у Telegram, бот надсилає вітальне повідомлення. Це підтверджує успішну ініціалізацію та встановлення зашифрованого каналу зв’язку між користувачем і сервером. На рис. 1.19 показано вітальне повідомлення, яке містить інформацію про основні функції бота;

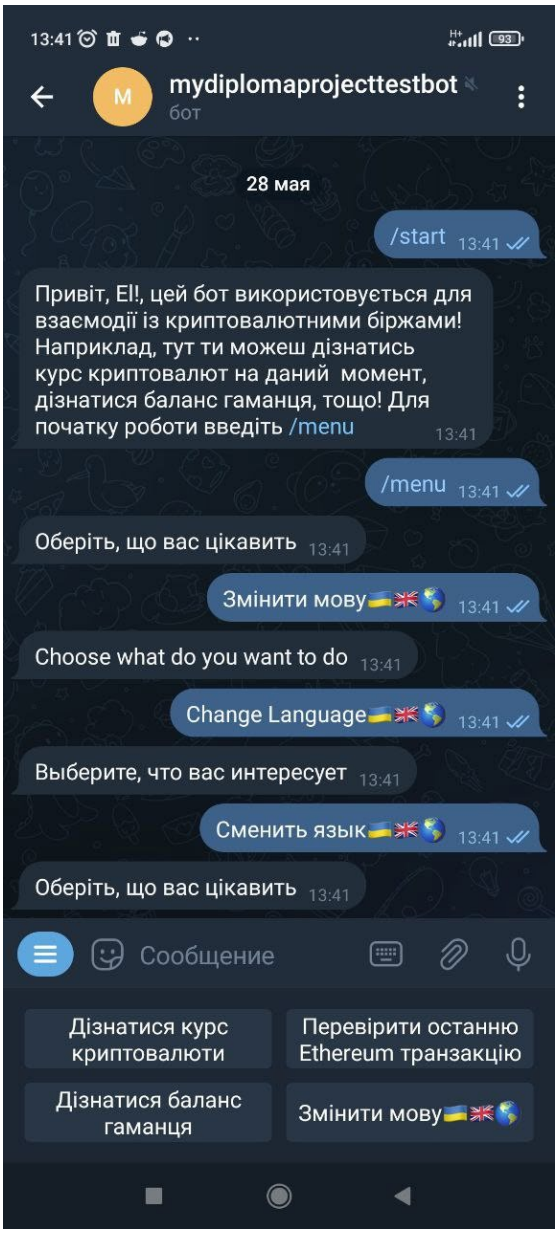


Рисунок 1.19. Привітальне повідомлення та зміна мови
у головному меню віртуального асистента

2. Перехід до головного меню: Після отримання команди /start користувач натискає на кнопку підсвіченої команди /menu, що відкриває головне меню застосунку. Меню містить основні функції, включаючи реєстрацію адреси, перегляд балансу гаманців, перевірку транзакцій, зміну мови та інші дії. На рис. 1.19 показано інтерфейс головного меню, з якого користувач може обрати потрібну функцію;

3. Зміна мови інтерфейсу: Натискання кнопки зміни мови призводить до відправлення нового меню на наступній запропонованій мові, після чого бот автоматично змінює мову реплік і взаємодії.

Rank	Address	Name Tag	Balance	Percentage	Txn Count
1	0x00000000219ab540356cbb839cbe05303d7705fa	Eth2 Deposit Contract	12,658,050.000069 Ether	10.62733945%	197,984
2	0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2	Wrapped Ether	5,817,396.53299427 Ether	4.88412099%	7,945,310
3	0xda9dfa130df4de4673b89022ee50ff26f6ea73cf	Kraken 13	2,113,030.00443456 Ether	1.77404001%	66
4	0xbe0eb53f46cd790cd13851d5eff43d12404d33e8	Binance 7	1,996,008.28210113 Ether	1.67579189%	1,092
5	0x73bceb1cd57c711feac4224d062b0f6ff338501e		1,900,505.52801049 Ether	1.59561049%	488
6	0x9bf4001d307dfd62b26a2f1307ee0c0307632d59		1,490,000.0180927 Ether	1.25096172%	103
7	0x4ddc2d193948926d02f9b1fe9e1daa0718270ed5	Compound: cETH Token	793,828.54129808 Ether	0.66647591%	277,097

Рисунок 1.20. Вибір гаманців Ethereum

Balance: 2,113,030.0044345678 Ether

Рисунок 1.21. Перегляд балансу гаманця (дані з Etherscan)

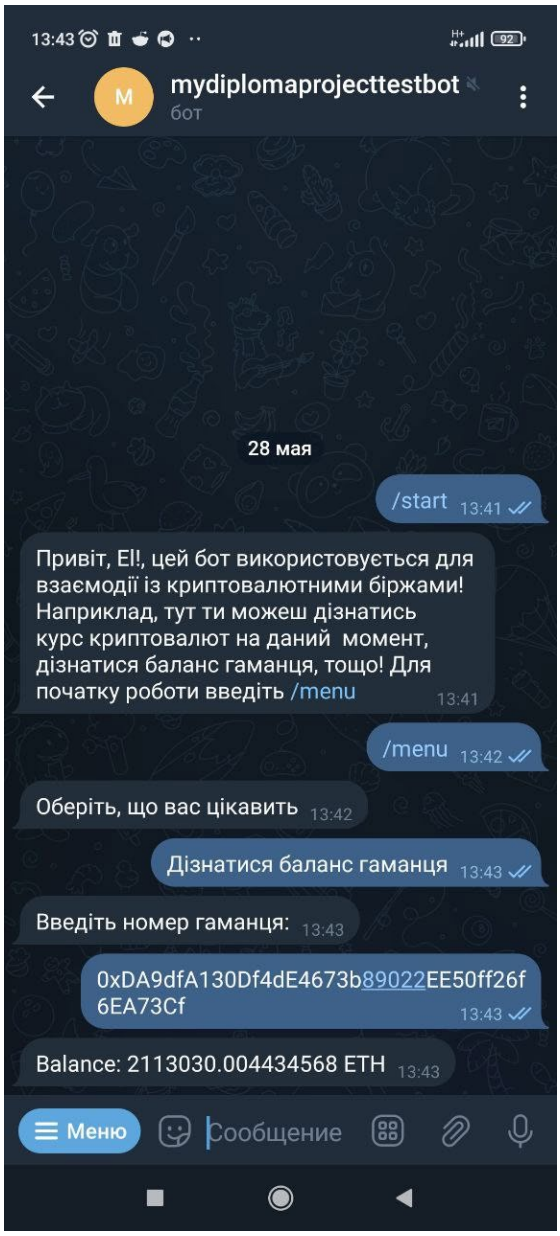


Рисунок 1.22. Баланс гаманця, відображений у інтерфейсі асистенту

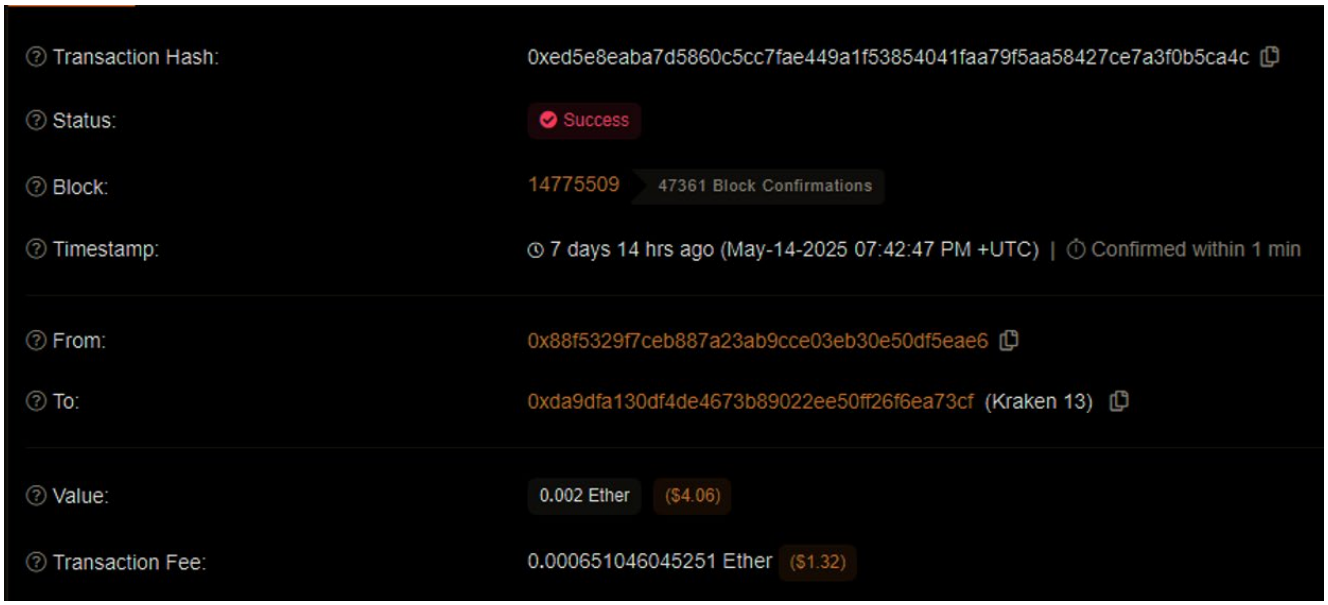


Рисунок 1.23. Деталі транзакції за даними Etherscan

Цей механізм забезпечує адаптацію системи до вимог користувача, при цьому підтримуючи безпечну передачу налаштувань. На рис. 1.19 показано, як меню оновлюється після зміни мови;

4. Перевірка балансу гаманця: Для перегляду балансу користувач натискає кнопку «Дізнатися баланс гаманця». Бот запитує унікальний номер гаманця, після чого відправляє запит до API Etherscan для отримання даних про баланс. Отримані дані аналізуються й форматуються у зручному вигляді перед відправкою користувачу. При тестуванні використовувалася відкрита база даних Etherscan для перевірки даних кращих гаманців мережі Ethereum. На рис. 1.20 показано перелік адрес, з якого користувач обирає потрібний гаманець. На рис. 1.21 показано отриманий баланс гаманця у форматі, оптимізованому для зручного перегляду. На рис. 1.22 показано, що значення балансу, надане ботом, збігається із даними з Etherscan, за винятком незначної округленості, що зумовлена застосуванням форматування для безпеки та зручності;

5. Перевірка транзакцій: Для перевірки останньої транзакції користувача обирається функція «Перевірити останню Ethereum транзакцію». Бот запитує унікальний номер гаманця, надсилає запит до Etherscan і отримує дані про останню транзакцію. На рис. 1.23 показані дані, отримані напряму від Etherscan щодо конкретної транзакції. На рис. 1.24 показана відповідь від бота. Отримані дані

повністю відповідають інформації з Etherscan за винятком невеликої різниці у часових позначеннях (через використання різних часових поясів), що не впливає на безпеку обробки даних;

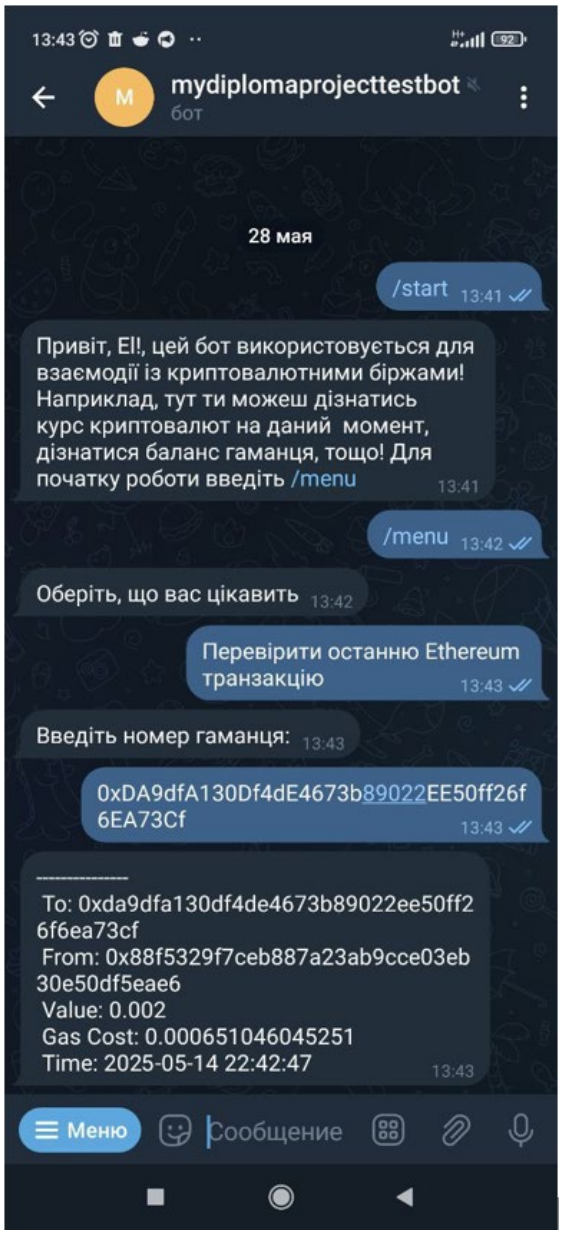


Рисунок 1.24. Перевірка транзакції у інтерфейсі асистенту

6. Перевірка курсу криптовалют: Для отримання актуальної ціни на найпопулярніші криптовалюти користувач натискає кнопку «Дізнатися курс криптовалюти». З’являється inline меню із переліком доступних криптовалют. Після вибору визначається поточна ціна в доларах, яка надійно отримується через CoinGecko API.

На рис. 1.25 показано процес перевірки роботи курсу криптовалют.

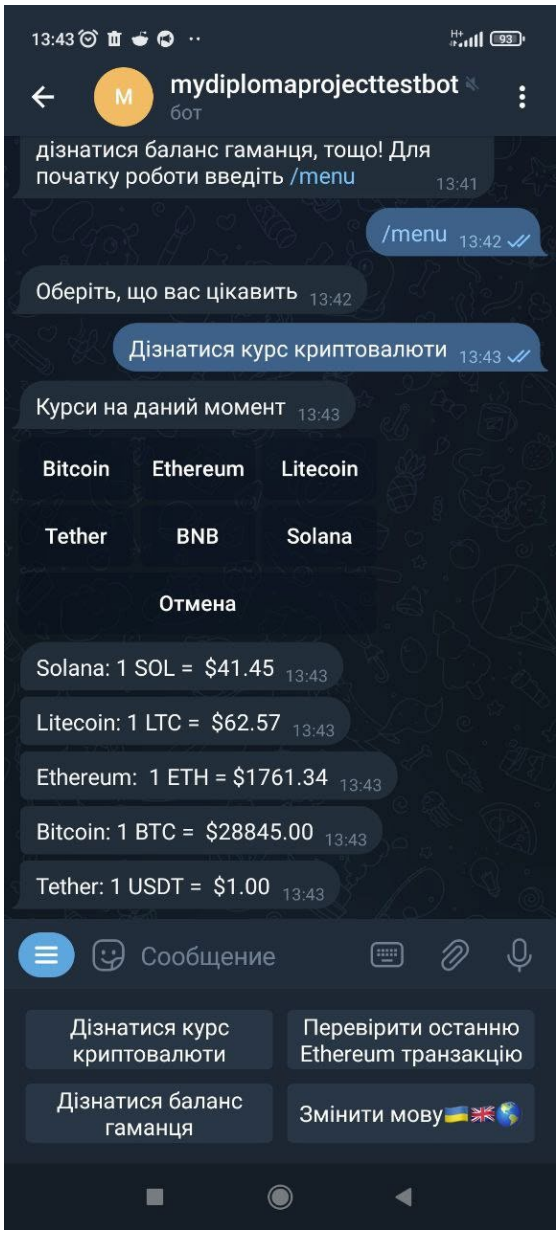


Рисунок 1.25. Запит перевірки курсу криптовалют у інтерфейсі асистенту

Під час тестування основну увагу приділено захисту даних та забезпеченню конфіденційності інформації:

- Кожне з'єднання з зовнішніми API здійснюється через зашифрований протокол HTTPS, що запобігає перехопленню даних у мережі;
- Етап отримання запитів від користувачів на сервері обробляється із застосуванням механізмів автентифікації та авторизації, що гарантує, що лише легітимні запити обробляються ботом;
- Операції з Redis (як тимчасовими, так і постійними даними) виконуються через захищені канали, що мінімізує ризик витоку конфіденційної інформації;

- Асинхронна обробка запитів забезпечує швидке реагування системи навіть під високим навантаженням, не впливаючи на рівень безпеки взаємодії.

Тестування роботи віртуального асистента продемонструвало коректність та безпечність роботи усіх функціональних компонентів застосунку: від початкового вітання та відображення інтерфейсу до обробки запитів на перевірку балансу, транзакцій та курсу криптовалют. Результати тестування підтверджують, що бот надійно взаємодіє із зовнішніми сервісами, зберігає дані у захищеному сховищі, а також формує відповіді у зручному форматі для користувачів. Таким чином, віртуальний асистент забезпечує безпечний перегляд криптовалютної статистики, що є критично важливим для сучасних фінансових застосунків.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Даний дипломний проект присвячений розробці віртуального асистента, який у поєднанні з технологіями чат-ботів дозволить користувачам безпечно та ефективно отримувати доступ до даних про криптовалюти. За допомогою запропонованого рішення створюється можливість моніторингу актуальних статистичних даних, що є важливою складовою сучасних фінансових сервісів, забезпечуючи при цьому високий рівень захисту інформації та користувацьких даних.

Ефективність програмного забезпечення залежить не лише від його якості, а й від результативності процесу його створення. Якість ПЗ оцінюється за кількома критеріями: з погляду користувача, раціонального використання ресурсів і відповідності заданим вимогам. З позиції користувача також враховуються витрати на розробку, зокрема трудові та фінансові. У цьому розділі представлено розрахунок вартості створеного програмного продукту.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість створення програмного продукту визначається його масштабом, рівнем трудомісткості, кваліфікацією виконавців та встановленими ринковими термінами. Використовуючи метод структурної аналогії та аналіз відповідних каталогів аналогічного ПЗ, можна встановити обсяг програмного засобу, що виражається у тисячах умовних машинних команд для аналога

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах.**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Норма часу

Обсяг ПЗ, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається окремо для кожного етапу розробки, виходячи з показників трудомісткості відповідного аналога. При цьому враховують рівень складності розробки, ступінь інноваційності та частку використання стандартних модулів. Розрахунки проводяться згідно з наступними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз}=T^a *L_1*К_н= 183,2 *0,12*0,7= 15,38 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$T_{тп}=T^a *L_2*К_н =183,2 *0,11*0,7 = 14,11 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$T_{рп}=T^a *L_3*К_н*К_т=183,2 *0,61*0,7*0,6= 46,94 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=23 (стр), розробка робочого проекту $N_{рп}$=6(стр), пояснювальна записка відповідно $N_{пз}$=25(стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}=0,15*2=0,3$
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}=0,7*N_{тп}=0,7*23=16,1$	$T_{нк}=0,15*N_{тп}=0,15*23=3,45$
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}=0,7*N_{рп}=0,7*6=4,2$	$T_{нк}=0,15*N_{рп}=0,15*6=0,9$
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$= 1,5*25 =37,5	$T_{кк}=0,7*N_{тз}=0,7*25=17,5$	$T_{нк}=0,15*N_{пз}=0,15*25$ =3,75
Усього, в т.ч.:	161,53		
- на розробку	$\Sigma T_р$=113,93		
- контроль керівника		$\Sigma T_{кк}$=39,2	
- нормоконтроль			$\Sigma T_{нк}$=8,4

2.3 Розрахунок ціни програмного продукту

Для оцінки вартості програмного продукту розглядаються основна заробітна плата виконавців, матеріальні витрати та загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати наведено у таблиці 2.7. Згідно зі статтею 8 «Закону про Державний бюджет України на 2025» з 1 січня 2025 року встановлено мінімальну місячну заробітну плату у розмірі 8000 гривень, а також мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	113,93	48,00	5468,64
2.Контроль керівника	39,2	115,00	4508,00
3.Нормоконтроль	8,4	110,00	924,00
Усього	-	-	ΣЗо= 10900,64

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	56	5.0	280,00
Разом	-	-	-	$В_{м\,і}$=280,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$ = 0,1*280=28,0
Усього				$В_м$=$В_{мі}$+$В_{тр\ з=}$ 308,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	308,00	$В_м$ (див. табл. 2.8.)
2. Основна заробітна плата	10900,64	$З_о$ (див. табл. 2.7.)
3.Додаткова заробітна плата	1090,06	$Зд = 0{,}1 \times Зо$ = 10900,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2637,95	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$=0,22* (10900,64+1090,06)
5. Накладні витрати	4360,25	$Внак. = 0{,}4 \times Зо$ =0,4 * 10900,64
6. Повна собівартість	19296,9	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 308,00+10900,64+1090,06+2637,95 +4360,25

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100=(19296,9*15)/100=2894,54$ грн (2.7)

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П = 19296,9+2894,54 = 22191,44$ грн; (2.8)

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$Ц_р=Ц_о+ПДВ = 22191,44 +22191,44 *0.2 = 26629,73$ грн; (2.9)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

Сучасний розвиток цифрових технологій та стрімке зростання криптовалютного ринку створюють нові виклики для безпеки інформації. Розробка віртуального асистента для безпечного перегляду криптовалютної статистики спрямована на забезпечення високого рівня кібербезпеки, оптимізацію робочого процесу та збереження конфіденційності даних користувачів. Інтеграція алгоритмів штучного інтелекту з передовими засобами захисту даних дозволяє оперативно отримувати актуальні дані при мінімізації ризиків несанкціонованого доступу та кібератак. Тема мого дипломного проєкту: Розробка віртуального асистента для безпечного перегляду криптовалютної статистики

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста

У процесі розробки та експлуатації віртуального асистента, орієнтованого на перегляд криптовалютної статистики, необхідно врахувати ряд потенційних ризиків. З одного боку, розробники стикаються з технічними загрозами, серед яких — кібератаки, витік конфіденційної інформації та експлуатація вразливостей програмного забезпечення. З іншого боку, інтенсивна робота з цифрованими даними може негативно впливати на фізичне та психологічне здоров'я: тривале перебування перед екраном, надмірне навантаження на зір і постійна емоційна напруга при моніторингу сучасних мережевих загроз. Саме тому важливо поєднувати заходи цифрової безпеки з принципами охорони праці, що забезпечують комфортне та здорове робоче середовище.

3.2 Гігієнічні вимоги до виробничого середовища

На робочому місці програміста повинні бути створені умови для безпечної та високопродуктивної праці. Це передбачає відповідність робочого середовища низці гігієнічних вимог, що впливають на фізичний та психологічний комфорт працівника.

Освітлення: Робоче місце повинно мати достатнє та рівномірне освітлення, щоб зменшити навантаження на зір програміста. Бажано використовувати

природне світло або комбіновані джерела освітлення.

Мікроклімат: Температура в приміщенні повинна бути комфортною (18–24°C), а рівень вологості підтримуватися у межах 40–60%. Необхідна також регулярна вентиляція для підтримки якісного повітряного обміну.

Ергономічність робочого місця: Робочий стіл і крісло мають відповідати принципам ергономіки, що сприятиме зменшенню ризиків розвитку захворювань опорно-рухового апарату.

Рівень шуму: Оптимальний рівень фонових шумів не повинен перевищувати 50–60 дБ. Надмірний шум може призводити до стресу, зниження концентрації та продуктивності.

Електромагнітні випромінювання: Використання захисних фільтрів на екранах, правильне розташування електронних пристроїв та дотримання правил електробезпеки допомагають знизити негативний вплив випромінювання.

Також важливо врахувати психоемоційний комфорт програміста: забезпечення можливості регулярного відпочинку, зручного простору для релаксації та достатнього фізичного руху протягом робочого дня.

Робоче місце програміста має бути організоване відповідно до ергономічних стандартів, забезпечуючи комфортні умови праці, які сприяють збереженню здоров’я та підвищенню продуктивності.

Регулювання меблів – робочий стіл та крісло повинні мати можливість індивідуального налаштування відповідно до росту працівника, щоб підтримувати правильну поставу та зменшити навантаження на м’язи та суглоби.

Розташування монітора – екран має бути встановлений так, щоб його верхня межа знаходилася на рівні очей, на відстані 60–90 см, оптимально – близько 70 см. Частота оновлення зображення повинна бути не менше 70 Гц, що допомагає уникнути зайвої втоми очей.

Освітлення – робоче місце рекомендується розташовувати перпендикулярно до вікон, щоб мінімізувати відблиски на екрані. Освітлення має бути рівномірним та достатнім, що сприяє зменшенню навантаження на зір.

Оздоблення робочої поверхні – стіл повинен бути пофарбований у матові

кольори, щоб уникнути небажаних відблисків.

Працівники, що працюють з відеодисплейними терміналами (ВДТ), проходять попередні медичні огляди при працевлаштуванні, а також періодичні огляди впродовж трудової діяльності, відповідно до наказу Міністерства охорони здоров'я України № 45.

При оцінці придатності до роботи з ВДТ враховуються:

- Гострота зору
- Параметри рефракції
- Стан бінокулярного апарату ока
- Загальний стан здоров'я

Дотримання цих вимог у поєднанні з профілактичними заходами сприяє підтримці здоров'я працівників та ефективності їхньої роботи.

Електроустановки повинні відповідати нормативним вимогам, передбаченим Правилами улаштування електроустановок (ПУЕ), Правилами технічної експлуатації споживачів (ПТЕ), Правилами техніки безпеки під час експлуатації електроустановок (ПТБ) та іншими регламентованими документами.

Основні правила експлуатації електропроводки:

З'єднання, розгалуження та закінчення електропроводів і кабелів необхідно виконувати виключно за допомогою зварювання, паяння, опресування або спеціальних затисків. Використання скручування жил проводів категорично забороняється.

Заборонені дії при роботі з електрообладнанням:

Прокладка кабелів через складські приміщення, пожежонебезпечні або вибухонебезпечні зони.

Експлуатація проводів із пошкодженою ізоляцією, що може призвести до короткого замикання та виникнення пожежонебезпечної ситуації.

Залишення під напругою неізольованих дротів та кабелів, що становить небезпеку для персоналу.

Використання саморобних подовжувачів, які не відповідають вимогам ПУЕ, що може призвести до аварій.

Застосування нестандартних електронагрівальних пристроїв для обігріву приміщень або використання ламп розжарювання як засобу обігріву.

Експлуатація пошкоджених електричних розеток, вимикачів та інших приладів, що може викликати несправності або електротравми.

Використання телефонного чи радіопроводу як електромережевого кабелю, що не відповідає вимогам безпеки.

Залишення електрообладнання під напругою без нагляду, що може спричинити перегрів або коротке замикання.

3.3 Пожежна безпека

Робоче приміщення, що відповідає вимогам ПБЕ та ОНТП 24–86 у сфері вибухово-пожежної безпеки, класифікується як об’єкт категорії «В».

Основними потенційними причинами виникнення пожежі в такому приміщенні є:

1. Коротке замикання електропроводки;
2. Використання побутових електрорадіоприладів;
3. Недотримання встановлених норм протипожежного захисту.

Виробничі приміщення, технологічні установки та будівлі повинні бути обладнані першоджерельними засобами пожежогасіння, до яких належать:

- вогнегасники,
- контейнери з піском,
- негорючі покривала з теплоізоляційного матеріалу,
- високоміцні тканинні вироби тощо.

Ці засоби повинні відповідати нормативним вимогам, затвердженим документами з технологічного проектування та Правилами пожежної безпеки в Україні (НАПБ А.О1.001-2014). Вогнегасники слід встановлювати в легкодоступних, добре помітних місцях (наприклад, в коридорах, біля входів та виходів або у зонах підвищеного ризику виникнення пожежі), захищаючи їх від прямого сонячного випромінювання та впливу опалювальних приладів. Розміщення вогнегасників має забезпечувати їхнє повне відкриття, причому вони встановлюються не вище 1,5 м від підлоги та на безпечній відстані від дверей.

Рисунок 3.2. Засоби пожежогасіння

Також засоби пожежогасіння (рис.3.2) не повинні заважати евакуації персоналу. Виробничі приміщення повинні бути обладнані кількома запасними виходами, доступ до яких має бути забезпечений у будь-який час. Двері, що ведуть до запасних виходів, мають бути позначені зрозумілими і яскравими освітленими написами, наприклад, «Запасний вихід», що сприяє чіткому сприйняттю інформації навіть у стресовій ситуації. План евакуації слід розмістити у видимому та доступному місці біля основного виходу, а також, за можливості, на всіх рівнях приміщення для забезпечення швидкого реагування у надзвичайному випадку.

Окрім базового розміщення знакових вказівників та плану евакуації, важливим аспектом є систематичний огляд та технічне обслуговування пожежного обладнання. Усі засоби пожежогасіння повинні проходити регулярну перевірку на справність, що включає тестування електронних компонентів, датчиків пожежної тривоги та систем автономного живлення. Такі заходи дозволяють виявити можливі несправності до моменту виникнення евенту та своєчасно вжити заходів для їх усунення.

ВИСНОВКИ

У даному дипломному проекті було розроблено комплексну систему – віртуального асистента для безпечного перегляду криптовалютної статистики із застосуванням Telegram-бота. Проект реалізовано з використанням сучасних технологій асинхронного програмування, зовнішніх API (CoinGecko та Etherscan) для отримання ринкових даних, а також сховища даних Redis для кешування та збереження як тимчасових, так і постійних даних. Структурна організація застосунку побудована за принципом модульності, що дозволяє чітко розділити відповідальність за обробку запитів, бізнес-логіку та управління даними. Такий підхід сприяє високій продуктивності, стабільності та безпеці системи.

Основний модуль інтерфейсу з користувачем забезпечує зручну, інтуїтивну та безпечну взаємодію через Telegram Bot API. Модуль обробки бізнес-логіки виконує валідацію запитів, асинхронні звернення до зовнішніх API, агрегацію отриманих даних та їх подальше форматування для користувача. Завдяки використанню Redis для кешування даних знижується кількість повторних запитів до зовнішніх сервісів, що дозволяє оптимізувати використання системних ресурсів і забезпечити оперативну відповідь навіть при високих навантаженнях.

Тестування роботи застосунку підтвердило його коректну та безпечну функціональність на всіх етапах: від початкової ініціації і переходу до головного меню до перевірки балансу, останніх транзакцій та поточного курсу криптовалют. Результати тестування свідчать про повну відповідність отриманих даних офіційним джерелам (Etherscan, CoinGecko), а також про стабільність і безпечність передачі конфіденційної інформації між користувачем та системою. Особлива увага була приділена захисту даних: всі операції здійснюються через зашифровані канали зв’язку, що гарантує високий рівень інформаційної безпеки.

Завдяки модульній структурі, можливе подальше розширення функціональності за рахунок інтеграції нових сервісів та оптимізації алгоритмів обробки даних. Реалізація проекту засвідчує ефективне застосування сучасних технологій і відповідає вимогам інформаційної безпеки та високої продуктивності в умовах динамічного ринку криптовалют.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Петренко В.В., Сидоренко О.О. Актуальні проблеми криптографії та інформаційної безпеки: монографія – Львів : ЛНУ ім. І. Франка, 2019. – 412 с.
2. Сидоренко І.М., Забіла Ю.О. Основи криптовалют та блокчейн технологій: навчальний посібник – Харків: ХНУ, 2020. – 300 с.
3. Гончаренко С. Сучасні аспекти інформаційної безпеки: монографія – Дніпро: Дніпровський державний університет, 2018. – 360 с.
4. Олійник, Є.О. Розробка систем штучного інтелекту для аналізу фінансових даних – Одеса: АстроПринт, 2019. – 280 с.
5. Логвін, А.В. Телеграм-боти: технології розробки та забезпечення безпеки – Львів: ЛІРІК, 2021. – 190 с.
6. Мельничук П.К. Асинхронне програмування в Python: навчальний посібник – Київ: Центр учбової літератури, 2020. – 220 с.
7. Волошин Ю.Ф. Блокчейн технології в сучасній економіці: монографія – Київ: Інститут економіки, 2021. – 350 с.
8. Калінін О.Л. Концепції безпеки в цифрових системах: підручник – Львів: Видавництво «Світ», 2020. – 400 с.
9. Бойко С.І. Розробка мобільних додатків з використанням Python та Telegram Bot API: навчальний посібник – Одеса: 2021. – 210 с.
10. Воронов С.Б. Інтернет-технології та безпека транспорту даних: монографія – Дніпро: Дніпровський національний університет, 2019. – 298 с.
11. CoinGecko. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.coingecko.com/uk. – (Дата звернення: 05.05.2025).
12. Etherscan. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://etherscan.io/. – (Дата звернення: 07.05.2025).
13. Telegram Bot API. Технічна документація [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api. – (Дата звернення: 05.05.2025).
14. Redis. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://redis.io/. – (Дата звернення: 05.05.2025).

ДОДАТОК А. Фрагмент коду (Python) файлів transactions.py та menu.py віртуального асистента

Файл transactions.py

```
import logging
import asyncio
from datetime import datetime
from aiogram import types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text
from requests import get
from pycoingecko import CoinGeckoAPI
from states import TransCheck
from loader import dp, bot
# Конфігурація API та константи
API_KEY = "91NEEET9HB38KWHAP46M8314WJ39ZNYXI5"
BASE_URL = "https://api.etherscan.io/api"
ETHER_VALUE = 10 ** 18

# Ініціалізація API CoinGecko (при потребі використовувати)
cg = CoinGeckoAPI()

def make_api_url(module, action, address, **kwargs):
    """
    Формує URL для запиту до Etherscan API із заданими параметрами.
    """
    url = BASE_URL +
f"?module={module}&action={action}&address={address}&apikey={API_KEY}"
    for key, value in kwargs.items():
        url += f"&{key}={value}"
    return url

# Об’єднаний обробник запиту на перевірку останньої трансакції у різних мовах
@dp.message_handler(Text(equals=["Перевірити останню Ethereum транзакцію",
                                 "Check last Ethereum transaction",
                                 "Последняя Ethereum транзакция"]))
async def request_wallet_for_transaction(message: types.Message):
    prompt = "Введіть номер гаманця:" if any(term in message.text for term in ["Перевірити",
"Последняя"]) else "Insert wallet number:"
    await message.answer(prompt)
    await TransCheck.askForTrans1.set()
@dp.message_handler(state=TransCheck.askForTrans1)
async def process_transaction_request(message: types.Message, state: FSMContext):
    wallet_address = message.text
    await state.update_data(address=wallet_address)
    # Функція для отримання трансакцій по вказаній адресі
    async def get_transactions(address: str):
        transactions_url = make_api_url("account", "txlist", address, startblock=0,
endblock=99999999, page=1, offset=10000, sort="asc")
        try:
            response = get(transactions_url)
            result = response.json().get("result", [])
        except Exception as e:
```

```
            logging.error(f"Error parsing response: {e}")
            result = []
        messages = []
        for tx in result:
            to_addr = tx.get("to", "N/A")
            from_addr = tx.get("from", "N/A")
            try:
                value_eth = int(tx.get("value", 0)) / ETHER_VALUE
            except Exception:
                value_eth = 0
            try:
                gas_cost = int(tx.get("gasUsed", 0)) * int(tx.get("gasPrice", 0)) / ETHER_VALUE
            except Exception:
                gas_cost = 0
            try:
                tx_time = datetime.fromtimestamp(int(tx.get("timeStamp", 0)))
            except Exception:
                tx_time = "N/A"
            msg_part = (f"---------------\n"
                        f"To: {to_addr}\n"
                        f"From: {from_addr}\n"
                        f"Value: {value_eth} ETH\n"
                        f"Gas Cost: {gas_cost} ETH\n"
                        f"Time: {tx_time}")
            messages.append(msg_part)
        full_message = "\n".join(messages) if messages else "No transactions found."
        await bot.send_message(message.from_user.id, full_message)

    await get_transactions(wallet_address)
    await state.finish()
```

Файл menu.py

```
import logging
import asyncio
from datetime import datetime
from aiogram import types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Command, Text
from requests import get
from pycoingecko import CoinGeckoAPI
from states import BalanceCheck
from loader import dp, bot
# Конфігурація API та константи
API_KEY = "91NEEET9HB38KWHAP46M8314WJ39ZNYXI5"
BASE_URL = "https://api.etherscan.io/api"
ETHER_VALUE = 10 ** 18
# Ініціалізація CoinGecko API
cg = CoinGeckoAPI()
from keyboards.default import menu
from keyboards.default.menu import menu_eng, menu_ru
from keyboards.inline.choice_buttons import choice
```

```
def make_api_url(module, action, address, **kwargs):
    """
    Формує URL для запиту до Etherscan API із заданими параметрами.
    """
    url = BASE_URL +
f"?module={module}&action={action}&address={address}&apikey={API_KEY}"
    for key, value in kwargs.items():
        url += f"&{key}={value}"
    return url
# Обробник команди стартового меню
@dp.message_handler(Command("menu"))
async def show_menu(message: types.Message):
    await message.answer("Оберіть, що вас цікавить", reply_markup=menu)
# Об'єднаний обробник запиту балансу для користувача (підтримка декількох мов)
@dp.message_handler(Text(equals=["Дізнатися баланс гаманця", "Ethereum Wallet
Balance", "Узнать баланс Ethereum кошелька"]))
async def request_wallet_for_balance(message: types.Message):
    prompt = ("Введіть номер гаманця:" if "Дізнатися" in message.text
              else "Insert wallet number:" if "Ethereum" in message.text
              else "Введите номер кошелька:")
    await message.answer(prompt)
    await BalanceCheck.askForBalance1.set()
@dp.message_handler(state=BalanceCheck.askForBalance1)
async def process_balance_request(message: types.Message, state: FSMContext):
    wallet_address = message.text
    await state.update_data(address=wallet_address)
    def get_account_balance(address: str):
        get_balance_url = make_api_url("account", "balance", address, tag="latest")
        try:
            response = get(get_balance_url)
            result_json = response.json()
            balance_int = int(result_json.get("result", "0"))
            balance = balance_int / ETHER_VALUE
        except Exception as e:
            logging.error(f"Error retrieving balance: {e}")
            balance = None
        return balance
    balance = get_account_balance(wallet_address)
    if balance is not None:
        await message.answer(f"Balance: {balance} ETH")
    else:
        await message.answer("Error retrieving balance. Please try again.")
    await state.finish()
# Об'єднані обробники для зміни мови
@dp.message_handler(Text(equals=["Змінити мовуUAGB 🌎", "Change LanguageUAGB 🌎",
"Сменить языкUAGB 🌎"]))
async def change_language(message: types.Message):
    if any(term in message.text for term in ["Змінити", "Сменить"]):
        await message.answer("Оберіть, що вас цікавить", reply_markup=menu)
    elif "Change" in message.text:
        await message.answer("Choose what do you want to do", reply_markup=menu_eng)
    else:
        await message.answer("Выберите, что вас интересует", reply_markup=menu_ru)
```

```
# Об'єднаний обробник для запиту курсу криптовалют
@dp.message_handler(Text(equals=["Дізнатися курс криптовалюти", "Узнать курс
криптовалюты", "Cryptocurrency rates"]))
async def request_crypto_rate(message: types.Message):
    text = ("Курси на даний момент" if "Дізнатися" in message.text
            else "Курсы на данный момент:" if "Узнать" in message.text
            else "Current rates:")
    await message.answer(text=text, reply_markup=choice)
# Callback-обробники для кожної криптовалюти (уникальні імена функцій)
@dp.callback_query_handler(text='ethereum')
async def handle_ethereum(call: types.CallbackQuery):
    price = cg.get_price(ids='ethereum', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"Ethereum: 1 ETH = ${price['ethereum']['usd']:.2f}")
@dp.callback_query_handler(text='bitcoin')
async def handle_bitcoin(call: types.CallbackQuery):
    price = cg.get_price(ids='bitcoin', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"Bitcoin: 1 BTC = ${price['bitcoin']['usd']:.2f}")
@dp.callback_query_handler(text='litecoin')
async def handle_litecoin(call: types.CallbackQuery):
    price = cg.get_price(ids='litecoin', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"Litecoin: 1 LTC = ${price['litecoin']['usd']:.2f}")
@dp.callback_query_handler(text='tether')
async def handle_tether(call: types.CallbackQuery):
    price = cg.get_price(ids='tether', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"Tether: 1 USDT = ${price['tether']['usd']:.2f}")
@dp.callback_query_handler(text='binancecoin')
async def handle_binancecoin(call: types.CallbackQuery):
    price = cg.get_price(ids='binancecoin', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"BNB: 1 BNB = ${price['binancecoin']['usd']:.2f}")
@dp.callback_query_handler(text='solana')
async def handle_solana(call: types.CallbackQuery):
    price = cg.get_price(ids='solana', vs_currencies='usd')
    await call.answer(cache_time=60)
    await call.message.answer(f"Solana: 1 SOL = ${price['solana']['usd']:.2f}")
@dp.callback_query_handler(text='cancel')
async def handle_cancel(call: types.CallbackQuery):
    await call.message.edit_reply_markup()
```

ДОДАТОК Б. Слайди мультимедійної презентації

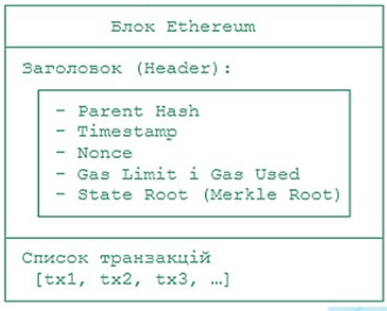
Розробка віртуального асистента для безпечного перегляду криптовалютної статистики

Мірзалієв Кирило, гр.4КБ-02

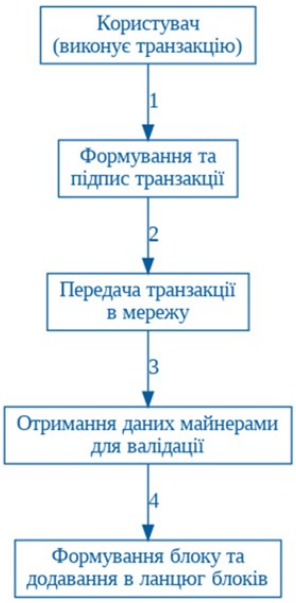
Порівняння ключових параметрів Ethereum

Показник	Ethereum
Тип блокчейну	Публічний, децентралізований
Основна валюта	Ether (ETH)
Консенсус	Первинно - Proof of Work (PoW); у процесі перехід до Proof of Stake (PoS)
Середній час блоку	~13 секунд (залежить від навантаження мережі)
Хеш-функція	Keccak-256
Смарт-контракти	Підтримка мов Solidity, Vyper та інших
Механізм оплати транзакцій	Газова модель: вартість транзакції обчислюється за формулою Transaction Cost (ETH) = Gas Price (ETH/unit) × Gas Used (units)

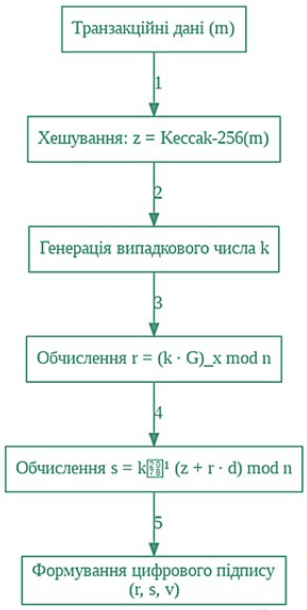
Основна структура блоку Ethereum



Процес обробки транзакції
у мережі Ethereum



Діаграма процесу
цифрового підпису транзакції



Порівняння показників платформ з підтримкою чат-ботів

Платформа	API Доступність	Ключові переваги	Обмеження
Telegram	Повністю відкритий (Telegram Bot API)	Швидка інтеграція, гнучкість у розробці, підтримка як синхронного режиму (пулінг), так і webhook-технології	Бот-повідомлення не мають end-to-end шифрування
WhatsApp	Обмежений, частково комерційний	Висока популярність, масова аудиторія	Суворі обмеження API, складніший процес реєстрації
Facebook Messenger	Відкритий, але зі специфічними вимогами	Інтеграція зі службою Facebook, можливість комплексної розробки бізнес-чатів	Інтеграція може вимагати додаткових витрат на сертифікацію
Viber	Відкритий API	Підтримка інтерактивних функцій, можливості для маркетингових кампаній	Менший ринок користувачів порівняно з Telegram

Схема інтеграції чат-бота у Telegram

Схема взаємодії компонентів системи

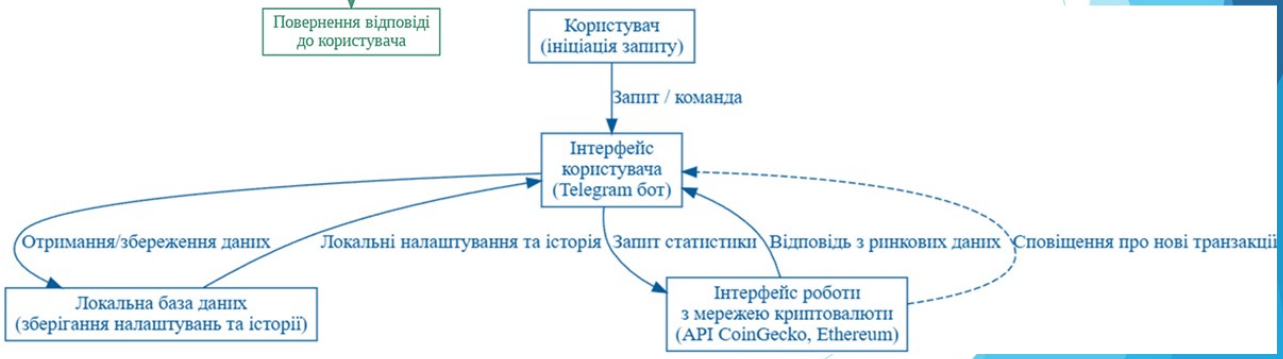
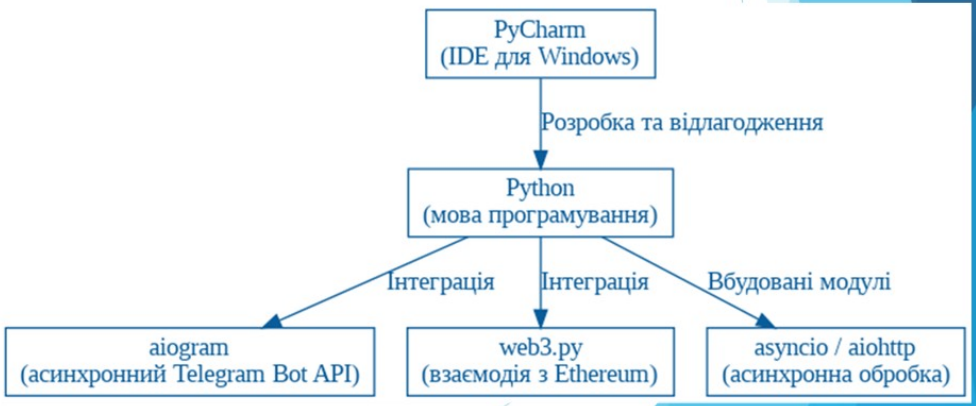


Схема взаємодії системи з джерелами даних криптовалютного ринку

Схема взаємодії між основними компонентами інтерфейсу

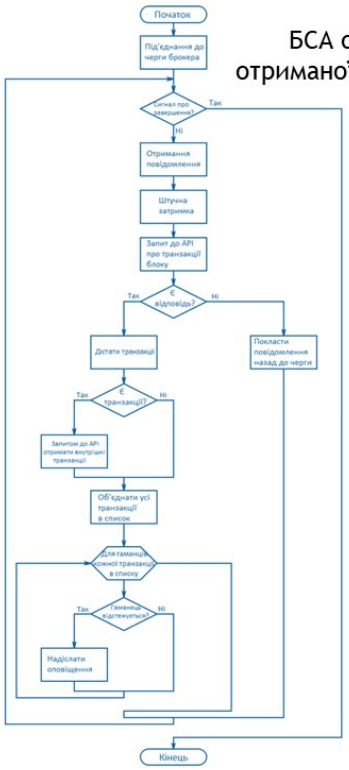
Загальна діаграма роботи асистента

Діаграма основних засобів розробки та бібліотек проекту



БСА прийому нових блоків даних

БСА обробки отриманої інформації



БСА реєстрації адреси цільової мережі криптовалюти

Схема модуля обробки бізнес-логіки

Схема модуля інтерфейсу з користувачем

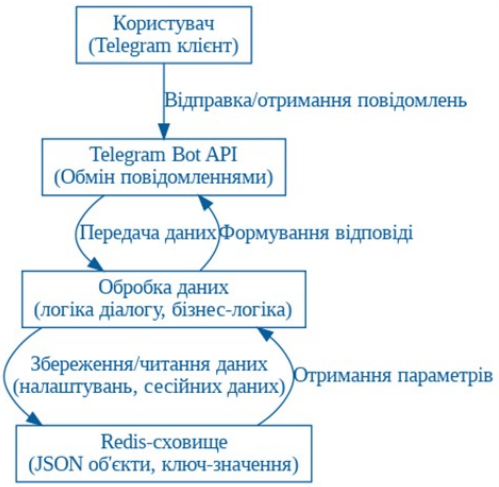


Схема бази даних управління основними даними застосунку

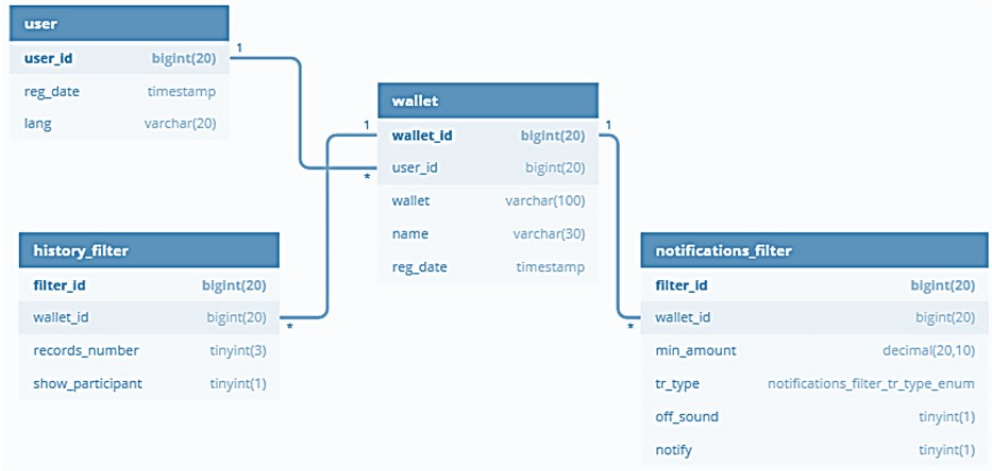
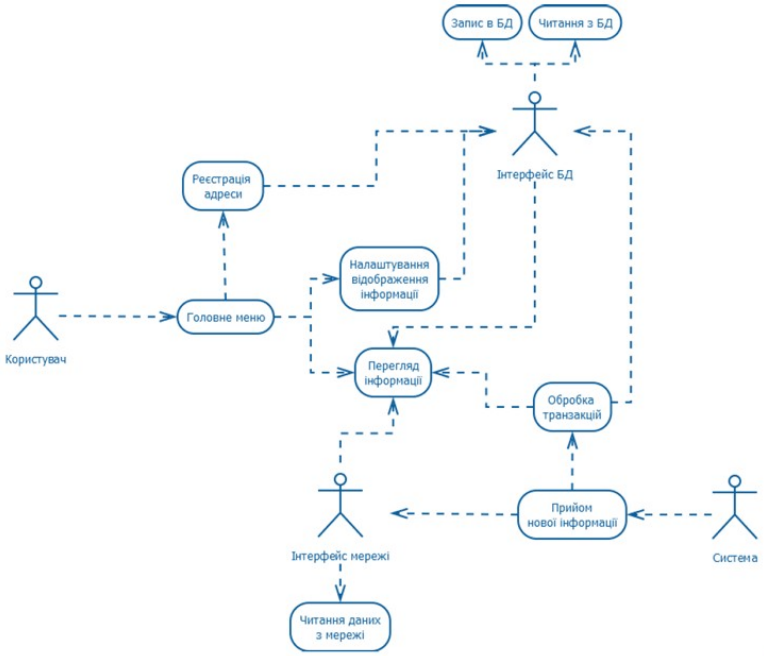
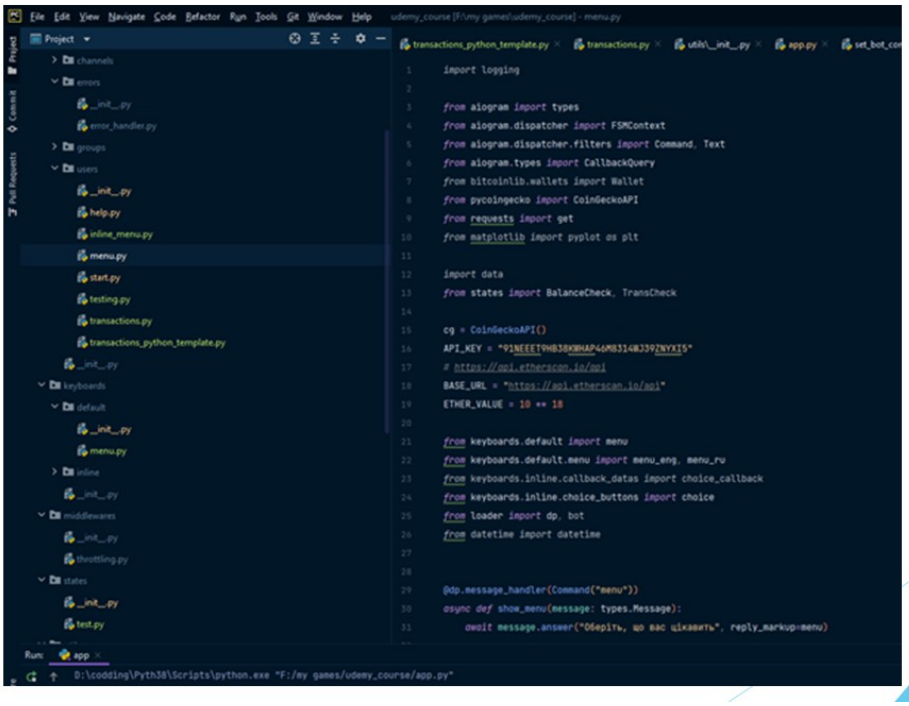


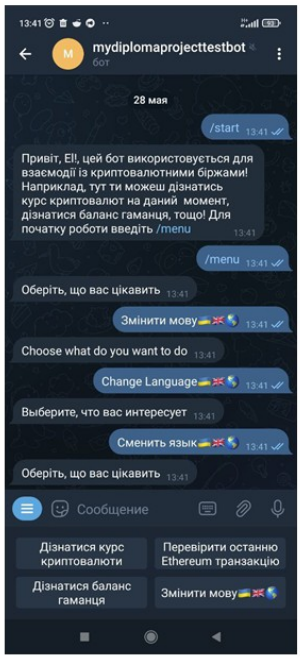
Схема управління даними застосунку віртуального асистента



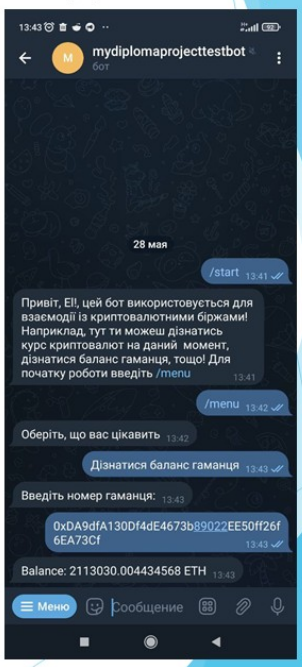
Структура проекту у ICP PyCharm



Привітальне повідомлення та зміна мови у головному меню віртуального асистента



Баланс гаманця, відображений у інтерфейсі асистенту



Вибір гаманців Ethereum

Rank	Address	Name Tag	Balance	Percentage	Txn Count
1	0x00000000219ab540356cbb839cbe05303d7705fa	Eth2 Deposit Contract	12,658,050.000069 Ether	10.62733945%	197,984
2	0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2	Wrapped Ether	5,817,396.53299427 Ether	4.88412099%	7,945,310
3	0xda9dfa130df4de4673b89022ee50ff26f6ea73cf	Kraken 13	2,113,030.00443456 Ether	1.77404001%	66
4	0xbe0eb53f46cd790cd13851d5eff43d12404d33e8	Binance 7	1,996,008.28210113 Ether	1.67579189%	1,092
5	0x73bceb1cd57c711feac4224d062b0f6ff338501e		1,900,505.52801049 Ether	1.59561049%	488
6	0x9bf4001d307dfd62b26a2f1307ee0c0307632d59		1,490,000.0180927 Ether	1.25096172%	103
7	0x4ddc2d193948926d02f9b1fe9e1daa0718270ed5	Compound: cETH Token	793,828.54129808 Ether	0.66647591%	277,097

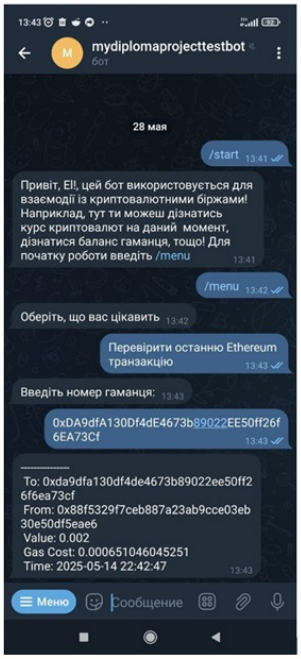
Перегляд балансу гаманця (дані з Etherscan)

Balance:	2,113,030.0044345678 Ether

Деталі транзакції за даними Etherscan

Transaction Hash:	0xed5e8eaba7d5860c5cc7fae449a1f53854041faa79f5aa58427ce7a3f0b5ca4c
Status:	Success
Block:	14775509 47361 Block Confirmations
Timestamp:	7 days 14 hrs ago (May-14-2025 07:42:47 PM +UTC) | Confirmed within 1 min
From:	0x88f5329f7ceb887a23ab9cce03eb30e50df5eae6
To:	0xda9dfa130df4de4673b89022ee50ff26f6ea73cf (Kraken 13)
Value:	0.002 Ether ($4.06)
Transaction Fee:	0.000651046045251 Ether ($1.32)

Перевірка транзакції
у інтерфейсі асистенту



Запит перевірки курсу криптовалют
у інтерфейсі асистенту

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мірзалієва Кирила Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Скорняков В'ячеслав Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка віртуального асистента для безпечного перегляду криптовалютної статистики*

Обсяг розрахунково-пояснювальної записки *78* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений реалізації віртуального асистента для безпечного перегляду криптовалютної статистики і складається з пояснювальної записки та мультимедійної презентації з відповідними схемами.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (Аналіз технологій для реалізації віртуального асистента; Розробка загальної моделі безпечного перегляду криптовалютної статистики; Вибір технологій розробки; Розробка БСА обробки статистики; Розробка модулів застосунку; Тестування роботи віртуального асистента), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 16 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні та функціональні схеми, діаграми та скріншоти, блок-схеми алгоритмів, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту

Розділення на handlers / markups / states, бізнес-логіку, кеш-рівень (Redis) і окремі API-клієнти робить код прозорим, легко масштабованим. Асинхронний стек — aiogram + aiohttp + asyncio — правильний вибір для I/O-навантажень. Використання HTTPS-запитів, токен-базової аутентифікації, кешування в Redis без зберігання приватних ключів підвищує рівень захисту.

д) основні недоліки дипломного проекту

Асистент лише «читає» дані. Було б цікаво перелічити, які є сценарії розширення (hardware-wallet / WalletConnect). Недостатня обґрунтованість вибору Redis як «довготривалої» БД: ризик втрати даних при аварійному перезапуску. Доцільно пояснити чому не обрали PostgreSQL для постійних таблиць.

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

«*20*» *червня* *2025 р.*

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мірзалієва Кирила Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка віртуального асистента для безпечного перегляду криптовалютної статистики*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 78 сторінок. У пояснювальній записці виконано огляд та аналіз методів створення чат-ботів та взаємодії з цифровою валютою, розроблено застосунок віртуального асистенту, що виконує дану задачу з урахуванням вимог до безпеки. Графічна частина складається з 16 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Мірзалієв Кирило поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Мірзалієв Кирило під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Мірзалієв Кирило показав вміння організовано працювати над поставленим завданням, застосовувати знання у сфері безпеки комп'ютерних систем і мереж, програмування, використовуючи сучасні комп'ютерні програмні засоби розробки, такі як ICP PyCharm, API CoinGecko, Etherscan, Redis*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Скорняков В'ячеслав Сергійович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерних технологій та програмної інженерії*

Підпис

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Мірзалієв К.С.,
здобувач освіти гр. 4КБ-02, та

Скорняков В.С.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка віртуального асистента для безпечного перегляду криптовалютної статистики» (автор роботи – Мірзалієв К.С., керівник роботи – Скорняков В.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Мірзалієв К.С. /

Керівник _____ / Скорняков В.С. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Мірзалієва Кирила Сергійовича

на тему *Розробка віртуального асистента для безпечного перегляду криптовалютної статистики*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 05.06.2025 р. значення коефіцієнту подібності в роботі становить 16,58%, коефіцієнт цитування – 2,55%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Мірзалієва К.С.* (П.І.Б.)

проведена «*16*» *червня* 20*25* р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/5/2025
Дата редагування 6/5/2025

Документ прийнятий

Звіт подібності

метадані

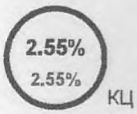
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка віртуального асистента для безпечного перегляду криптовалютної статистики

Автор
Мірзалієв Кирило Сергійович
Науковий керівник / Експерт
Скорняков В'ячеслав Сергійович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



25	13780	113588
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	20
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	77

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	103 0.75 %
2	Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів [redacted] 6/3/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	80 0.58 %
3	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	71 0.52 %

4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	58 0.42 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	57 0.41 %
6	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	52 0.38 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	43 0.31 %
8	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	38 0.28 %
9	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	33 0.24 %
10	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	31 0.22 %

з домашньої бази даних (0.97 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аналіз ефективності алгоритмів стискування відео-даних для стрімінгових сервісів [redacted] 6/3/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**104 (2) 0.75 %**
2	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	29 (2) 0.21 %

з програми обміну базами даних (2.23 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Система обробки транзакцій в мережі криптовалюти Ethereum [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**210 (20) 1.52 %**
2	Розробка програмного забезпечення Телеграм-бота магазину [redacted] 6/5/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	55 (6) 0.40 %
3	ФКПІ_ІПЗ_2023_121_ПивоварВ.Д [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	21 (2) 0.15 %
4	КОМП`ЮТЕРНА СИСТЕМА ДІАГНОСТУВАННЯ ПСИХІЧНОГО СТАНУ ЛЮДИНИ [redacted] 6/10/2021 National University Chernihiv Politechnika (NUCP) 2 (Дипломні роботи)	21 (2) 0.15 %

з Інтернету (13.39 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	**710 (58) 5.15 %**
2	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	**339 (10) 2.46 %**
3	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	**204 (14) 1.48 %**

4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	114 (3) 0.83 %
5	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	85 (3) 0.62 %
6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	56 (3) 0.41 %
7	https://card-file.ontu.edu.ua/bitstreams/62baa43e-b968-4993-bb54-8cf8761a89b2/download	51 (4) 0.37 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	42 (3) 0.30 %
9	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	39 (3) 0.28 %
10	http://prometey.in.ua/ua/vimogi/	39 (3) 0.28 %
11	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	35 (2) 0.25 %
12	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	23 (1) 0.17 %
13	https://card-file.ontu.edu.ua/server/api/core/bitstreams/f5042058-9544-4ac7-bd33-42bd733c8e6b/content	23 (2) 0.17 %
14	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	20 (1) 0.15 %
15	https://card-file.ontu.edu.ua/bitstreams/11562741-24e6-4201-bc41-a00c8013fca1/download	19 (1) 0.14 %
16	https://docplayer.net/39668359-Issn-x-7-2016.html	12 (2) 0.09 %
17	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	9 (1) 0.07 %
18	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	9 (1) 0.07 %
19	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	9 (1) 0.07 %
20	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	7 (1) 0.05 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж» Група: 4КБ- 02

Дипломний проєкт
здобувача **освіти денної форми навчання КБ. 02.12.000.ДП**

МІРЗАЛІЄВА
КИРИЛА СЕРГІЙОВИЧА м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж» Група: 4 КБ-02

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проєкту на тему: _____
_____ **Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах)** Дипломник _____ **(Мірзалієв К.С.)**
_____ Керівник _____ **(Скорняков В.С.) Консультанти: з економічного розділу**_____ **(Канський М.Ю.)**