СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) Інформаційних технологій та електроніки

Кафедра Інформаційних технологій та програмування
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 126 Інформаційні системи та технології
(шифр і назва спеціальності)
спеціалізація Інформаційні системи та технології
(назва спеціалізації)

на тему Програмне забезпечення для відстеження та аналізу криптовалютного ринку

Виконав: студент групи ІСТ-20д _____ О.О. Шкаров
(підпис) (ініціали і прізвище)

Керівник _____ Д.М. Марченко
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О. І. Захожай
(підпис) (ініціали і прізвище)

Рецензент Доц., д.т.н. Лифар В.О.

Київ – 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) Інформаційних технологій та електроніки
Кафедра Інформаційних технологій та програмування
(повна назва кафедри)
Освітній ступінь бакалавр
(бакалавр, магістр)
спеціальність 126 Інформаційні системи та технології
(шифр і назва спеціальності)
спеціалізація Інформаційні системи та технології
(назва спеціалізації)

Захожай О.І.
"___" _____ 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Шкаров Олексій Олександрович
(прізвище, ім'я, по-баткові)

1. Тема роботи Програмне забезпечення для відстеження та аналізу криптовалютного ринку

Керівник роботи Марченко Дмитро Миколайович, Проф., д.т.н.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від "___"_____20__року №___

2.Строк подання студентом роботи_____
3. Вихідні дані до роботи Об'єктом кваліфікаційної роботи є розробка програмного забезпечення для відстеження та аналізу криптовалютного ринку

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітична частина, з висвітленням наступних питань: принцип роботи блокчейну, аналіз існуючих рішень. Методи та алгоритми аналізу, технічний, фундаментальний та сентиментальний аналізи, математичні та статистичні методи аналізу. Основна частина, в якій висвітлено: розробку алгоритму, архітектуру програмного забезпечення, вибір технологій для розробки, вибір бази даних та підключення до Telegram бота, етап створення програмного забезпечення та тестування. Висновок. Перелік використаних джерел.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників)
ДодатокА

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Огляд предметної області	10.04.24-18.04.24	
2	Аналіз існуючих рішень	19.04.24-25.04.24	
3	Визначення вимог для програмного забезпечення	26.04.24-03.05.24	
4	Проєктування архітектури програмного забезпечення	04.05.24-12.05.24	
5	Розробка програмного забезпечення	12.05.24-27.05.24	
6	Апробація програмного забезпечення	28.05.24-30.05.24	
7	Оформлення пояснювальної записки	31.05.24-04.06.24	
8	Підготовка та подання дипломної роботи	04.06.24-07.06.24	

Студент _____ О.О. Шкаров
(підпис) (ініціали і прізвище)

Керівник роботи _____ Д.М. Марченко
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІСТ-20д Шкаров О.О.

Науковий керівник

Професор, д.т.н. _____ Марченко Д.М.

Оцінка наукового керівника: _____

Рецензент Лифар В.О., доц., д.т.н._____
ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

Голова ЕК
Професор кафедри ІТП
д.т.н. _____ Меняйленко О.С.
підпис

ЗМІСТ

ВСТУП

Ринок криптовалют у сучасному світі набуває все більшої популярності і значущості. Від моменту появи першої криптовалюти, Bitcoin, у 2009 році, цей сегмент фінансового ринку значно розширився і включає в себе тисячі різних криптовалют. Вони використовуються не лише як засіб обміну, але й як інвестиційний інструмент, що привертає увагу мільйонів інвесторів та трейдерів по всьому світу. Незважаючи на високу волатильність та ризики, пов'язані з криптовалютами, їх популярність зростає, що робить питання відстеження та аналізу ринку надзвичайно актуальним.

Мета даної дипломної роботи полягає у розробці програмного забезпечення, яке дозволяє ефективно відстежувати та аналізувати ринок криптовалют, з подальшою інтеграцією в Telegram бот для забезпечення зручності використання і мультиплатформеності. Такий підхід дозволить користувачам оперативно отримувати актуальну інформацію про стан ринку, здійснювати аналіз даних і приймати обґрунтовані рішення щодо купівлі або продажу криптовалют.

Завданнями роботи є:

1. Аналіз існуючих методів та інструментів для відстеження і аналізу криптовалютного ринку.
2. Розробка алгоритмів обробки даних та прогнозування курсів криптовалют.
3. Створення програмного забезпечення, що дозволяє збирати, обробляти та аналізувати дані з криптовалютних бірж.
4. Інтеграція розробленого ПЗ з Telegram ботом для забезпечення зручного доступу користувачів до інформації.
5. Проведення тестування та оцінка ефективності створеного програмного забезпечення.

Об'єктом дослідження є криптовалютний ринок, його структура та динаміка. Предметом дослідження виступають алгоритми та методи, що використовуються для аналізу та прогнозування курсів криптовалют, а також засоби інтеграції з платформами обміну повідомленнями.

Методи дослідження включають аналіз літератури та існуючих рішень, розробку алгоритмів, програмування та тестування програмного забезпечення, а також проведення експериментів для оцінки його ефективності.

Ця робота є актуальною через швидкий розвиток ринку криптовалют та необхідність у нових інструментах для його аналізу. Створене програмне забезпечення дозволить не лише підвищити ефективність торгівлі на криптовалютному ринку, але й забезпечить користувачів зручними засобами для отримання та аналізу інформації, що, у свою чергу, сприятиме прийняттю більш обґрунтованих інвестиційних рішень.

РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Криптовалюти з'явилися як відповідь на потребу у децентралізованій формі грошей, що не залежить від урядів та фінансових установ. Першою та найвідомішою криптовалютою є Bitcoin, створений у 2008 році анонімною особою або групою осіб під псевдонімом Сатоші Накамото [1]. У своїй науковій роботі "Bitcoin: A Peer-to-Peer Electronic Cash System" Накамото описав механізм децентралізованої системи обліку, яка отримала назву блокчейн[1]. Ця технологія дозволила вирішити проблему подвійного витрачання коштів без потреби у центральному контролюючому органі.

Поява Bitcoin започаткувала нову еру у фінансовій сфері. З моменту свого запуску в 2009 році, Bitcoin став першим успішним прикладом використання блокчейну для створення цифрових грошей. На відміну від традиційних валют, біткоїн децентралізований і не контролюється жодною центральною владою чи установою. Це робить його незалежним від урядових політик та економічних умов, що сприяє його популярності серед користувачів, які шукають альтернативу традиційним фінансовим системам.

З моменту появи Bitcoin криптовалютний ринок зазнав значних змін. З'явилися тисячі інших криптовалют, таких як Ethereum, Ripple, Litecoin, які впроваджують нові концепції та технології [2]. Наприклад, Ethereum представив концепцію смарт-контрактів, які дозволяють виконувати автоматизовані угоди без участі посередників. Ці смарт-контракти є програмами, які автоматично виконують задані умови угоди, що забезпечує надійність та безпеку транзакцій.

Криптовалюти стали об'єктом значної уваги з боку інвесторів, що призвело до їх швидкого зростання та популяризації. Однак, разом з цим зростанням виникли і численні виклики, такі як висока волатильність, регуляторні ризики та загрози безпеці [3]. Незважаючи на ці виклики, криптовалюти продовжують розвиватися, пропонуючи нові можливості для інновацій у фінансовій сфері.

1.1 Принцип роботи блокчейну

Блокчейн - це децентралізована база даних, яка складається з блоків, що містять інформацію про транзакції. На відміну від традиційних баз даних, які керовані центральним органом, блокчейн розподілений між всіма учасниками мережі [4].

Принцип роботи блокчейну:

- Децентралізація: всі учасники мережі зберігають копію блокчейну, що забезпечує прозорість та захист від маніпуляцій.
- Прозорість: кожна транзакція є доступною для перегляду всіма учасниками мережі.
- Незмінність: інформація, записана в блокчейн, не може бути змінена або видалена, що забезпечує високий рівень безпеки.

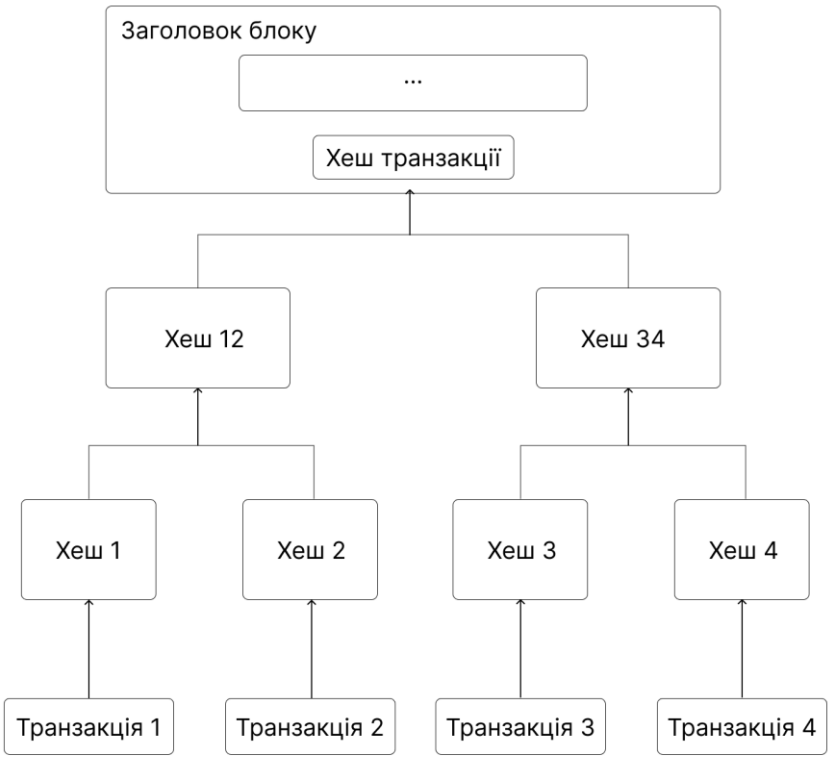

Рис. 1.1 — Робота блокчейну

Технологія блокчейну має широкий спектр застосувань, включаючи фінанси, логістику, управління ланцюгами постачання та багато іншого. Важливим аспектом технології блокчейн є її можливість забезпечувати високий рівень безпеки даних за рахунок криптографічних методів. Кожен

блок містить криптографічний хеш попереднього блоку, що створює нерозривний зв'язок між блоками та забезпечує незмінність даних.

Блокчейн-технологія забезпечує надійний та безпечний спосіб зберігання даних, який не залежить від центрального органу. Це робить блокчейн ідеальним для використання у різних сферах, де важлива безпека та прозорість даних [4]. Наприклад, у фінансових транзакціях блокчейн дозволяє зменшити витрати на обробку платежів, підвищити швидкість транзакцій та забезпечити надійний захист від шахрайства[5].

1.2 Існуючі рішення

Сьогодні існує безліч програмних рішень для відстеження та аналізу криптовалютного ринку. Кожне з них пропонує унікальні можливості та функції, які допомагають користувачам отримати актуальну інформацію про ринок, проводити аналіз та приймати обґрунтовані рішення. Однак, ці сервіси мають свої переваги та недоліки, які можуть впливати на вибір користувачів.

1.2.1 CoinMarketCap

CoinMarketCap [6] є одним з найпопулярніших і найвідоміших сервісів для відстеження криптовалют. Він пропонує широкий спектр інформації про ринок криптовалют, включаючи ціни, обсяги торгів, ринкову капіталізацію та інші важливі показники.

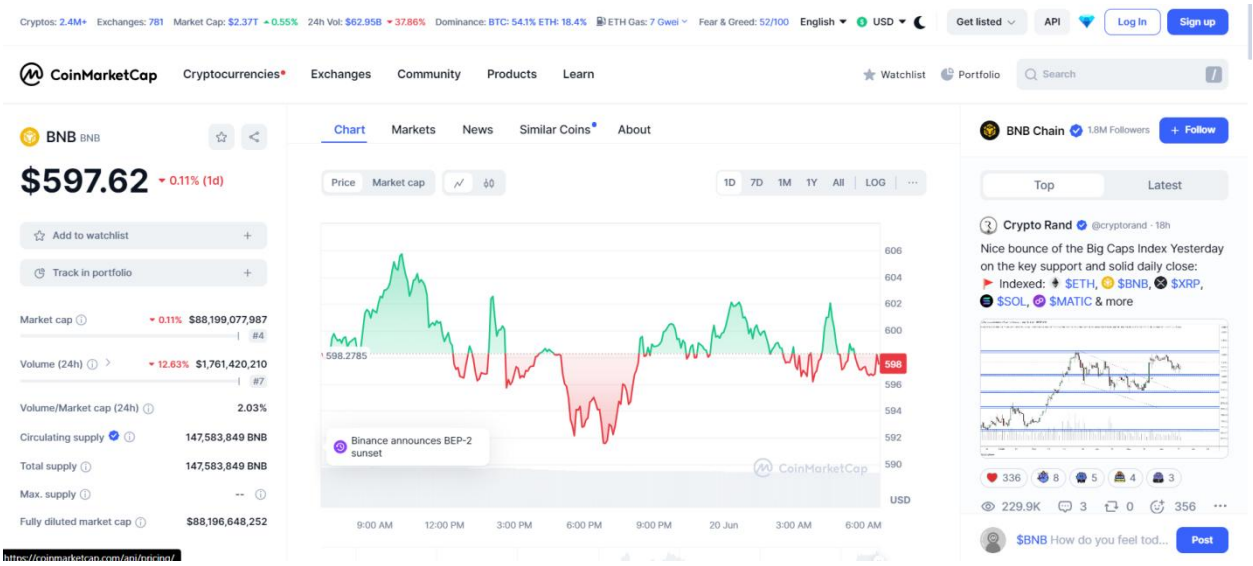

Рис. 1.2 — Інтерфейс CoinMarketCap

Переваги:

- Широке покриття ринку: CoinMarketCap надає дані про велику кількість криптовалют та бірж, що дозволяє користувачам отримувати повну картину ринку.
- Простий та зручний інтерфейс: Інтерфейс користувача є інтуїтивно зрозумілим та легким у використанні, що робить сервіс доступним як для новачків, так і для досвідчених трейдерів.
- Актуальність даних: Сервіс регулярно оновлює дані, забезпечуючи користувачів найсвіжішою інформацією.

Недоліки:

- Обмежені можливості для технічного аналізу: CoinMarketCap надає обмежені інструменти для технічного аналізу, що може бути недостатньо для деяких користувачів.
- Залежність від сторонніх джерел: Дані сервісу збираються з різних бірж, що може призводити до неточностей або затримок в оновленні інформації.

1.2.2 CoinGecko

CoinGecko [7] є ще одним популярним сервісом для відстеження криптовалют, який надає детальну інформацію про різноманітні криптовалюти та біржі. CoinGecko також пропонує аналітичні інструменти та рейтинги криптовалют за різними критеріями.

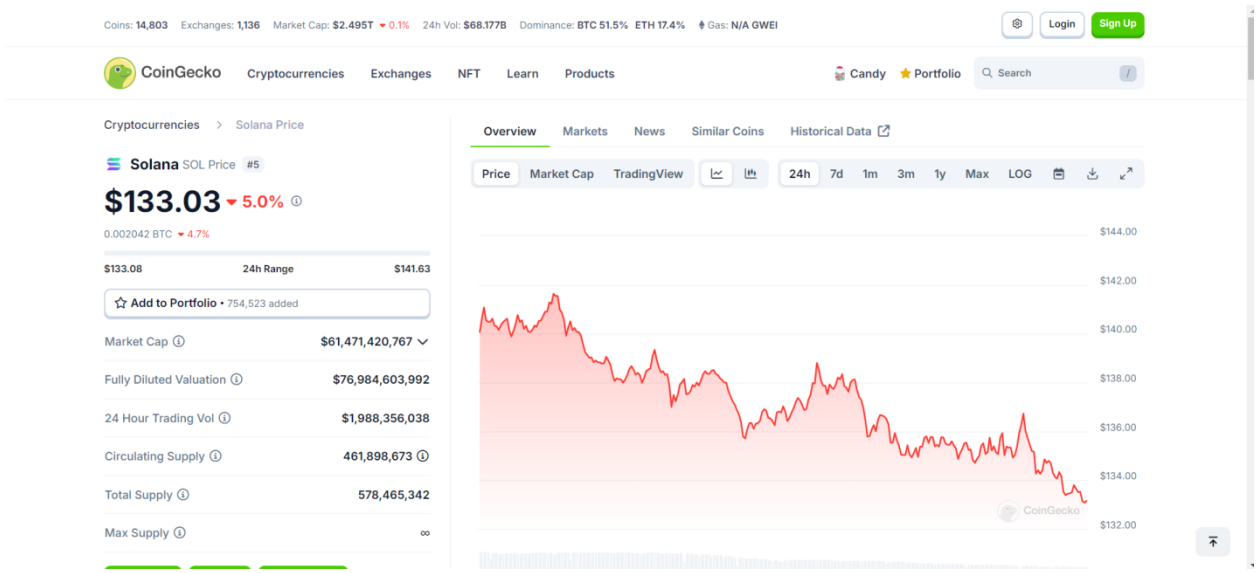

Рис. 1.3 — Інтерфейс CoinGecko

Переваги:

- Глибокий аналіз: CoinGecko надає широкий спектр аналітичних даних, включаючи оцінки розробницької активності, соціальної активності та ліквідності.
- Підтримка DeFi: Сервіс надає інформацію про децентралізовані фінанси (DeFi), що робить його корисним для користувачів, зацікавлених у цьому сегменті ринку.
- Регулярні оновлення: CoinGecko постійно оновлює свої дані, забезпечуючи користувачів актуальною інформацією.

Недоліки:

- Складніший інтерфейс: Інтерфейс CoinGecko може здатися складнішим для новачків порівняно з CoinMarketCap.
- Обмежені можливості для технічного аналізу: Подібно до CoinMarketCap, CoinGecko має обмежені інструменти для технічного аналізу.

1.2.3 TradingView

TradingView [8] є професійною платформою для технічного аналізу, яка пропонує інструменти для аналізу фінансових ринків, включаючи криптовалюти. TradingView дозволяє користувачам створювати власні

графіки, використовувати різноманітні індикатори та ділитися своїми ідеями з іншими трейдерами.

Рис. 1.4 — Інтерфейс TradingView

Переваги:

- Інструменти технічного аналізу: TradingView надає великий набір індикаторів, інструментів для малювання та налаштувань графіків, що робить його ідеальним для детального технічного аналізу.
- Спільнота трейдерів: Платформа має активну спільноту трейдерів, які діляться своїми ідеями та аналізами, що може бути корисним джерелом інформації.
- Можливість налаштування: Користувачі можуть налаштовувати інтерфейс під свої потреби та зберігати налаштування графіків для майбутнього використання.

Недоліки:

- Складність використання для новачків: Інтерфейс TradingView може здатися складним для новачків через велику кількість функцій та налаштувань.
- Висока вартість підписки: Деякі розширені функції доступні лише за платною підпискою, що може бути недосяжним для деяких користувачів.

1.2.4 CryptoCompare

CryptoCompare [9] є платформою, що надає дані про криптовалюти, включаючи ціни, обсяги торгів, ринкову капіталізацію та інші показники. CryptoCompare також пропонує новини, аналітику та рейтинги криптовалют.

Рис. 1.5 — Інтерфейс CryptoCompare

Переваги:

- Широкий спектр даних: CryptoCompare надає велику кількість даних про різні аспекти ринку криптовалют, що робить його корисним для комплексного аналізу.
- Інтеграція з новинами: Сервіс включає новини та аналітику, що дозволяє користувачам залишатися в курсі останніх подій на ринку.
- Зручний інтерфейс: Інтерфейс користувача є зручним та інтуїтивно зрозумілим, що полегшує роботу з даними.

Недоліки:

- Обмежені інструменти технічного аналізу: CryptoCompare пропонує обмежені можливості для технічного аналізу, що може бути недостатнім для деяких трейдерів.
- Залежність від сторонніх джерел: Дані збираються з різних бірж, що може призводити до неточностей або затримок в оновленні інформації.

Кожне з цих програмних рішень має свої сильні та слабкі сторони, що можуть впливати на вибір користувачів залежно від їх потреб та рівня досвіду. CoinMarketCap та CoinGecko є відмінними інструментами для загального огляду ринку та базового аналізу. TradingView пропонує професійні інструменти для технічного аналізу, але може бути складним для новачків. CryptoCompare забезпечує широкий спектр даних та новин, але обмежений в інструментах технічного аналізу. Однак більшість з цих рішень мають обмеження у вигляді дороговартісних підписок або складних інтерфейсів, що робить їх недоступними для широкого кола користувачів

Інтеграція програмного забезпечення з Telegram ботом є перспективним напрямом, оскільки вона дозволяє користувачам отримувати актуальну інформацію та проводити аналіз безпосередньо у зручному для них месенджері. Це забезпечує високу мобільність та доступність сервісу, що є важливим фактором у сучасному швидкоплинному світі [24].

Отже, на основі огляду предметної галузі та аналізу літератури існуючих рішень можна зробити висновок про необхідність створення інтегрованого програмного забезпечення, яке об'єднує найкращі практики технічного та фундаментального аналізу.

РОЗДІЛ 2. МЕТОДИ ТА АЛГОРИТМИ АНАЛІЗУ КРИПТОВАЛЮТ

Криптовалютний ринок є одним з найбільш динамічних і волатильних сегментів сучасної фінансової системи. Для ефективного управління ризиками та прийняття обґрунтованих інвестиційних рішень інвестори та трейдери активно використовують методи відстеження та аналізу, які базуються на різних теоретичних основах [10].

2.1 Методи аналізу криптовалют

Існує безліч методів аналізу криптовалютного ринку, які можна класифікувати за різними критеріями. Одним із основних поділів є на технічний та фундаментальний аналіз.

2.1.1 Технічний аналіз

Технічний аналіз [25] базується на обробці інформації, що стосується самого ринку і курсів криптовалют. Він включає аналіз історичних цінових даних, зміни цін на графіках та застосування різноманітних технічних індикаторів. Такі індикатори, як ковзні середні, індекс відносної сили (RSI), стохастичний осцилятор та інші, дозволяють оцінити моменти перегріву чи розпроданності ринку, а також визначити загальну тенденцію і можливі майбутні зміни цін [11].

2.1.2 Фундаментальний аналіз

Фундаментальний аналіз, натомість, зосереджується на зовнішніх чинниках, які можуть вплинути на цінову динаміку криптовалют. Сюди входять новини, що стосуються регуляторних змін у сфері криптовалют, економічні показники країн і глобальні тренди відносно цифрових активів. Фундаментальний аналіз допомагає інвесторам та трейдерам зрозуміти фундаментальну справедливу вартість криптовалют та прогнозувати їхні майбутні перспективи на основі фундаментальних факторів [12].

2.1.3 Сентиментальний аналіз

Сентиментальний аналіз оцінює настрої та психологічний стан учасників ринку щодо криптовалют. Він вивчає суспільні настрої, ставлення

масових медіа та соціальних мереж до конкретних криптовалют, а також зміни у відчуттях ринкових учасників, що можуть впливати на цінові тенденції.

У сучасному світі інтеграція цих різних методів аналізу є ключовим аспектом успішної торгівлі на криптовалютних ринках. Використання комплексного підходу дозволяє знижувати ризики і приймати більш обґрунтовані інвестиційні рішення в умовах високої волатильності цифрових активів [13].

2.2 Математичні та статистичні методи аналізу даних

Математичні та статистичні методи є невід'ємною частиною комплексного аналізу даних на криптовалютному ринку. Вони дозволяють систематично вивчати та аналізувати історичні дані цінових рухів криптовалют для виявлення ключових патернів, трендів та залежностей, що є критичними для прийняття інвестиційних рішень [14]. Серед основних математичних та статистичних методів використовуються:

- Кореляційний аналіз

Кореляційний аналіз використовується для встановлення статистичних зв'язків між цінами різних криптовалют або між цінами криптовалют і іншими фінансовими активами, такими як фондові ринки. Цей метод дозволяє розуміти, які активи можуть реагувати схожим чином на зміни у загальному фінансовому середовищі [15]. Наприклад, виявлення високої кореляції між криптовалютами і певними фондовими індексами може свідчити про тісну взаємодію між цими ринками у контексті глобальних фінансових трендів.

- Регресійний аналіз

Регресійний аналіз використовується для прогнозування майбутніх цінових рухів на основі історичних даних. Цей метод дозволяє побудувати математичні моделі, які враховують залежність між ціною криптовалюти та іншими факторами, такими як часові ряди, обсяги торгів, новини та інші економічні показники. Регресійні моделі допомагають інвесторам та

аналітикам розуміти, які чинники впливають на цінову динаміку і як ці залежності можуть проявлятися у майбутньому [16].

- Кластерний аналіз

Кластерний аналіз використовується для групування схожих криптовалют за різними параметрами, такими як структура цін, торговий обсяг, технічні показники та інші характеристики. Цей метод дозволяє ідентифікувати групи активів з схожими властивостями і поведінкою на ринку. Групування криптовалют в класи дозволяє аналітикам та трейдерам розуміти, які категорії активів можуть мати подібні цінові та торгові характеристики, що є важливим для побудови стратегій торгівлі та управління портфелем [17].

Ці методи в сукупності допомагають не лише аналізувати минулі події на ринку криптовалют, а й прогнозувати майбутні цінові траєкторії та виявляти можливості для отримання прибутку в умовах високої волатильності і невизначеності.

2.3 Методи прогнозування курсу криптовалют

Прогнозування цінових рухів криптовалют викликає великий інтерес через високу волатильність ринку та значення швидких реакцій на зовнішні події. Для досягнення цієї мети використовуються різноманітні методи, які враховують як історичні дані, так і сучасні тренди на ринку. Основні підходи включають:

- Часові ряди

Аналіз історичної поведінки цін є основою для побудови моделей прогнозування на основі часових рядів. Цей метод полягає в детальному розгляді попередніх цінових даних і побудові прогностичних моделей, що базуються на патернах і трендах, виявлених в історичних даних. Аналіз часових рядів дозволяє ідентифікувати циклічність, сезонність та інші регулярні закономірності[22], що можуть використовуватися для прогнозування майбутніх цінових рухів.

- Моделі машинного навчання

Застосування моделей машинного навчання, таких як нейронні мережі, дерева рішень, метод опорних векторів та інші, стало популярним напрямком в прогнозуванні цінових траєкторій криптовалют. Ці алгоритми можуть ефективно аналізувати великі обсяги даних і виявляти складні залежності між різними факторами, які впливають на ринок. Моделі машинного навчання використовуються для створення прогностичних моделей, які можуть адаптуватися до змінюючихся умов ринку та приймати швидкі рішення з урахуванням нових даних [18].

- Аналіз сентименту

Врахування громадської думки, настроїв ринку і реакції на новини та події є ще однією важливою складовою прогнозування цінових змін криптовалют. Аналіз сентименту базується на обробці текстової інформації з соціальних мереж, форумів та новинних порталів для визначення настроїв учасників ринку. Позитивні або негативні новини можуть впливати на інвестиційні рішення та цінові рухи, тому аналіз сентименту дозволяє урахувати психологічний аспект в оцінці майбутніх трендів на ринку криптовалют [19].

Ці методи взаємодіють між собою, створюючи комплексні моделі прогнозування, які допомагають інвесторам та трейдерам приймати обґрунтовані рішення в умовах непередбачуваності ринку криптовалют.

2.4 Роль API в криптовалютних біржах

API (Application Programming Interface) — це набір інструкцій і структур даних, які визначають способи взаємодії одного комп'ютерного програмного забезпечення з іншим. API дозволяє різним програмам спілкуватися між собою [20]. Основна ідея полягає в тому, щоб інші програми (або розробники) могли використовувати певні функції або отримувати доступ до певних даних без необхідності розкриття всіх внутрішніх деталей цієї програми чи системи.

API криптовалютних бірж є невід'ємною складовою для отримання цінових даних, здійснення торгівельних операцій та інтеграції з різноманітними програмними рішеннями. Важливі аспекти використання API включають:

- Автентифікація і безпека

Автентифікація є першочерговим завданням для забезпечення безпеки під час взаємодії з API криптовалютних бірж. Це включає використання ключів доступу, схем шифрування та протоколів безпеки для забезпечення конфіденційності та цілісності даних. [23] Автентифікаційні механізми дозволяють інвесторам та трейдерам захищати свої аккаунти та уникати несанкціонованого доступу до фінансових ресурсів.

- Отримання даних через API

Для отримання реального часу цінових оновлень, обсягів торгів та іншої важливої інформації інвестори використовують REST або WebSocket API. REST API надає можливість здійснювати запити на отримання даних, які включають котирування, історичні ціни і інші ринкові параметри [20]. WebSocket API забезпечує миттєву передачу живих даних, що робить його ідеальним для реального часу аналізу ринку та моніторингу.

- Виконання торгових операцій

API криптобірж дозволяють здійснювати торгівельні операції безпосередньо через програмні інтерфейси. Це включає в себе можливість робити покупки, продажі, розміщувати та відміняти замовлення, управляти портфелем активів і виконувати інші операції без необхідності входити в особистий кабінет біржі через веб-інтерфейс.

Застосування цих теоретичних основ дозволяє інвесторам та трейдерам ефективно аналізувати ринок криптовалют, приймати обґрунтовані рішення та мінімізувати ризики у високоволатильному середовищі цифрових активів. Використання API значно спрощує процес отримання та аналізу даних, що є критичним для успішної торгівлі та управління криптовалютними активами.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

За допомогою аналізу існуючих рішень та методів аналізу курсу криптовалют на криптовалютних біржах, було визначено вимоги до програмного забезпечення та було розроблено архітектуру системи, в основі якої лежить клієнт-серверна частина [21] та алгоритм для аналізу, який складається з модулів, які взаємодіють між собою .

Загальні вимоги ПЗ включають:

- Функціональні вимоги: реалізація алгоритмів відстеження та аналізу цінових даних криптовалют, прогнозування цінових рухів, візуалізація статистики та трендів.
- Надійність: забезпечення стабільної роботи системи при високих навантаженнях та у високоволатильному середовищі ринку криптовалют.
- Безпека: захист конфіденційності та цілісності даних користувачів, забезпечення безпеки під час взаємодії з API криптовалютних бірж.
- Інтеграція: можливість взаємодії з Telegram ботом для зручного доступу до основних функцій системи на різних платформах.

3.1. Архітектура алгоритму

Алгоритм буде побудований з використанням клієнт-серверної архітектури. Клієнтська частина буде представлена Telegram ботом, що забезпечить зручний інтерфейс для користувачів, та мультиплатформність у взаємодії з системою. Серверна частина буде відповідальною за збір та обробку даних, а також взаємодію з зовнішніми API для отримання цінової інформації.

Модулі архітектури:

- **Модуль збору даних.**

Модуль збору даних відповідає за збір інформації з різних джерел, включаючи криптовалютні біржі, ресурси новин та соціальні мережі.

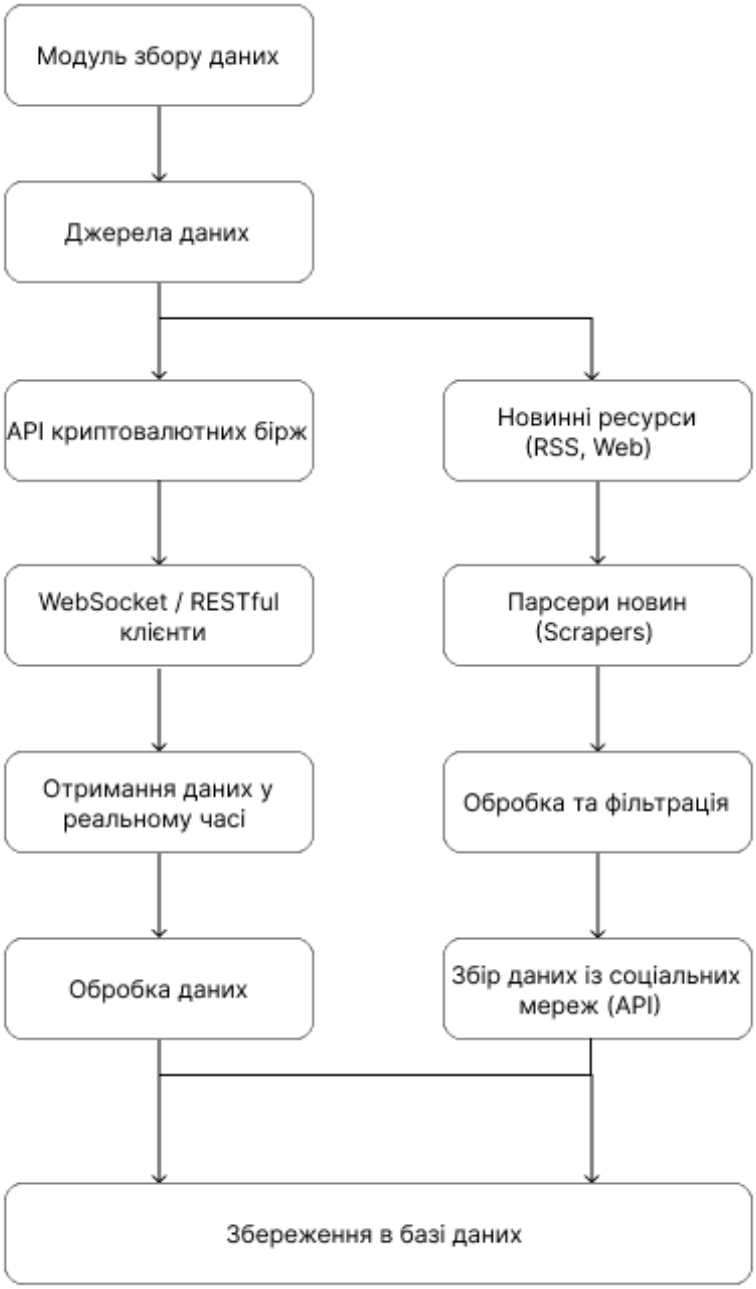

Рис. 3.1 — Модуль збору даних

Опис компонентів:

1. Модуль збору даних: центральний компонент, відповідальний за координацію збору даних з різних джерел.
2. Джерела даних: включають API криптовалютних бірж, ресурси новин, соціальні мережі тощо.
3. API криптовалютних бірж: використовуються для отримання даних про ціни, обсяги торгів тощо в реальному часі або пакетним способом.

4. Ресурси новин: містять актуальні новини, що можуть впливати на ринок криптовалют.
5. WebSocket / RESTful клієнти: інструменти для підключення до бірж та отримання даних.
6. Парсери новин: інструменти для збору та обробки новин з різних веб-ресурсів.
7. Обробка даних: процес очищення та нормалізації даних для подальшого аналізу.
8. Збір даних із соціальних мереж: використання API для отримання даних із соціальних мереж.
9. Збереження в базу даних: зберігання оброблених даних у базу даних для подальшого аналізу та використання іншими модулями системи.

- **Модуль обробки даних**

Забезпечує обробку та аналіз зібраних даних, включаючи визначення трендів, прогнозування цін та виявлення аномалій.

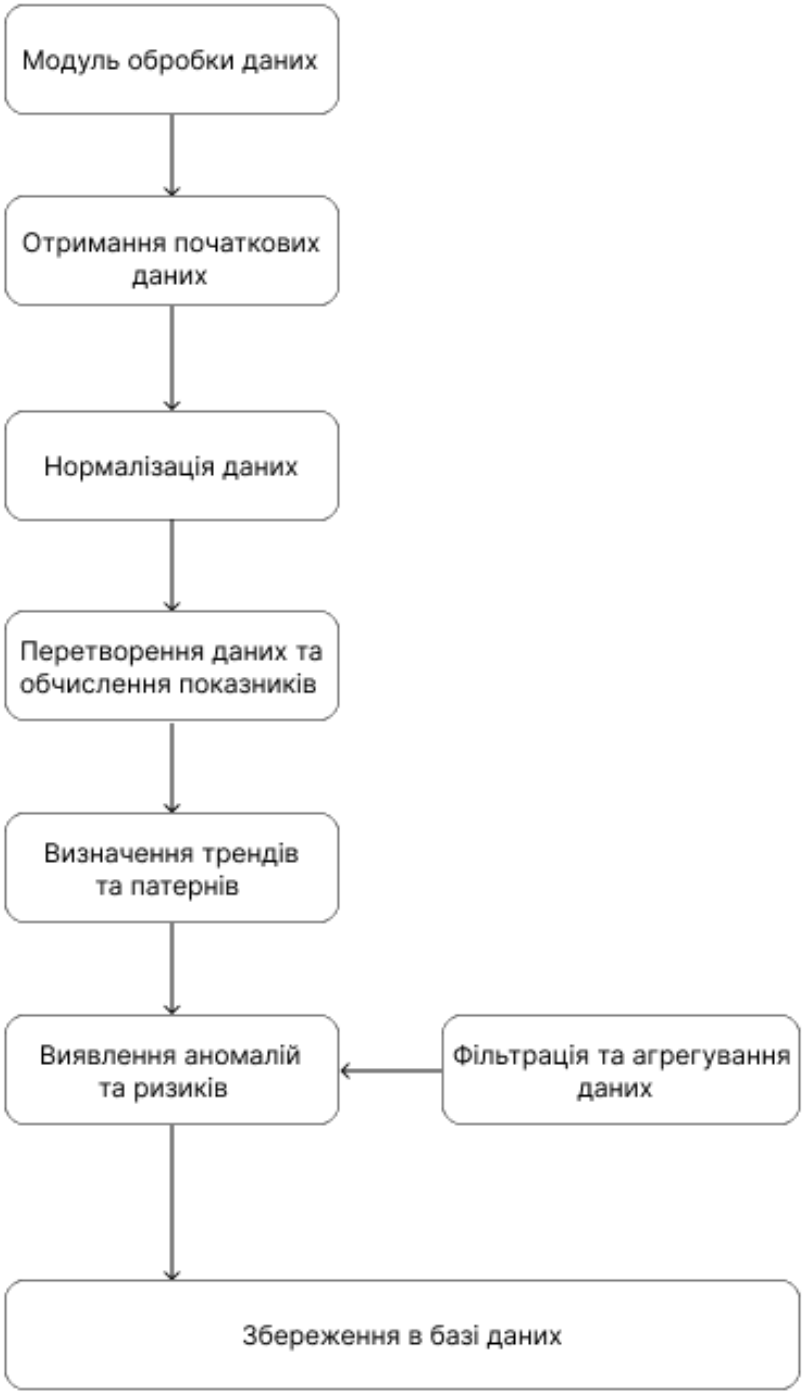

Рис. 3.2 — Модуль обробки даних

Опис компонентів:

1. Модуль обробки даних: центральний компонент, відповідальний за обробку та аналіз зібраних даних.
2. Отримання сирих даних: збір даних з різних джерел, включаючи API криптовалютних бірж, ресурси новин та соціальні мережі.

3. Очищення та нормалізація даних: процес видалення шуму, пропущених значень, дублікованих записів та нормалізації даних для забезпечення їхньої узгодженості.
4. Перетворення даних та обчислення показників: обчислення додаткових метрик, таких як середні значення, відхилення, індекси та інші показники для аналізу.
5. Визначення трендів та патернів: вявлення трендів, повторюваних шаблонів та аномальних подій у даних.
6. Виявлення аномалій та ризиків: ідентифікація незвичайних змін у даних, які можуть вказувати на потенційні ризики або можливості.
7. Фільтрація та агрегування даних: відбір важливих даних та їх агрегування для спрощення подальшого аналізу.
8. Збереження оброблених даних в базі даних: збереження оброблених і проаналізованих даних у базу даних для доступу іншими модулями системи та для подальшого аналізу.

- **База даних**

Використовується для зберігання даних про криптовалютний ринок, включаючи історію змін цін, обсяги торгів, новини та соціальні сигнали.

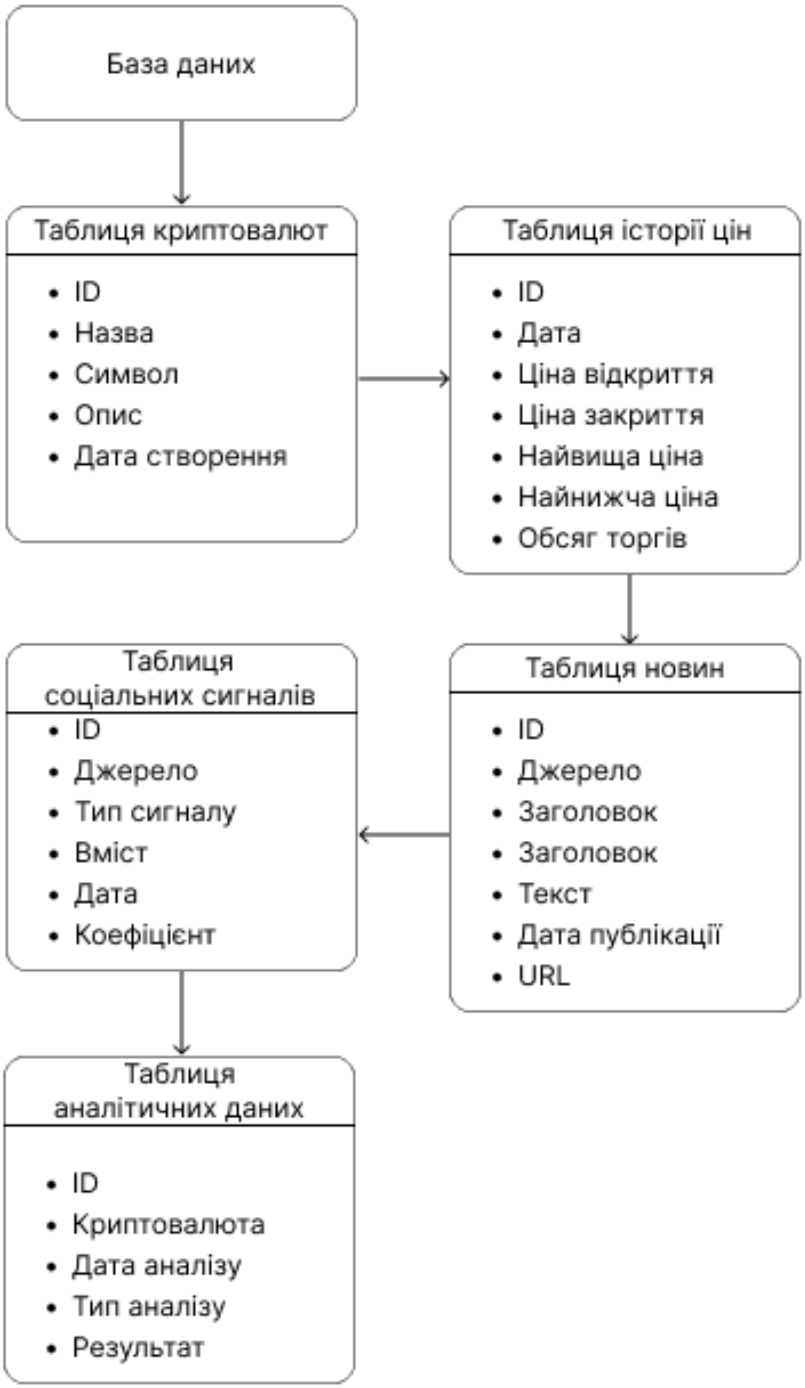

Рис. 3.3 — Модуль база даних

Опис компонентів:

1. Таблиця криптовалют: містить основну інформацію про криптовалюти, включаючи їхню назву, символ та опис.
2. Таблиця історії цін: зберігає історичні дані про ціни криптовалют, включаючи ціну відкриття, закриття, найвищу та найнижчу ціну, а також обсяг торгів за кожен день.

3. Таблиця новин: містить дані про новини, які можуть впливати на ринок криптовалют. Включає джерело новини, заголовок, текст, дату публікації та URL.
4. Таблиця соціальних сигналів: зберігає дані з соціальних мереж, такі як твіти або коментарі, що можуть впливати на ринок. Включає джерело, тип сигналу, вміст, дату та аналіз настроїв (sentiment score).
5. Таблиця аналітичних даних: містить результати аналітичних обчислень, таких як виявлені тренди, аномалії та інші типи аналізу, разом із відповідними датами.

- **Компонент отримання даних**

Інтеграція з API криптовалютних бірж для отримання реального часу цінових даних та інших параметрів.

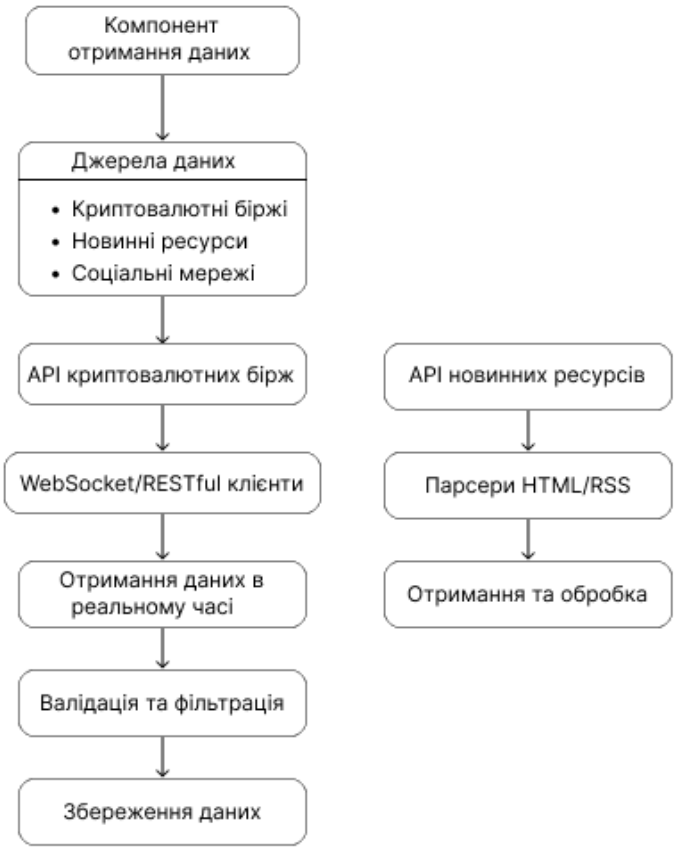

Рис. 3.4 — Компонент отримання даних

Опис компонентів:

1. Компонент отримання даних: центральний модуль, що відповідає за збір даних з різних джерел.

2. Джерела даних: основні джерела інформації, такі як криптовалютні біржі, ресурси новин та соціальні мережі.
3. API криптовалютних бірж: інтерфейси для отримання даних про ціни та обсяги торгів з криптовалютних бірж.
4. API ресурсів новин: інтерфейси для збору новин, що можуть впливати на ринок криптовалют.
5. WebSocket / RESTful клієнти: використовуються для підключення до бірж і отримання даних у реальному часі або через періодичні запити.
6. Парсери HTML / RSS:інструменти для збору новин з веб-ресурсів, таких як HTML-парсери або парсери RSS.
7. Отримання даних в реальному часі / пакетний збір: збір даних як у режимі реального часу, так і у вигляді пакетів для подальшої обробки.
8. Отримання даних із соціальних мереж: використання API для отримання даних із соціальних мереж.
9. Валідація та фільтрація: процес перевірки та очищення даних для забезпечення їхньої коректності та узгодженості.
10. Збереження сирих даних у базу даних: збереження зібраних даних у базу даних для подальшої обробки та аналізу.

- **Аналітичний модуль**

Модулі для математичного та статистичного аналізу даних, включаючи методи прогнозування та алгоритми обробки великих обсягів даних.

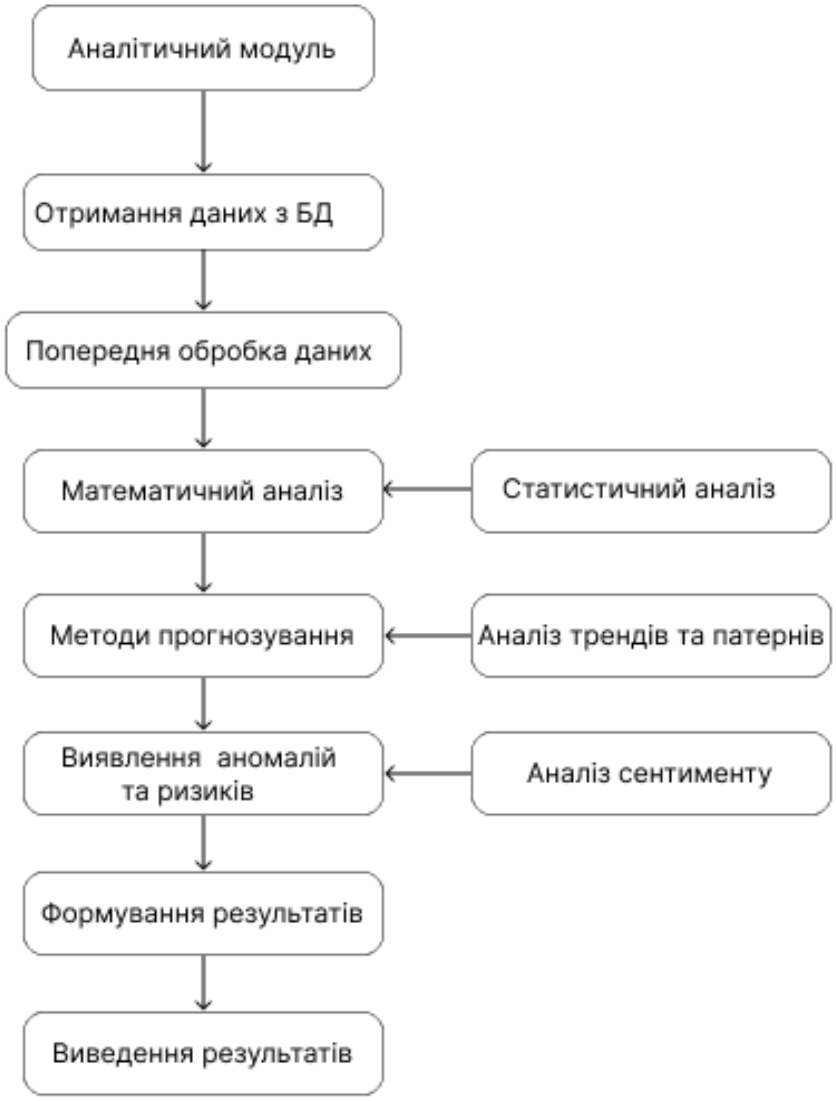

Рис. 3.5 — Аналітичний модуль

Опис компонентів:

1. Аналітичний модуль: центральний компонент для виконання аналітичних завдань.
2. Отримання даних з БД: отримання оброблених даних із бази даних для подальшого аналізу.
3. Попередня обробка даних: очищення та нормалізація даних перед аналізом.
4. Математичний аналіз: виконання регресійного, кореляційного аналізу та інших математичних методів.
5. Статистичний аналіз: виконання аналізу розподілу даних, тестування гіпотез та інших статистичних методів.

6. Методи прогнозування: використання моделей часових рядів, машинного навчання для прогнозування майбутніх значень.
7. Аналіз трендів та патернів: виявлення трендів та патернів у даних для кращого розуміння ринкової динаміки.
8. Виявлення аномалій та ризиків: виявлення незвичайних змін або ризиків на ринку.
9. Аналіз сентименту: оцінка настроїв на ринку на основі соціальних сигналів та новин.
10. Формування результатів та рекомендацій: Підготовка висновків та рекомендацій на основі проведеного аналізу.
11. Виведення результатів: Візуалізація даних та формування звітів для користувачів.

- **Модуль візуалізації**

Інтерфейс для відображення графіків, діаграм та інших візуальних елементів для зручного аналізу інформації.

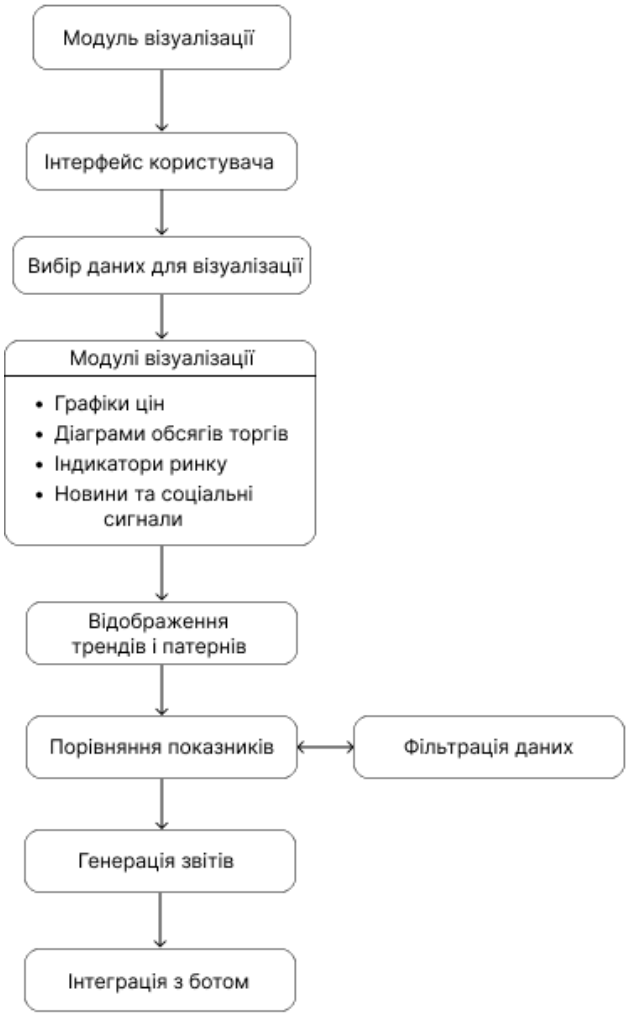

Рис. 3.6 — Модуль візуалізації даних

Опис компонентів:

1. Модуль візуалізації: відповідає за створення і відображення графіків, діаграм та інших візуальних елементів для користувачів.
2. Інтерфейс користувача: графічний інтерфейс для взаємодії користувача з модулем візуалізації.
3. Вибір даних для візуалізації: можливість вибору користувачем, які дані потрібно візуалізувати.
4. Модулі візуалізації:
5. Діаграми обсягів торгів: візуалізація обсягів торгів за певні періоди.
6. Індикатори ринку: Відображення технічних індикаторів, таких як ковзні середні, RSI, MACD тощо.
7. Новини та соціальні сигнали: відображення важливих новин та соціальних сигналів, які можуть вплинути на ринок.

8. Відображення трендів та патернів: візуалізація виявлених трендів і патернів у даних.
9. Порівняння різних криптовалют та показників: можливість порівняння кількох криптовалют або різних показників на одному графіку.
10. Фільтрація даних для візуалізації: відбір важливих даних для візуалізації на основі певних критеріїв.
11. Генерація звітів та експорт графіків: створення звітів у різних форматах (PDF, Excel тощо) та можливість експорту графіків.
12. Інтеграція з Telegram ботом: відправка графіків та звітів користувачам через Telegram бот для зручного доступу до інформації.

3.2. Вибір технологій та інструментів

Для розробки програмного забезпечення будуть використані передові технології та інструменти, що відповідають сучасним вимогам індустрії криптовалют та фінансових технологій

3.2.1 Python

Python – це універсальна інтерпретована мова програмування загального призначення. Мова стала популярною завдяки своїй простоті, читабельності та великій кількості бібліотек, що дозволяють швидко вирішувати різноманітні завдання [26].

Основні характеристики Python

- Синтаксис: Python має чистий і зрозумілий синтаксис, який легко читати та писати. Відсутність складних структур робить код доступним навіть для новачків.
- Інтерпретація: Python є інтерпретованою мовою, що означає, що код виконується рядок за рядком, що полегшує тестування та налагодження програм.
- Мультиплатформенність: Python підтримується на різних операційних системах, включаючи Windows, macOS, Linux та інші. Це дозволяє розробникам писати код, який може працювати на будь-якій з цих платформ без необхідності вносити зміни.

3.2.1.1 Бібліотека Pandas

Pandas – це аналітична бібліотека для мови програмування Python, яка використовується для обробки та аналізу даних [26]. Вона була створена Весом МакКінні у 2008 році і з тих пір стала одним із найпопулярніших інструментів для роботи з даними в наукових дослідженнях, фінансах, машинному навчанні та інших галузях.

Основні характеристики Pandas

- DataFrame: Основний об'єкт у Pandas, двовимірна таблиця з гнучкою індексацією рядків і стовпців. Це схоже на таблицю в базі даних або електронній таблиці, що дозволяє легко маніпулювати даними.
- Series: Одновимірний масив, подібний до списку або масиву, але з індексами, що дозволяє легко звертатися до даних.

Інструменти для маніпуляції даними:

- Фільтрація та сортування: Легко фільтрувати дані за умовами та сортувати їх за різними критеріями.
- Агрегація та групування: Виконання складних операцій агрегації та групування для отримання корисної інформації з великих наборів даних.
- Об'єднання та злиття: Інструменти для об'єднання різних наборів даних, включаючи злиття на основі ключів (merge), конкатенацію (concat) та інші операції.

Обробка відсутніх даних:

Pandas пропонує інструменти для виявлення, заповнення та видалення відсутніх значень, що є критично важливим для забезпечення якості даних.

- Підтримка різноманітних форматів файлів для імпорту та експорту даних, включаючи CSV, Excel, SQL, HDF5 та інші. Це дозволяє легко інтегрувати Pandas у різні робочі процеси.
- Операції з часовими рядами:

- Pandas має розширені можливості для роботи з часовими рядами, включаючи обробку дат і часу, створення часових інтервалів, об'єднання та індексацію часових рядів.

3.2.1.2 Бібліотека Matplotlib

Matplotlib – це графічна бібліотека для візуалізації даних у мові програмування Python [26]. Вона була розроблена Джоном Д. Хантером у 2003 році з метою створення статичних, анімованих та інтерактивних графіків. Matplotlib стала стандартним інструментом для створення візуалізацій у наукових дослідженнях, інженерії, фінансах та інших галузях.

Основні характеристики Matplotlib

1. Широкий спектр графічних можливостей:
    - Лінійні графіки: Побудова простих і багатосерійних лінійних графіків для відображення змін параметрів у часі або по іншій змінній.
    - Стовпчикові діаграми: Відображення категоріальних даних у вигляді стовпчиків різної висоти.
    - Гістограми: Візуалізація розподілу числових даних у вигляді стовпчиків, що показують частоту значень у інтервалах.
    - Кругові діаграми: Відображення часток категоріальних даних у вигляді секторів кола.
    - Розсіювальні діаграми: Відображення взаємозв'язків між двома змінними у вигляді точок на координатній площині.
2. Анімовані графіки:
    - Matplotlib дозволяє створювати анімовані графіки, що можуть змінюватися з часом, що є корисним для демонстрації динаміки даних або процесів.
3. Інтерактивність:

- Інтеграція з інтерактивними середовищами, такими як Jupyter Notebook, дозволяє користувачам взаємодіяти з графіками, змінюючи параметри в реальному часі.

4. Налаштування та стилізація:
    - Matplotlib надає великий вибір інструментів для налаштування графіків, включаючи зміну кольорів, маркерів, ліній, шрифтів та інших елементів.
    - Підтримка тем оформлення для швидкої зміни зовнішнього вигляду графіків.
5. Підтримка різних форматів файлів:
    - Графіки можна зберігати у різних форматах, включаючи PNG, PDF, SVG та інші. Це дозволяє легко інтегрувати візуалізації у публікації, презентації та звіти.

Matplotlib є професійною та гнучкою бібліотекою для візуалізації даних у Python. Вона надає широкий спектр можливостей для створення різноманітних типів графіків та діаграм, що дозволяє ефективно аналізувати та представляти дані. Завдяки своїй простоті використання та багатим можливостям налаштування Matplotlib стала стандартом де-факто для візуалізації даних у наукових дослідженнях, інженерії, фінансах та інших галузях.

3.3 База даних SQLite

Для збереження даних у програмному забезпеченні для відстеження та аналізу криптовалюти, їх ціни та новин, використовується база даних SQLite.

SQLite — це легка реляційна база даних, яка є популярним вибором для багатьох додатків завдяки своїй простоті у використанні, відсутності необхідності в налаштуванні сервера та чудовій продуктивності. SQLite відрізняється від інших систем керування базами даних (СКБД) тим, що вона інтегрується безпосередньо в програму і зберігає всі дані в одному файлі.

Основні характеристики SQLite:

- Легка вага: SQLite дуже компактна і вбудовується безпосередньо в додаток, що робить її ідеальною для мобільних додатків, веб-додатків і невеликих десктопних програм.
- Самодостатність: Для роботи SQLite не потрібен окремий сервер або процес, що значно спрощує її використання і налаштування.
- Повна підтримка SQL: SQLite підтримує більшість стандартних SQL-запитів, включаючи SELECT, INSERT, UPDATE, DELETE, JOIN та інші.
- Переносимість: Дані зберігаються в одному файлі, який можна легко переміщати між різними платформами і середовищами.
- Продуктивність: Хоча SQLite не призначена для високонавантажених систем, вона забезпечує достатню продуктивність для більшості звичайних додатків.

3.4 Інтеграція з Telegram ботом

Інтеграція з Telegram ботом є ключовою складовою програмного забезпечення для відстеження та аналізу криптовалютного ринку, оскільки вона забезпечує користувачам зручний і мультиплатформенний спосіб взаємодії з системою. Завдяки цій інтеграції користувачі зможуть отримувати актуальну інформацію про ціни на криптовалюти та результати ринкового аналізу безпосередньо в месенджері Telegram, що значно підвищує зручність та ефективність використання системи.

Функціональні можливості Telegram бота

1. Отримання поточних цін на криптовалюти

- Користувачі можуть запитувати поточні ціни на різні криптовалюти, наприклад, Bitcoin, Ethereum та інші. Бот, у відповідь на запит, надсилатиме актуальні дані про ціни.

2. Історичний аналіз та графіки

- Бот може надавати користувачам історичні дані у вигляді графіків, що ілюструють зміни цін на криптовалюти за певні періоди. Це дозволяє

користувачам здійснювати технічний аналіз ринку безпосередньо в Telegram.

3. Сповіщення про зміни цін

- Користувачі можуть налаштувати сповіщення, які будуть надсилатися ботом у разі досягнення криптовалютою певних цінових рівнів. Це допоможе користувачам швидко реагувати на важливі ринкові події.

4. Аналітичні звіти

- Бот може надсилати регулярні аналітичні звіти, що містять важливу інформацію про ринкові тенденції, прогнози та рекомендації. Звіти можуть бути налаштовані на щоденну, тижневу або місячну розсилку.

Підключення системи

1. Вибір бібліотеки для розробки бота

- Для створення Telegram бота буде використана бібліотека pyTelegramBotAPI, яка забезпечує простий та інтуїтивний інтерфейс для взаємодії з API Telegram.

3. Архітектура системи

- Telegram бот буде інтегрований з серверною частиною системи, що написана на Python. Бот буде обробляти запити користувачів і передавати їх на сервер, де виконуватиметься основна логіка обробки даних.
- Сервер буде відповідати за взаємодію з зовнішніми API для отримання актуальних даних про ціни на криптовалюти, обробку цих даних та їх збереження у базі даних.

4. Взаємодія з користувачем

- Telegram бот буде використовувати командний інтерфейс для обробки запитів користувачів. Наприклад, користувачі зможуть вводити команди /price BTC для отримання поточної ціни на Bitcoin або /chart ETH 7d для отримання графіка зміни ціни на Ethereum за останні 7 днів.

5. Обробка та зберігання даних

- Дані про запити користувачів та результати обробки будуть зберігатися у базі даних, що дозволить забезпечити історичність та контекстність інформації. База даних буде оптимізована для швидкого доступу та обробки великих обсягів даних.

Інтеграція з Telegram ботом надасть користувачам зручний та ефективний інструмент для відстеження та аналізу криптовалютного ринку. Завдяки використанню сучасних технологій та інструментів, бот забезпечить інтерактивний інтерфейс, що дозволить отримувати актуальну інформацію, аналітичні звіти та налаштовувати сповіщення в реальному часі. Це значно підвищить зручність та ефективність використання системи для кінцевих користувачів.

РОЗДІЛ 4. АПРБАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Апробація програмного забезпечення є дуже важливим етапом у процесі розробки, що дозволяє оцінити працездатність та ефективність створеної системи. Вона включає в себе тестування функціональності, виявлення та усунення помилок, а також збір зворотного зв'язку від кінцевих користувачів для подальшого вдосконалення продукту.

Основною метою апробації є перевірка відповідності програмного забезпечення встановленим вимогам та очікуванням користувачів. Це включає перевірку коректності обробки даних, стабільності роботи системи, зручності користувацького інтерфейсу та відповідності функціональності заявленим можливостям.

Процес апробації програмного забезпечення для відстеження та аналізу криптовалютного ринку можна розділити на кілька ключових етапів:

1. Тестування функціональності

На цьому етапі проводиться тестування всіх функціональних можливостей системи, включаючи збір та обробку даних, генерацію звітів, інтеграцію з Telegram ботом та інші основні функції. Це дозволяє виявити та виправити будь-які помилки чи неточності в роботі системи.

Рис. 4.1 — Новини

2. Тестування продуктивності

Перевіряється здатність системи обробляти великі обсяги даних та виконувати запити користувачів у режимі реального часу. Це включає навантажувальне тестування для оцінки стабільності та швидкодії системи під високим навантаженням.

Рис. 4.2 — Торгова пара

3. Тестування зручності користування

Апробація програмного забезпечення продемонструвала, що система для відстеження та аналізу криптовалютного ринку відповідає встановленим вимогам та готова до впровадження у реальних умовах. Тестування підтвердило її здатність забезпечувати стабільну роботу, швидку обробку запитів та надання користувачам актуальної та корисної інформації у зручному форматі. Отримані результати підтверджують, що створене програмне забезпечення є надійним інструментом для відстеження та аналізу криптовалютного ринку.

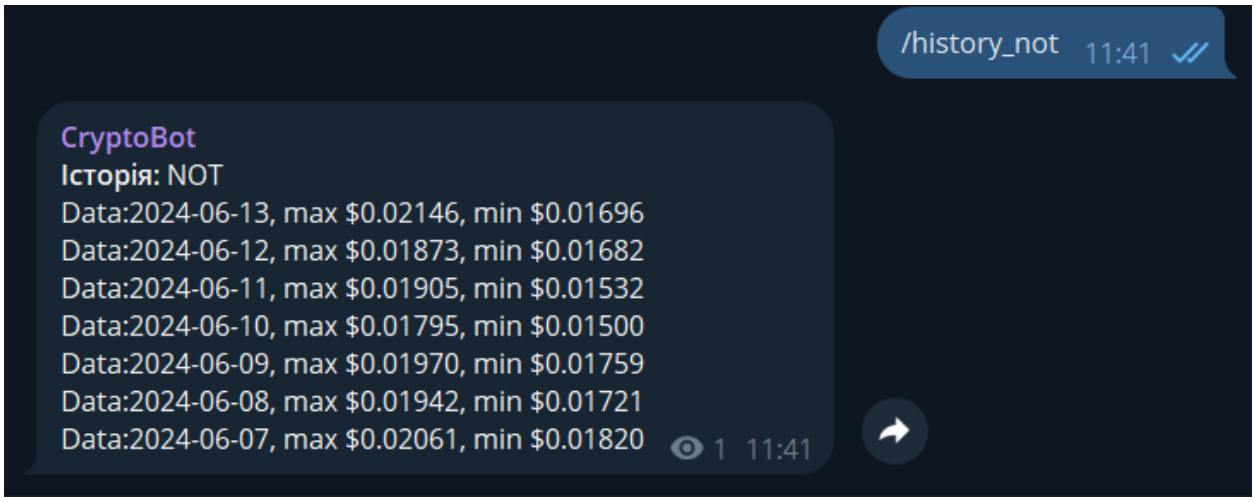

Рис. 4.3 — Історія цін

ВИСНОВОК

У кваліфікаційній роботі було розроблено програмне забезпечення для відстеження та аналізу криптовалютного ринку, що поєднує в собі різноманітні методи аналізу та прогнозування цінових рухів цифрових активів. Основними аспектами реалізації програми є використання математичних та статистичних методів для виявлення патернів у цінових рухах, моделі машинного навчання для прогнозування майбутніх значень цін, а також інтеграція з Telegram ботом для зручного доступу до аналітичної інформації.

Проєктування системи передбачало створення модульної архітектури з ефективними механізмами отримання та обробки цінових даних, використання сучасних технологій програмування та інструментів для реалізації функціональності системи. Була розроблена база даних для зберігання цінових та інших параметрів, що забезпечило надійне зберігання та швидкий доступ до інформації.

Інтеграція з Telegram ботом дозволила користувачам отримувати актуальні цінові оновлення та аналітичні звіти безпосередньо через популярний месенджер, що значно полегшило процес моніторингу ринку та прийняття інвестиційних рішень.

Отже, програмне забезпечення успішно відповідає вимогам сучасного інвестиційного ринку криптовалют, надаючи інструменти для аналізу, прогнозування та прийняття обґрунтованих інвестиційних рішень у високоволатильному середовищі цифрових активів.

СПИСОК ЛІТЕРАТУРИ

1. Mastering Bitcoin: Unlocking Digital Cryptocurrencies / Antonopoulos, A. M. – Sebastopol: O'Reilly Media, 2014. 45 – 67 с.
2. Cryptocurrency: How Bitcoin and Digital Money are Challenging the Global Economic Order / Vigna, P., Casey, M. J. – New York: St. Martin's Press, 2015. 85 – 102 с.
3. Blockchain Basics: A Non-Technical Introduction in 25 Steps / Drescher, D. – New York: Apress, 2017. 39 – 58 с.
4. Digital Gold: Bitcoin and the Inside Story of the Misfits and Millionaires Trying to Reinvent Money / Popper, N. – New York: Harper, 2015. 112 – 133 с.
5. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction / Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S. – Princeton: Princeton University Press, 2016. 67 – 89 с.
6. Site CoinMarketCap — URL: https://coinmarketcap.com/ua/
7. Site CoinGecko — URL: https://www.coingecko.com/ua
8. Site TradingView — URL: https://tradingview.com/
9. Site CryptoCompare — URL: https://www.cryptocompare.com/
10. The Age of Cryptocurrency: How Bitcoin and Digital Money are Challenging the Global Economic Order / Vigna, P., Casey, M. J. – New York: St. Martin's Press, 2016. 44 – 69 с.
11. Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World / Tapscott, D., Tapscott, A. – New York: Portfolio, 2016. 50 – 72 с.
12. Bitcoin: The Future of Money? / Beck, M. – Cambridge: Polity Press, 2015. 76 – 93 с.
13. The Business Blockchain: Promise, Practice, and the Application of the Next Internet Technology / Mougayar, W. – Hoboken: Wiley, 2016. 59 – 81 с.

14. Cryptoassets: The Innovative Investor's Guide to Bitcoin and Beyond / Burniske, C., Tatar, J. – New York: McGraw-Hill Education, 2017. 95 – 112 c.
15. Blockchain: Blueprint for a New Economy / Swan, M. – Sebastopol: O'Reilly Media, 2015. 68 – 91 c.
16. Mastering Blockchain: Deeper Insights into Decentralization, Cryptography, Bitcoin, and Popular Blockchain Frameworks / Bashir, I. – Birmingham: Packt Publishing, 2017. 100 – 123 c.
17. Ethereum: Building Blockchain Decentralized Apps / Dannen, C. – Berkeley: Apress, 2017. 45 – 65 c.
18. Decentralized Applications: Harnessing Bitcoin's Blockchain Technology / El Messiry, A. – Birmingham: Packt Publishing, 2016. 55 – 77 c.
19. Blockchain Applications: A Hands-On Approach / Watt, A. – New York: Springer, 2018. 120 – 141 c.
20. Algorithmic Trading and DMA: An Introduction to Direct Access Trading Strategies / Johnson, B. – London: 4Myeloma Press, 2010. 67 – 90 c.
21. Automated Trading with R: Quantitative Research and Platform Development / Thomas, C. – Berkeley: Apress, 2017. 90 – 113 c.
22. Hands-On Blockchain with Hyperledger: Building Decentralized Applications with Hyperledger Fabric and Composer / Thakkar, N., Ruan, X. – Birmingham: Packt Publishing, 2018. 88 – 111 c.
23. Bitcoin for the Befuddled / Pritzker, J., Onofrio, S. – San Francisco: No Starch Press, 2014. 105 – 128 c.
24. Cryptocurrency Mining For Dummies / Holmes, P. – Hoboken: Wiley, 2018. 45 – 67 c.
25. Crypto: How the Code Rebels Beat the Government—Saving Privacy in the Digital Age / Levy, S. – New York: Penguin Books, 2001. 89 – 110 c.
26. Python Site — URL: https://www.python.org/

ДОДАТОК А

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
class Window:


    def __init__(self):
        root3 = tk.Toplevel(root)
        root3.title(username1 +"'s account")
        root3.geometry("700x600+500+50")position of root3 and its size.
        self.canvas = tk.Canvas(root3,height=500,width=600,bg='#263d42')
        self.canvas.pack()
        self.frame.place(relwidth=.8, relheight=0.6, relx=0.1, rely=0.1)
        button=ttk.Button(root3, text="Add files", command=self.addfiles)


        button.pack(padx=5 , pady=5, side=tk.RIGHT)
        button=ttk.Button(root3, text="Add folder", command=self.add_folder)
        button.pack(padx=5 , pady=5, side=tk.RIGHT)
        button = ttk.Button(root3,text="Clear the screen",command=self.delete_data)
        button.pack(padx=5 , pady=5, side=tk.LEFT)
        button= ttk.Button(root3, text= "Backup",command = self.file_backup)
        button.pack(padx=5 , pady=5, side=tk.RIGHT)
        button= ttk.Button(root3, text= "AES(Fernet)",command = self.aes_fer_key)
        button.pack(padx=5 , pady=5, side=tk.RIGHT)
        button=ttk.Button(root3, text="RSA",command = self.rsa_keys)
        button.pack(padx=5 , pady=5, side=tk.RIGHT)
        tk.Label(self.frame,text="Add your files or folders here and choose one of the
opetions",bg="gray").pack()
        self.files = []
    def addfiles(self):
```

```
        for widget in self.frame.winfo_children():
            widget.destroy()
        types = [("all Files", " *.* "),("key Files", "*.key"),("encrypted Files",
"*.encrypted"),("decrypted Files", "*.decrypted")]
        filename = filedialog.askopenfilename(initialdir=".", title="Select Files ",
filetypes=types)
        self.files.append(filename)

        for file in self.files:
            self.files = [x for x in self.files if x.strip()]
            tk.Label(self.frame,text=file,bg="gray").pack()
    def delete_data(self):
        for widget in self.frame.winfo_children():
            widget.destroy()
        self.files.clear()
    def add_folder(self):
        for widget in self.frame.winfo_children():
            widget.destroy()
        filename = filedialog.askdirectory(initialdir=".", title="Select Files ")
        self.files.append(filename)
        for file in self.files:
            self.files = [x for x in self.files if x.strip()]
            tk.Label(self.frame,text=file,bg="gray").pack()



    def rsa_keys(self):
        global action1
        global pass_var1
```

```
        global root5
        root5 = tk.Toplevel(root)
        root5.geometry("700x600")
        root5.title("Options")
        tk.Label(root5,text="Actions",justify=tk.CENTER,padx=20).pack()
        action1= tk.IntVar()

tk.Radiobutton(root5,text="Encrypt",value=1,variable=action1,padx=20).pack()

tk.Radiobutton(root5,text='Decrypt',value=2,variable=action1,padx=20).pack()
        Label(root5,text="Enter your passphrase:").pack()
        pass_var1 = StringVar()
        Entry (root5,textvariable=pass_var1,fg= 'green',width=40,show="*").pack()
        for path in os.listdir():
            if os.path.isfile(username1+'.receiver.pem'):
                Label(root5,text="Your Public key:").pack()
                T = Text(root5,wrap=WORD,width=40,height=18)
                T.pack()
                with open(username1+'.receiver.pem', 'rb') as f:
                    T.insert(INSERT, f.read())
                    break

        ttk.Button(root5, text= "Close",width=
20,command=root5.destroy).pack(padx=5 , pady=5, side=tk.LEFT)
        ttk.Button(root5, text= "Start",width= 20,command=self.rsa).pack(padx=5 ,
pady=5, side=tk.RIGHT)

    def rsa(self):
        pass_phrase=pass_var1.get()
        byte_pass = pass_phrase.encode('utf-8')
```

```
        des = filedialog.askdirectory(initialdir='.',title='select a path')
        for path in os.listdir():
            if not os.path.isfile(username1+'.private.pem'):
                key = RSA.generate(2048)
                encrypted_key = key.export_key(passphrase=byte_pass, pkcs=8,
                                               protection="scryptAndAES128-CBC")
                public_key = key.publickey().export_key()
                with open(username1+".private.pem",'wb') as key:
                    key.write(encrypted_key)
                with open(username1+".receiver.pem", "wb") as key:
                    key.write(public_key)
        if action1.get() == 1:
            recipient_key = RSA.import_key(open(username1+".receiver.pem").read())
            session_key = get_random_bytes(16)
            cipher_rsa = PKCS1_OAEP.new(recipient_key)
            enc_session_key = cipher_rsa.encrypt(session_key)

            for line in self.files:
                source = (line.rstrip())
                with open(source, 'rb') as f:
                    data= f.read()
                cipher_aes = AES.new(session_key, AES.MODE_EAX)
                ciphertext, tag = cipher_aes.encrypt_and_digest(data)
                tail = (ntpath.split(source))
                file_Name = os.path.join(des, exusername1+tail[1]+".encrypted")
                with open(file_Name, 'wb') as ef:
                    [ef.write(x) for x in (enc_session_key, cipher_aes.nonce, tag,
ciphertext)].
            text = tk.Label(self.frame,text="Process Done!",fg='green')
            text.pack()
```

```
            root5.destroy()
            self.files.clear()


        elif action1.get() == 2:
            try:
             private_key =
RSA.import_key(open(username1+".private.pem").read(),byte_pass)
             for line in self.files:
                  source = (line.rstrip())
                  with open(source, 'rb') as f:
                     enc_session_key, nonce, tag, ciphertext = \
                     [ f.read(x) for x in (private_key.size_in_bytes(), 16, 16, -1) ]

                     cipher_rsa = PKCS1_OAEP.new(private_key)
                     session_key = cipher_rsa.decrypt(enc_session_key)
                     tail = (ntpath.split(source))
                     file_Name = os.path.join(des, username1+tail[1]+".decrypted")
                     with open(file_Name, 'wb') as ef:
                        cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce)
                        data = cipher_aes.decrypt_and_verify(ciphertext, tag)
                        ef.write(data)
                  self.failed_pass_rsa()
                  return 0
    def failed_pass_rsa(self):

        root5.destroy()
        text = tk.Label(self.frame,text="Process failed!",fg='red')
        text.pack()
        self.files.clear()
```

```
    def failed_pass_fernet(self):

        root4.destroy()
        text = tk.Label(self.frame,text="Process failed!",fg='red')
        text.pack()
        self.files.clear()

    def cry_aes(self):
            des = filedialog.askdirectory(initialdir='.',title='select a path')
            pass_phrase=pass_var.get()
            byte_pass = pass_phrase.encode('utf-8')
            with open(username1,"rb") as fs:  stored in and put it in the variable salt.
                salt = fs.readlines()[1]

            kdf =
PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=salt,iterations=1)

            key = base64.urlsafe_b64encode(kdf.derive(byte_pass))
            if action.get() == 1:
                for line in self.files:
                    source = (line.rstrip())
                    with open(source, 'rb') as f:
                        data= f.read()
                        f = Fernet(key)
                        encrypted_file = f.encrypt(data)
                    tail = (ntpath.split(source))
                    file_Name = os.path.join(des, username1+tail[1]+".encrypted")
                    with open(file_Name, 'wb') as ef:
                        ef.write(encrypted_file)
                text = tk.Label(self.frame,text="Process Done!",fg='green')
```

```
            text.pack()
            root4.destroy()
            self.files.clear()
        elif action.get() == 2:
            for line in self.files:
                source = (line.rstrip())
                with open(source, 'rb') as f:
                    data= f.read()
                    f = Fernet(key)
                    try:
                        decrypted_file = f.decrypt(data)
                    except InvalidToken:
                        self.failed_pass_fernet()
                        return 0
                tail = (ntpath.split(source))
                file_Name = os.path.join(des,tail[1]+".decrypted")
                with open(file_Name, 'wb') as df:
                 df.write(decrypted_file)
            text = tk.Label(self.frame,text="Process Done!",fg='green')
            text.pack()
            root4.destroy()
    def file_backup(self):

        des = filedialog.askdirectory(initialdir='.',title='select a path')
        for line in self.files:
            source = line.rstrip()
            if os.path.isdir(source):

shutil.copytree(source,des,symlinks=False,ignore=None,copy_function=shutil.cop
y2,ignore_dangling_symlinks=False,dirs_exist_ok=True)
```

```
            else:
                shutil.copy2(source,des)
        text = tk.Label(self.frame,text="Process Done!",fg='green')
        text.pack()
        self.files.clear()
    def aes_fer_key(self):

        global action
        global pass_var
        global root4  functions(cry_aes fun)
        root4 = tk.Toplevel(root)
        root4.geometry("400x250")
        root4.title("Options")
        tk.Label(root4,text="Actions",justify=tk.CENTER,padx=20).pack()
        action= tk.IntVar()


tk.Radiobutton(root4,text="Encrypt",value=1,variable=action,padx=20).pack()
        tk.Radiobutton(root4,text='Decrypt',value=2,variable=action,padx=20).pack()

        with open(username1,'rb+') as fk:
            if len(fk.readlines()) < 2 :

                salt =b'sal
                fk.write(salt)
        pass_var = StringVar()
        Label(root4,text="Enter your passphrase:").pack()# importing the passphrase
        Entry (root4,textvariable=pass_var,fg= 'green',width=40,show="*").pack()
```

```
        ttk.Button(root4, text= "Close",width=
20,command=root4.destroy).pack(padx=5 , pady=5, side=tk.LEFT)
        ttk.Button(root4, text= "Start",width=
20,command=self.cry_aes).pack(padx=5 , pady=5, side=tk.RIGHT)



def login_verify():

    global username1
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_entry1.delete(0, END)
    password_entry1.delete(0, END)
    list_of_files = os.listdir()
    if username1  in list_of_files:
        file1 = open(username1, 'r')
        verify = file1.read().splitlines()
        if password1 in verify:
            root2.destroy()
            Window()
        else:
            Label(root2, text="Password is not correct", fg='green',).pack()
    else:
        Label(root2, text="User is not found!", fg='green',).pack()
def login():

    global root2
    global username_verify
    global password_verify
```

```
    global username_entry1
    global password_entry1
    root2 = Toplevel(root)
    root2.title('Login')
    root2.geometry('400x500')
    Label(root2,text="Please enter your details to login").pack()
    Label(root2,text=" ").pack()
    username_verify = StringVar()
    password_verify = StringVar()
    Label(root2,text="Username *").pack()
    username_entry1 = Entry (root2, textvariable= username_verify)
    username_entry1.pack()
    Label(root2,text="Password *").pack()
    password_entry1 = Entry (root2, textvariable= password_verify, show="*")
    password_entry1.pack()
    Label(root2,text=" ").pack()
    Button(root2, text= 'Confirm', width='10', height='2', command=
login_verify).pack()
def register():

    global root1
    root1 = Toplevel(root)
    root1.title("Register")
    root1.geometry("500x400")
    global username
    global password
    global  username_entry
    global password_entry
    username = StringVar()
    password = StringVar()
```

```
    Label(root1,text="Please enter your details").pack()
    Label(root1,text=" ").pack()
    Label(root1,text="Username *").pack()
    username_entry = Entry(root1,textvariable = username)
    username_entry.pack()
    Label(root1,text="Password *").pack()
    password_entry = Entry(root1,textvariable = password, show="*")
    password_entry.pack()
    Label(root1,text=" ").pack()
    Button(root1, text="Register", width="10", height="2",command=
register_user).pack()


def register_user():
    username_info = username.get()
    password_info = password.get()
    if username_info in os.listdir(os.getcwd()):
        Label(root1, text="You already registred.", fg='green',).pack()
    elif username_info ==":
        Label(root1, text="Username field must not be emptry", fg='green',).pack()
    elif password_info == ":
            Label(root1, text="Password field must not be emptry",
fg='green',).pack()
    else:
        with open(username_info,'w',newline='') as new_user:
            writer = csv.writer(new_user, delimiter='\n',quoting=csv.QUOTE_NONE)
            writer.writerow([password_info])

        username_entry.delete(0, END)
        password_entry.delete(0, END)
```

```
Label(root1, text="Registration Sucess", fg='green',).pack()
```