Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтелектуальна система пошуку арбітражних можливостей між криптобіржами
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-22-1
Кирило КОТЕНКО
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Дмитро МАЛЄЄВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Котенку Кирилу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система пошуку арбітражних можливостей між криптобіржами

затверджена наказом університету від 5 червня 20 26 р. № 488Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 9 липня 20 26 р.
3. Вихідні дані до роботи моніторинг криптовалютного арбітражу; ринкові дані централізованих бірж: Binance, Bybit, OKX, KuCoin, Kraken, Coinbase отримані через публічні API.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичні дослідження
3) Проєктування та розробка моделі
4) Проєктування та розробка кінцевого застосунку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі та існуючих аналогів	08.06.2026 – 09.06.2026	виконано
3	Постановка задачі та обґрунтування технологій	10.06.2026 – 12.06.2026	виконано
4	Теоретичні дослідження та аналіз методів	13.06.2026 – 14.06.2026	виконано
	розробки		
5	Проєктування загальної архітектури та моделі	15.06.2026 – 18.06.2026	виконано
	даних		
6	Програмна реалізація вебзастосунку та модулів	19.06.2026 – 20.06.2026	виконано
	ШІ		
7	Тестування системи та оцінювання її	21.06.2026 – 22.06.2026	виконано
	функціональності		
8	Оформлення пояснювальної записки та додатків	23.06.2026 – 24.06.2026	виконано
9	Створення презентації та ілюстративних	25.06.2026 – 26.06.2026	виконано
	матеріалів		
10	Захист перед ЕК	09.07.2026	виконано

Дата видачі завдання 8 червня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Дмитро МАЛЄЄВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 74 с., 13 рис., 15 табл., 5 дод., 24 джерела.

АРБІТРАЖ КРИПТОВАЛЮТ, БЕКТЕСТИНГ, ВЕБЗАСТОСУНОК, ДЕРИВАТИВИ, КНИГА ЗАЯВОК, КРИПТОВАЛЮТНА БІРЖА, МОНІТОРИНГ РИНКУ, СПРЕД, СТАТИСТИЧНА ОЦІНКА РИЗИКУ, ТРІАНГУЛЯРНИЙ АРБІТРАЖ, ASYNCIO, CCXT, FUNDING RATE, OPEN INTEREST, STREAMLIT.

Об'єкт дослідження – процес виявлення та оцінювання арбітражних можливостей між централізованими криптобіржами в режимі реального часу.

Предмет дослідження – методи, алгоритми та програмні засоби автоматизації моніторингу міжбіржового, тріангулярного та крос-ринкового арбітражу з урахуванням комісій, ліквідності стакану та аналітики деривативів.

Мета роботи – розробити веб-систему для пошуку, оцінювання та візуалізації арбітражних можливостей між криптобіржами в реальному часі з урахуванням комісій, глибини стакану, оцінки ризику та аналітики деривативів.

Методи дослідження: системний і порівняльний аналіз; об'єктно-орієнтоване та модульне проєктування ПЗ; асинхронне програмування; статистичний аналіз часових рядів; метод VWAP для розрахунку fair price; бектестинг на OHLCV-даних; модульне тестування (pytest).

Технічні засоби: Python 3.9+, Streamlit, ccxt (async), Pandas, NumPy, Plotly, SQLite, asyncio, aiohttp, pytest.

Результати: розроблено локальний вебзастосунок (≈4 100 рядків, 8 пакетів, 34 модулі) трирівневої архітектури UI / Domain / Data Access.

ABSTRACT

Bachelor’s thesis contains: 74 pp., 13 fig., 15 tabl., 5 ann., 24 references.

ARBITRAGE, ASYNCIO, BACKTESTING, CCXT, CRYPTOCURRENCY ARBITRAGE, CRYPTOCURRENCY EXCHANGE, DERIVATIVES, FUNDING RATE, LIQUIDITY DENSITY, MARKET MONITORING, OPEN INTEREST, ORDER BOOK, RISK STATISTICAL ASSESSMENT, SPREAD, STREAMLIT, TRIANGULAR ARBITRAGE, WEB APPLICATION.

Object of research – the process of identification and evaluation of arbitrage opportunities between centralized cryptocurrency exchanges in real time.

Subject of research – methods, algorithms and software tools for automating inter-exchange, triangular and cross-market arbitrage monitoring with account for fees, order book liquidity and derivatives analytics.

Goal of the work – to develop a web-based system for detection, assessment and visualization of arbitrage opportunities between centralized cryptocurrency exchanges in real time with account for fees, order book depth, risk assessment and derivatives analytics.

Research methods: systems and comparative analysis; object-oriented and modular design; asynchronous programming; statistical time-series analysis; VWAP method; backtesting on OHLCV data; unit testing (pytest).

Technical tools: Python 3.9+, Streamlit, ccxt (async), Pandas, NumPy, Plotly, SQLite, asyncio, aiohttp, pytest.

Results: a local Streamlit web application (~4,100 lines, 8 packages, 34 modules) with a three-tier design (UI / Domain / Data Access) and seven tabs was developed. It performs concurrent polling of 5 exchanges (<2 s), multi-level caching, signal deduplication and executable-USD estimation by order book.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 10
1 Аналіз предметної галузі та постановка задачі 13
1.1 Опис предметної галузі 13
1.2 Поняття та види арбітражу на криптовалютному ринку 16
1.3 Аналіз чинників прибутковості арбітражу 19
1.4 Огляд існуючих рішень-аналогів 21
1.5 Обґрунтування актуальності розробки власної системи 24
1.6 Постановка задачі кваліфікаційного проєктування 25
2 Вибір технологій та проєктування системи 28
2.1 Обґрунтування вибору стека технологій 28
2.2 Архітектура застосунку 30
2.3 Проєктування моделей даних 33
2.4 Проєктування ключових алгоритмів 36
2.5 Проєктування інтерфейсу користувача 39
3 Практична реалізація системи 43
3.1 Структура проєкту та модулів 43
3.2 Реалізація шару доступу до бірж 44
3.3 Реалізація формування universe торгових пар 45
3.4 Реалізація основної арбітражної логіки 45
3.5 Реалізація сканера multi-symbol 46
3.6 Реалізація аналітики деривативів 46
3.7 Реалізація модуля статистичного оцінювання ризику 47
3.8 Реалізація шару персистенції 48
3.9 Реалізація бектестингу арбітражної стратегії 48
3.10 Реалізація графічного інтерфейсу користувача 49
3.11 Допоміжна інфраструктура 49
4 Тестування та аналіз результатів 50

4.1 Стратегія тестування 50
4.2 Модульне тестування 51
4.3 Інтеграційне тестування з реальними API бірж 51
4.4 Сценарне тестування інтерфейсу 52
4.5 Оцінка продуктивності системи 52
4.6 Аналіз отриманих арбітражних можливостей 54
4.7 Результати бектестингу типових стратегій 56
Висновки 59
Перелік джерел посилання 61
Додаток А Структура бази даних системи 63
Додаток Б Лістинги ключових модулів 65
Додаток В Схематичні представлення інтерфейсу системи 68
Додаток Г Звіт виконання тестів і фрагмент журналу подій 72
Додаток Д Відомість кваліфікаційної роботи 74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

Бектестинг – перевірка торгової стратегії на історичних ринкових даних;

Демо-режим – режим роботи системи із синтетичними даними;

Деривативи – похідні фінансові інструменти (ф’ючерси, перпетуальні контракти);

Кешування – тимчасове збереження даних для пришвидшення повторного доступу;

Крос-ринковий арбітраж – арбітраж між спотовим і деривативним ринками;

Латентність – сукупна затримка отримання, обробки та передавання ринкових даних;

Ставка фінансування – періодичний платіж між сторонами перпетуального контракту для утримання ціни біля спота;

СУБД – система управління базами даних;

Тикер – поточний знімок ринкових цін активу (bid, ask, last);

Тріангулярний арбітраж – арбітраж у межах однієї біржі через невузгодженість курсів трьох торгових пар;

AI – Artificial Intelligence – штучний інтелект;

AMM – Automated Market Maker – автоматизований маркет-мейкер;

API – Application Programming Interface – програмний інтерфейс;

CEX – Centralized Exchange – централізована криптобіржа;

CSS – Cascading Style Sheets – каскадні таблиці стилів;

CSV – Comma-Separated Values – значення, розділені комами;

DEX – Decentralized Exchange – децентралізована криптобіржа;

HTTP – HyperText Transfer Protocol – протокол передавання гіпертексту;

I/O – Input/Output – операції введення-виведення;

OHLCV – Open, High, Low, Close, Volume – свічкові дані ціни та обсягу;

OI – Open Interest – відкритий інтерес;

REST – Representational State Transfer – архітектурний стиль вебсервісів;

SQL – Structured Query Language – мова структурованих запитів;

TTL – Time To Live – час життя кешованих даних;

UI – User Interface – інтерфейс користувача;

UX – User Experience – досвід користувача;

VaR – Value at Risk – вартість під ризиком;

VWAP – Volume-Weighted Average Price – середньозважена ціна;

WebSocket – протокол двостороннього обміну даними.

ВСТУП

Актуальність теми. Ринок криптовалют за останнє десятиліття став повноцінним сегментом світової фінансової системи із капіталізацією понад 2 трлн дол. США та середньодобовим спот-оборотом понад 100 млрд дол. Інфраструктура ринку фрагментована: один і той самий актив торгується паралельно на десятках централізованих (CEX) і сотнях децентралізованих (DEX) бірж, кожна з власною книгою заявок, глибиною ліквідності та політикою комісій. Унаслідок цього ціна одного інструмента (наприклад, BTC/USDT) у будь-який момент відрізняється між майданчиками, формуючи арбітражні спреди [9], [17]. Купівля на біржі з нижчою ціною і продаж на біржі з вищою дає прибуток без напрямної позиції – у цьому сутність міжбіржового арбітражу [21]. Стрімкий розвиток фінансових технологій та алгоритмічної торгівлі лише підкреслює необхідність системного аналізу ринкових неефективностей, що робить цей напрямок досліджень особливо затребуваним у сучасних умовах.

Попри концептуальну простоту, реальне виявлення прибуткових можливостей ускладнюється короткочасністю спредів (частки-десятки секунд), нерівномірністю комісій (0,1–0,6 % taker), обмеженою ліквідністю стакану та обмеженнями API. Це робить ручний моніторинг непридатним і обумовлює потребу в спеціалізованих засобах, здатних паралельно опитувати біржі, нормалізувати дані, рахувати чистий прибуток з урахуванням комісій і надавати зрозумілу візуалізацію в реальному часі. Існуючі комерційні рішення (ArbitrageScanner, CoinGlass, Hummingbot, Cryptohopper) [10], [11], [14] переважно закриті, платні й зосереджені на двобіржевому арбітражі без урахування деривативів, тріангулярного арбітражу та глибини ліквідності. Відкритих модульних рішень, які одночасно покривали б ці напрями, виконували бектестинг і оцінювали ризик, практично немає – що формує наукове та прикладне підґрунтя для розробки відповідної системи з відкритою архітектурою. Створення такого

інструменту не лише автоматизує процес пошуку прибуткових угод, але й сприяє загальному підвищенню ефективності крипторинку за рахунок швидкого згладжування цінових диспропорцій. Водночас відкритий вихідний код проєкту стимулює академічну спільноту до подальших досліджень у сфері мікроструктури ринку.

Мета роботи – розробити інтелектуальну веб-систему для автоматичного пошуку, оцінювання та візуалізації арбітражних можливостей між централізованими криптобіржами в реальному часі з урахуванням реальних торгових комісій, виконуваності за глибиною стакану, статистичної оцінки ризику та інтегрованої аналітики ринку деривативів.

Для досягнення мети сформульовано задачі: проаналізувати предметну галузь і рішення-аналоги; сформулювати функціональні та нефункціональні вимоги; обґрунтувати вибір стека технологій; спроєктувати архітектуру з розшаруванням UI/Domain/Data Access; розробити доменну модель і реляційну схему сховища; реалізувати асинхронний шар доступу до API бірж, алгоритми дво- і тріангулярного арбітражу з урахуванням комісій, tiered-сканер universe топ-200 USDT-пар Binance, модуль аналітики деривативів (OI, funding, спреди, плотності), модуль статистичного оцінювання ризику, модуль бектестингу з розрахунком метрик; розробити веб-інтерфейс із сімома вкладками; провести модульне, інтеграційне та сценарне тестування.

Об'єкт дослідження – процес виявлення та оцінювання арбітражних можливостей між централізованими криптобіржами в режимі реального часу. Предмет дослідження – методи, алгоритми та програмні засоби автоматизації моніторингу міжбіржового, тріангулярного та крос-ринкового арбітражу з урахуванням комісій, ліквідності стакану та аналітики деривативів.

Методи дослідження: системний аналіз предметної галузі; порівняльний аналіз рішень-аналогів; методи об'єктно-орієнтованого та

модульного проєктування ПЗ; асинхронне програмування (модель event loop); статистичний аналіз часових рядів (стандартне відхилення, дисперсія); метод VWAP для розрахунку fair price; бектестинг на історичних OHLCV-даних; модульне тестування з pytest [20].

Наукова новизна. Запропоновано модульну архітектуру системи моніторингу, що поєднує в одному рішенні чотири різнорідні класи арбітражних стратегій (двобіржевий спот, тріангулярний, futures-futures, spot-futures), традиційно реалізовані окремо; розроблено механізм багаторівневого tiered-сканування universe з диференційованим інтервалом оновлення (15/30/60 с) за об'ємними тирами; запропоновано алгоритм оцінювання реально виконуваного обсягу угоди (executable USD) через послідовне проходження рівнів стакану замість лише best bid/ask; інтегровано статистичне оцінювання ризику за коефіцієнтом варіації цін із класифікацією LOW/MEDIUM/HIGH і дедуплікацією сигналів за часовим інтервалом.

Практичне значення. Розроблено повнофункціональний прототип веб-системи, придатний до безпосереднього використання для моніторингу та як база для подальших досліджень мікроструктури ринку. Відкрита архітектура дозволяє розширювати перелік бірж, додавати класи стратегій та інтегрувати моделі машинного навчання. Demo Mode без зовнішніх API робить систему придатною для презентаційних і навчальних цілей.

Структура роботи. Записка складається зі вступу, чотирьох розділів, висновків, переліку джерел і чотирьох додатків. Розділ 1 – аналіз предметної галузі та постановка задачі; розділ 2 – обґрунтування технологій, проєктування архітектури, моделей даних і алгоритмів; розділ 3 – програмна реалізація модулів; розділ 4 – тестування, оцінка продуктивності та аналіз результатів. Записку оформлено відповідно до чинних стандартів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Розробка системи моніторингу арбітражу потребує розуміння інфраструктури, у якій формуються ціни на цифрові активи. Криптовалютний ринок відрізняється від традиційних відсутністю єдиного централізованого майданчика: торгівля одним активом ведеться одночасно на десятках незалежних бірж, кожна з власною книгою заявок, політикою комісій і API. Ця особливість є передумовою існування арбітражних можливостей і одночасно джерелом інженерних викликів. Така фрагментація ринкової ліквідності створює унікальне середовище, де цінові дисбаланси виникають регулярно, проте існують протягом дуже коротких проміжків часу і вимагають миттєвої реакції для їх успішної експлуатації.

За принципом організації обігу активів біржі поділяють на централізовані (CEX), децентралізовані (DEX) та гібридні (таблиця 1.1). CEX зберігають активи на власних рахунках (custodial), виконують заявки централізованим matching engine за алгоритмом «ціна-час», забезпечують мілісекундну швидкість, глибоку ліквідність і стандартизовані REST/WebSocket-API. DEX працюють non-custodial через смарт-контракти, переважно за моделлю автоматизованого маркет-мейкера (AMM, формула $x \cdot y = k$), надаючи контроль над активами, але поступаючись швидкістю (час блоку) і ліквідністю та потребуючи газу. Гібридні біржі поєднують off-chain matching з on-chain розрахунками. Основним джерелом даних для системи є саме CEX; у роботі розглянуто п'ять провідних бірж – Binance, Kraken, KuCoin, Bybit, Coinbase[3], [23], [15], [4], [8], що сумарно дають понад 60 % світового спот-обороту. Децентралізований сегмент залучено опосередковано через DEX-proxy (агрегований spot з Kraken і Coinbase).

Таблиця 1.1 – Класифікація криптовалютних бірж

Характеристика	CEX	DEX	Гібридні
Зберігання активів	На рахунках біржі (custodial)	У гаманці користувача (non-custodial)	Поєднання обох моделей
Механізм виконання	Централізований matching engine	Smart contract / AMM	Off-chain matching + on-chain settlement
Швидкість виконання	Мілісекунди	Секунди (block time)	Мілісекунди – секунди
Глибина ліквідності	Висока	Помірна / низька	Висока
Типові комісії taker	0,10-0,60 %	0,20-0,50 % + gas fee	0,05-0,25 %
Регуляторна підзвітність	Висока (KYC/AML)	Низька	Помірна
Доступ через API	Стандартизований REST/WebSocket	Web3 RPC / GraphQL	Гібридний
Типові представники	Binance,Coinbase, Kraken, KuCoin, Bybit	Uniswap, PancakeSwap, SushiSwap	dYdX, Serum

Центральним елементом CEX є книга заявок (order book) – упорядкований реєстр лімітних заявок [13] на купівлю (bid, за спаданням ціни) і продаж (ask, за зростанням). Найкращі ціни – best bid і best ask, їх різниця – bid-ask spread. Ринкова заявка послідовно виконується з кращих рівнів протилежної сторони, формуючи last price. Для арбітражу принциповим є спред між best bid однієї біржі та best ask іншої. Структуру книги заявок і кумулятивну криву глибини показано на рисунку 1.1. Аналізуються топ-20 рівнів стакану з кожної сторони, оскільки саме на цій глибині реалізуються типові арбітражні обсяги (100–10 000 USD).

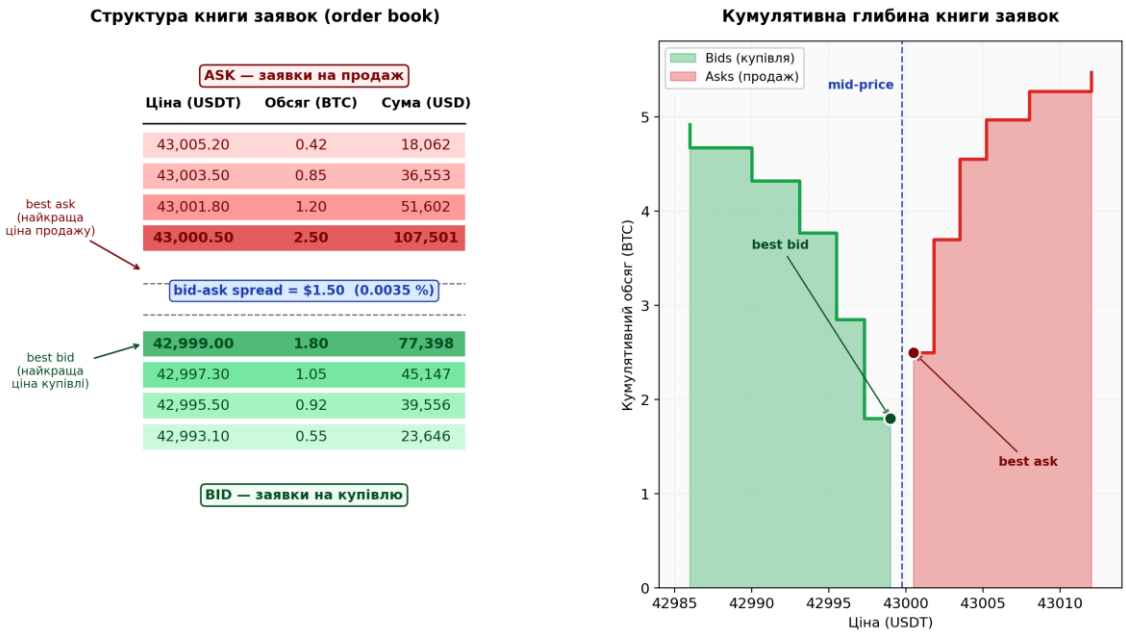


Рисунок 1.1 – Структура книги заявок (order book) на CEX та відповідна кумулятивна крива глибини

Ліквідність характеризують трьома вимірами: щільність (tightness – малість спреду), глибина (depth – обсяг без зміщення ціни) та пружність (resilience). Відносний спред: spread % = (ask − bid)/bid 100 %. Міжбіржевий арбітражний спред: arb_spread % = (max_bid − min_ask)/min_ask · 100 % – валовий, без комісій і цінового впливу. Чистий прибуток: net_profit % = arb_spread % − fee_buy % − fee_sell %. Додатній net_profit ще не гарантує реалізації: якщо інвестиційна сума перевищує обсяг top-of-book, частина угоди виконується за гіршими цінами (проскальзування, slippage). Для врахування цього запропоновано концепцію executable USD – інтегральну оцінку максимального обсягу, що реально виконується при поточному стані стакану; алгоритм послідовно проходить рівні стакану, накопичуючи добутки ціни на обсяг. Ліквідність є нестаціонарною (змінюється протягом доби, у вихідні, під час новин), що система враховує через перевірку executable depth і візуалізацію статусу доступності бірж.

1.2 Поняття та види арбітражу на криптовалютному ринку

Арбітраж – отримання прибутку з різниці цін одного активу на різних ринках без напрямного цінового ризику. Через фрагментацію між майданчиками і сегментами (спот, перпи, опціони) сформувалося кілька класів стратегій [7] (рисунок 1.2). Система моніторить чотири: просторовий, тріангулярний, статистичний/latency (оглядово) та крос-ринковий.

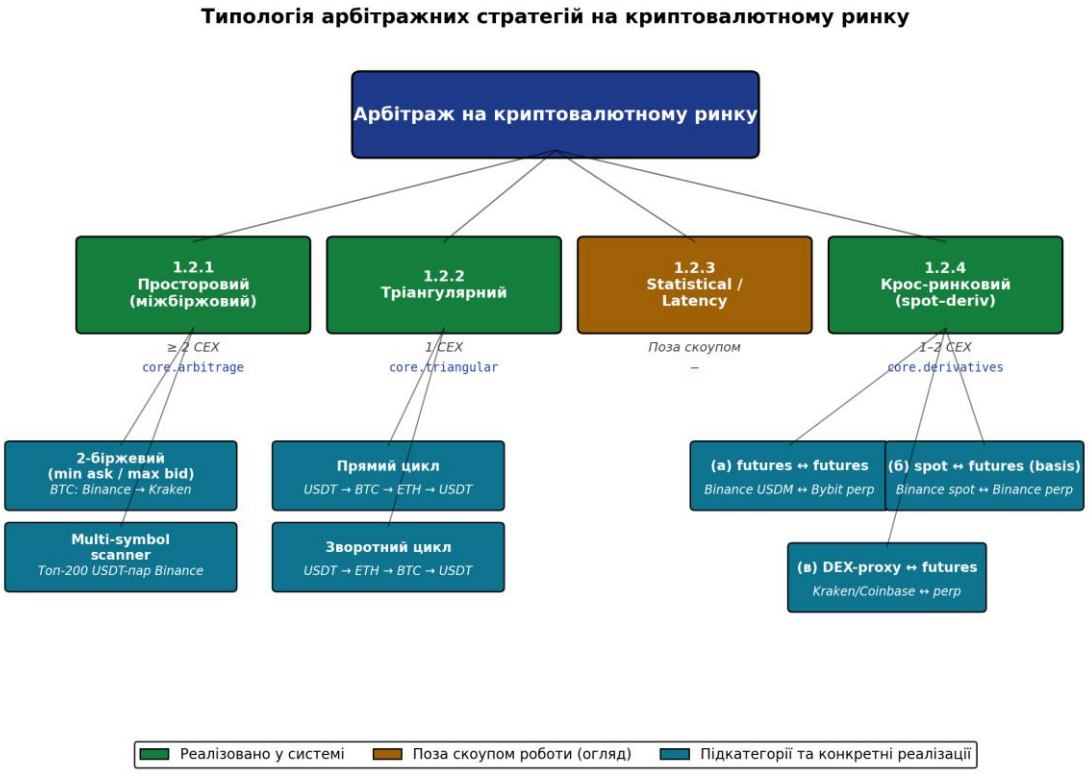


Рисунок 1.2 – Типологія арбітражних стратегій на криптовалютному ринку

Просторовий (міжбіржовий) арбітраж – одночасна купівля на біржі з нижчим ask і продаж на біржі з вищим bid (модуль core.arbitrage, вкладка LIVE MONITOR).

Валовий спред spread_gross = (max bid − min ask)/min ask 100 %; чистий profit_net = spread_gross − (f_buy + f_sell)·100 %. Опція реалізована за $i \neq j$ та profit_net > 0. Переваги - простота, відсутність утримання позицій;

обмеження – потреба попереднього розміщення капіталу на обох біржах, ризик закриття вікна, вищі комісії на біржах нижчого рівня.

Тріангулярний арбітраж виконується в межах однієї біржі через невузгодженість курсів трьох пар (наприклад, USDT→BTC→ETH→USDT; модуль core.triangular). Прямий цикл з інвестицією I і комісією f: $N_1 = (I/ask(P_1))\cdot(1-f)$; $N_2 = (N_1/ask(P_2))\cdot(1-f)$; $R = N_2\cdot bid(P_3)\cdot(1-f)$; $profit_tri\ \% = (R-I)/I\cdot 100\%$. Симетрично визначається зворотний цикл. Числовий приклад наведено на рисунку 1.3. Перевага – атомарність на одній біржі (мінімум latency-ризику); недолік – високоліквідні пари (BTC, ETH) майже миттєво арбітражуються маркет-мейкерами, тож реальні можливості частіше на alt-парах.

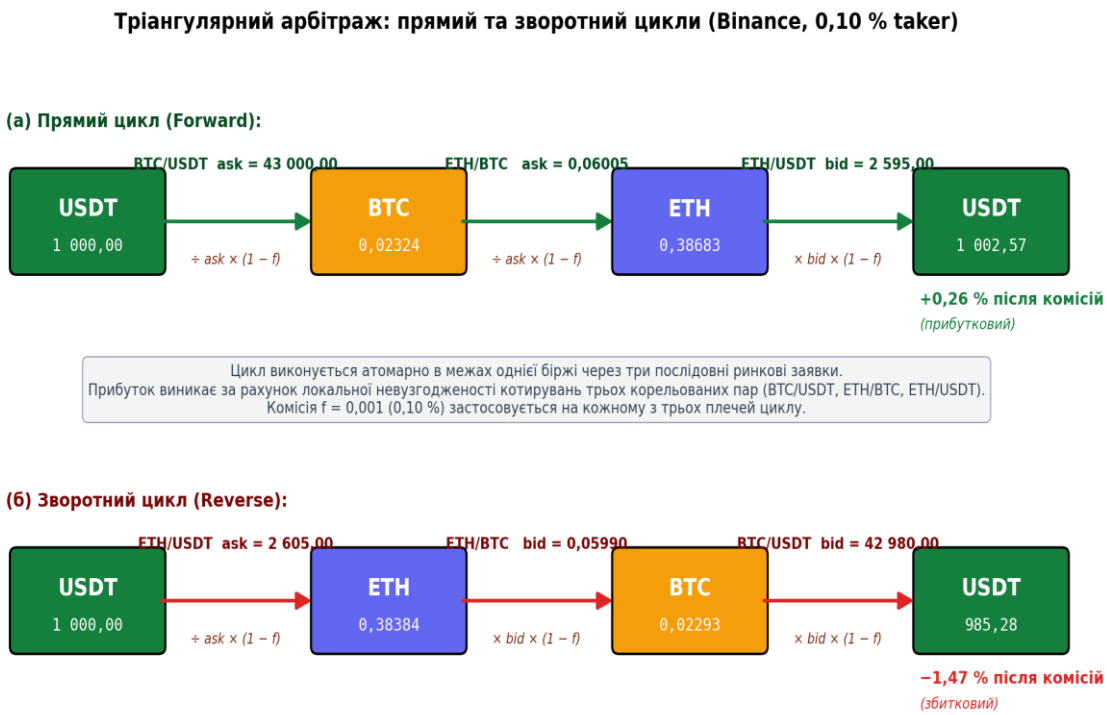


Рисунок 1.3 – Числовий приклад тріангулярного арбітражу: прямий (прибутковий) та зворотний (збитковий) цикли

Статистичний арбітраж (pair trading, mean-reversion, VAR, ML) ґрунтується на історичних кореляціях [6]; latency-арбітраж експлуатує

мікросекундні затримки між біржами [1] й потребує co-location та низькорівневих мов. Обидва класи згадано оглядово й винесено за межі скоупу як такі, що потребують принципово іншої інфраструктури.

Крос-ринковий арбітраж (спот-деривативи; модуль core.derivatives.spread_scanner, вкладка DERIVATIVES) має три підвиди: (а) futures-futures – різниця mid-ціни перпа між двома деривативними біржами (Binance USDM, Bybit perp); (б) spot-futures (basis trade, «cash and carry») – різниця між ціною спота і перпа, прибуткова за позитивного funding rate; (в) DEX-futures (DEX-proxy) – розрив між централізованими деривативами й проксі децентралізованого спота. Критичну роль відіграють метрики Open Interest і funding rate (модуль core.derivatives.oi_monitor). Порівняння класів стратегій наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльні характеристики класів арбітражних стратегій

Характеристика	Просторовий	Тріангулярний	Statistical / Latency	Крос-ринковий (spot-deriv)
Кількість майданчиків	≥ 2 CEX	1 CEX	≥ 2 CEX + інфраструктура	1-2 CEX (spot + deriv)
Тип позиції	Спот ↔ спот	Спот-цикл усередині біржі	Довга + коротка корельовані	Спот ↔ perp / perp ↔ perp
Тривалість позиції	Секунди	Секунди	Години - дні	Години - тижні
Типовий чистий прибуток за угоду	0,05-0,5 %	0,01-0,3 %	0,1-2 % на цикл	0,1-1 % + funding
Інженерна складність	Помірна	Низька	Висока (ML, низька latency)	Помірно-висока

Продовження таблиці 1.2

Характеристика	Просторовий	Тріангулярний	Statistical / Latency	Крос-ринковий (spot-deriv)
Реалізація в системі	core.arbitrage	core.triangular	Поза скоупом (лише огляд)	core.derivatives. spread_scanner
Основні ризики	Latency, ліквідність	Конкуренція маркет-мейкерів	Розрив кореляції	Зміна funding, ліквідація

Для дослідницької системи пріоритетними є просторовий, тріангулярний і крос-ринковий арбітраж: вони мають прозору математичну природу і реалізуються в межах типових REST/WebSocket API публічних бірж.

1.3 Аналіз чинників прибутковості арбітражу

Реальна прибутковість визначається не лише валовим спредом, а й чинниками, що зменшують теоретичну вигоду: торгові комісії, латентність та обмежена ліквідність.

Торгові комісії – найпередбачуваніший і найвпливовіший чинник. Застосовується модель maker-taker; арбітраж потребує негайного виконання, тож використовується taker-комісія. Її розмір суттєво різниться між біржами (таблиця 1.3): 0,10% на Binance/KuCoin/Bybit, 0,26% на Kraken, 0,60% на Coinbase. Деривативні комісії нижчі (Binance USDM 0,04%, Bybit perp 0,06%). Обчислення винесено в модуль core.fees (get_fee, calculate_net_profit). Через розкид комісій спред 0,3% прибутковий на Binance-KuCoin (сумарна комісія 0,2%), але збитковий на Binance-Coinbase (0,7%).

Таблиця 1.3 – Базові taker-комісії централізованих бірж (спот) станом на 2026 рік

Біржа	Taker-комісія, %	Taker-комісія, частка	Примітка
Binance	0,10	0,001	Знижка до 0,075 % при оплаті в BNB
KuCoin	0,10	0,001	Знижка до 0,08 % при оплаті в KCS
Bybit	0,10	0,001	Знижка для VIP-рівнів
Kraken	0,26	0,0026	Висока стартова комісія, знижка від обороту
Coinbase Advanced	0,60	0,006	Одна з найвищих комісій серед топ-CEX

Латентність. Сукупна затримка (отримання тикеру, обробка, передача заявки, matching) для архітектури на REST-API складає 0,5–2 с. За цей час ціна може змінитися або можливість зникнути (execution slippage). Дослідження свідчать, що до 70 % формально виявлених можливостей не реалізуються через цей чинник [17]. Для дослідницької системи проблема вирішується кешуванням з обмеженим TTL (15 с для тикерів) та візуалізацією часу оновлення й доступності бірж (exchange_health).

Ліквідність та проскальзування. Ефективна ціна заявки на обсяг Q розраховується за формулою:

$$Price_{eff(Q)} = \sum \frac{p_k * q_k}{Q}. \quad (1.1)$$

де p_k – ціна на k-му рівні стакану;

q_k – обсяг на k-му рівні стакану;

Q – загальний обсяг заявки.

Чим менш ліквідна пара і більший обсяг, тим сильніше відхилення від best ask (відносний slippage). На рисунку 1.4 показано: при сумах до 1 000 USD ефект несуттєвий, але понад 5 000–10 000 USD slippage поглинає значну частину прибутку, а від 50 000 USD вікно для другорядних пар повністю «з'їдається». Тому система оцінює максимальний прибутковий обсяг через executable USD (функція estimate_executable_usd модуля core.orderbook) та візуалізує плотності ліквідності (heatmap, модуль core.derivatives.orderbook_density).

Отже, ідентифіковано три чинники (комісії, латентність, ліквідність), реалізовані у відповідних модулях: core.fees, TTL-кешування + exchange_health, executable USD + плотності стакану.

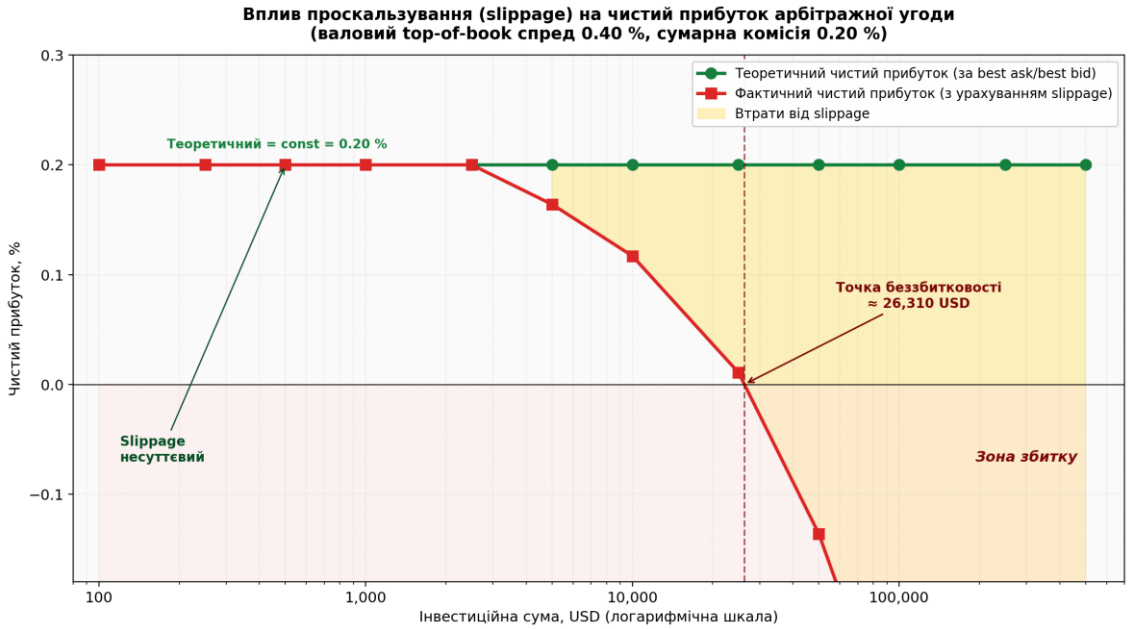


Рисунок 1.4 – Залежність чистого прибутку арбітражної угоди від інвестиційної суми з урахуванням slippage за стаканом

1.4 Огляд існуючих рішень-аналогів

Рішення поділяють на веб-сервіси моніторингу та торгові боти. ArbitrageScanner.com – комерційний сканер 80+ бірж (CEX/DEX) з

фільтрацією за спредом, підписка від 70 USD/міс, код закритий. CoinGlass – аналітична платформа деривативів (OI, funding, long/short, ліквідації, basis-спреди) без налаштовуваного фільтру за прибутковістю. Coinmonitor.info – просте табличне порівняння цін без аналітики й комісій. Спільна риса – закрита архітектура, відсутність доступу до сирих даних, бектестингу і моделювання executable depth (типовий вигляд інтерфейсів – рисунок 1.5).

Торгові боти: Hummingbot – відкрите (Apache 2.0) рішення для алгоритмічної торгівлі, 30+ бірж, кілька стратегій, фреймворк на Python, але потребує приватних API-ключів із дозволом на торгівлю. Cryptohopper – комерційний «no-code» сервіс (підписка 19–99 USD), маркетплейс стратегій, теж потребує приватних ключів і не має режиму «лише моніторинг». Нішові Trality, Pionex, Bitsgap – вертикально інтегровані.

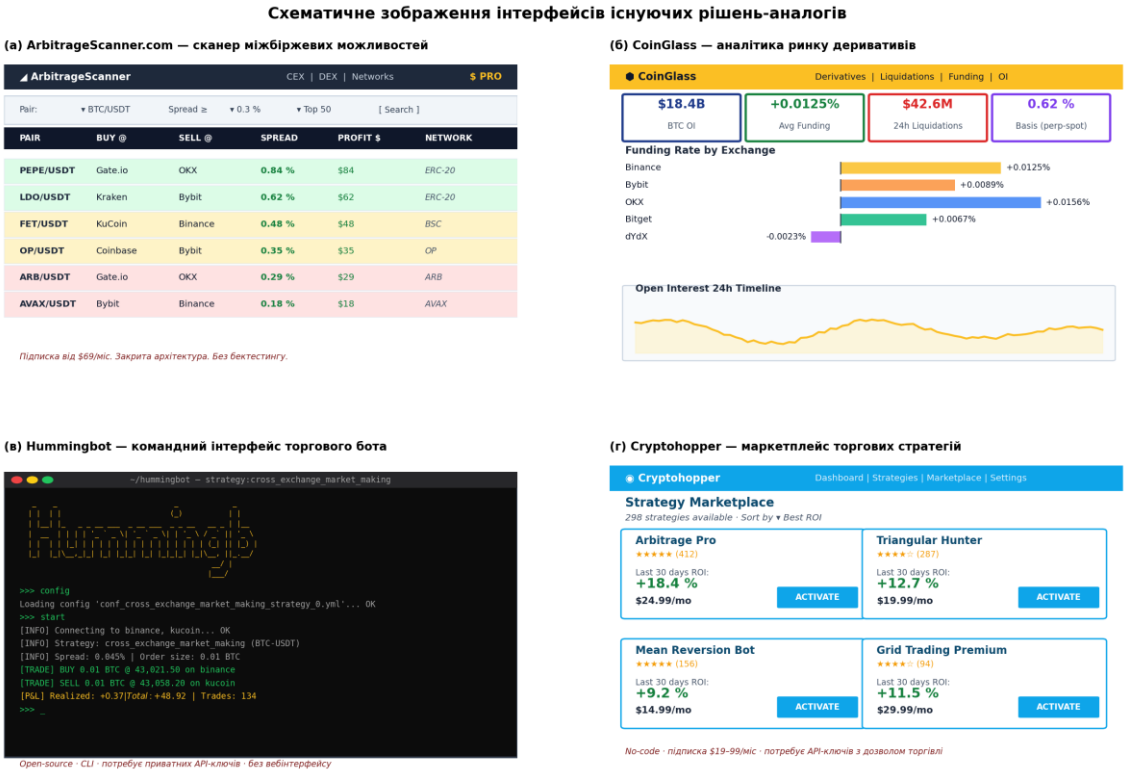


Рисунок 1.5 – Схематичне зображення типових інтерфейсів існуючих рішень-аналогів: (а) ArbitrageScanner.com, (б) CoinGlass, (в) Hummingbot, (г) Cryptohopper

Порівняння рішень за 19 характеристиками наведено в таблиці 1.4.

Таблиця 1.4 – Порівняльна характеристика існуючих рішень моніторингу та автоматизації криптоарбітражу

Характеристика	Arbitrage Scanner	CoinGlass	Hummingbot	Cryptohopper	Розроблювана система
Тип ліцензії / моделі	Підписка ($70+/міс)	Freemium + API	Open-source (Apache 2.0)	Підписка ($19-99/міс)	Освітня, відкрита
Відкритість коду	-	-	+	-	+
Кількість CEX	50+	20+	30+	15+	5+ (модульно)
Підтримка DEX	+	Частково	+	-	DEX-proxy
2-біржевий арбітраж	+	-	+	+	+
Тріангулярний арбітраж	Частково	-	+	+	+
Futures-futures спред	-	+ (огляд)	+	Частково	+
Spot-futures (basis)	-	+	+	-	+
Open Interest / Funding	-	+	-	-	+
Плотності стакану	-	-	-	-	+
Урахування комісій	Частково	-	+	+	+
Executable depth	-	-	+	-	+
AI Risk Score	-	-	-	-	+
Бектестинг на OHLCV	-	Частково	+	Частково	+
Демо-режим (без API ключів)	+	+	-	-	+

Продовження таблиці 1.4

Характеристика	Arbitrage Scanner	CoinGlass	Hummingbot	Cryptohopper	Розроблювана система
Реальне виконання угод	-	-	+	+	- (моніторинг)
Веб-інтерфейс	+	+	- (CLI)	+	+ (Streamlit)
Цільова аудиторія	Трейдери	Аналітики	Розробники	Роздрібні трейдери	Дослідники / студенти

Виявлено п'ять обмежень аналогів: (1) закритість архітектури комерційних сервісів – неможливість адаптації під дослідницькі задачі; (2) вертикальна інтеграція ботів - обов’язкові приватні ключі, непридатність для навчальних сценаріїв; (3) фрагментарність – жодне рішення не поєднує чотири класи стратегій + ризик + бектестинг; (4) відсутність моделювання executable depth (крім Hummingbot); (5) відсутність демо-режиму у виконавчих ботах. Ці обмеження формують функціональну дельту для розробки власної системи: відкрита модульна архітектура, виключно моніторинговий характер, інтегроване покриття чотирьох класів, реалістичне executable depth, демонстраційний режим.

1.5 Обґрунтування актуальності розробки власної системи

Обґрунтування ґрунтується на трьох групах факторів. Ринкові: ринок зростає кількісно (>2 трлн дол. капіталізації, >100 млрд дол. обороту) і структурно ускладнюється (перпи з OI >100 млрд дол., опціони, кросчейн-мости); фрагментація є джерелом постійних арбітражних можливостей, попит на засоби моніторингу не насичений. Технологічні: зрілість ccxt (єдиний інтерфейс до 100+ бірж), asyncio/aiohttp (паралельне опитування) та Streamlit (повноцінні веб-застосунки силами одного розробника) дозволяє реалізувати систему обсягом ~4 тис. рядків коду в межах однієї кваліфікаційної роботи. Методологічні: сформована

функціональна прогалина – інтегрований відкритий дослідницький інструмент, що поєднує всі чотири класи стратегій, аналітику деривативів, executable depth, бектестинг, демо-режим і відкритий код; жодне рішення не задовольняє всі шість критеріїв. Додатково система має педагогічну цінність як відкритий приклад застосування асинхронного програмування, модульного проєктування, статистики, реляційних СУБД та веб-візуалізації.

1.6 Постановка задачі кваліфікаційного проєктування

Ефективна розробка будь-якого складного програмного забезпечення обов'язково розпочинається з чіткого та однозначного формулювання вимог до майбутньої системи. У контексті даної кваліфікаційної роботи ключова мета полягає у створенні надійного аналітичного інструментарію, що зумовлює жорсткі критерії щодо його функціонального наповнення. Функціональні вимоги (за сімома вкладками): ФВ-1 – збір ринкових даних з 5 CEX у реальному часі (паралельно, ≤2 с, зі статусом доступності); ФВ-2 – розрахунок міжбіржевого і тріангулярного арбітражу з валовим спредом, комісіями, чистим прибутком і fair price (VWAP-mid); ФВ-3 – сканування universe топ-200 USDT-пар Binance за трьома тирами з фільтрами та експортом у CSV; ФВ-4 – аналітика деривативів (OI, funding, спреди futures↔futures/spot↔futures/DEX-proxy, плотності стакану); ФВ-5 – статистичне оцінювання ризику за коефіцієнтом варіації з класифікацією LOW/MEDIUM/HIGH; ФВ-6 – персистенс та історія сигналів з дедуплікацією у вікні 60 с; ФВ-7 – бектестинг на історичних OHLCV з розрахунком win rate, прибутку, просадки, коефіцієнта Шарпа; ФВ-8 - облік портфоліо (депозит, баланс, equity curve). Такий деталізований розподіл завдань гарантує комплексне охоплення всіх етапів арбітражного процесу, починаючи від первинного збору даних і закінчуючи їх глибоким ретроспективним аналізом. Зведення наведено в таблиці 1.5.

Таблиця 1.5 – Зведення функціональних вимог до системи

Код	Назва вимоги	Відповідальний модуль / вкладка
ФВ-1	Збір ринкових даних з 5 CEX у реальному часі	core.fetcher, core.market.ticker_service / LIVE MONITOR
ФВ-2	Розрахунок міжбіржевого і тріангулярного арбітражу	core.arbitrage, core.triangular / LIVE MONITOR
Код	Назва вимоги	Відповідальний модуль / вкладка
ФВ-3	Сканування universe топ-200 USDT-пар Binance	core.scanner, core.symbols.universe / SCANNER
ФВ-4	Аналітика OI, funding, basis, плотностей	core.derivatives.* / DERIVATIVES
ФВ-5	Статистичне оцінювання ризику сигналу	core.risk / AI RISK
ФВ-6	Персистенс та історія сигналів із дедупом	database.queries / HISTORY LOG
ФВ-7	Бектестинг на історичних OHLCV-даних	backtesting.engine / БЕКТЕСТИНГ
ФВ-8	Облік портфоліо: депозит, баланс, equity curve	database.queries, ui.charts / ПОРТФОЛІО
ФВ-9	Демонстраційний режим (Demo Mode)	усі core-модулі, прапор demo_mode
ФВ-10	Експорт результатів у CSV	вкладки SCANNER, HISTORY LOG, ПОРТФОЛІО

Окрім безпосереднього виконання бізнес-логіки, критично важливим аспектом є забезпечення стабільної та швидкої роботи додатку в умовах інтенсивного потоку біржових даних. Саме тому було сформовано розширений перелік нефункціональних вимог, які гарантують високу якість кінцевого програмного продукту. Нефункціональні вимоги: НФВ-1 (продуктивність) – відгук LIVE MONITOR ≤3 с, повне сканування universe ≤30 с за рахунок batch-запитів (fetch_tickers) і asyncio.gather [16];

НФВ-2 (кешування) – багаторівневе: пул клієнтів @st.cache_resource, snapshot тикерів TTL 15 с, стакан 10 с, universe 1 год з disk-кешем, карта ринків 24 год, OHLCV 1 год; НФВ-3 (відмовостійкість) – продовження аналізу при частковій недоступності бірж, логування винятків у arbitrage.log; НФВ-4 (розширюваність) – додавання бірж через словник EXCHANGES без зміни логіки; НФВ-5 (зручність) – браузерний інтерфейс, запуск streamlit run app.py; НФВ-6 (тестованість) – відокремлення логіки від UI, покриття core ≥60 %; НФВ-7 (сумісність) – Windows/macOS/Linux, Python 3.9+.

Обмеження та припущення: 5 CEX + 2 деривативні майданчики (розширення тривіальне); реалізовано просторовий, тріангулярний і крос-ринковий арбітраж (statistical/latency - поза скоупом); система виключно моніторингова, без реального виконання угод; використано REST API (не WebSocket); СУБД – SQLite; прийнято базові (найгірші) taker-комісії за тарифами 2026 року; executable USD аналізує топ 10–20 рівнів; цільове розгортання – локальний запуск. Визначені вище обмеження та допущення не зменшують практичної цінності проєкту, а навпаки, дозволяють сфокусувати дослідницькі зусилля на найбільш значущих векторах арбітражного аналізу. Сформована таким чином постановка задачі слугує надійним фундаментом для подальшого проєктування архітектури системи та її безпосередньої програмної реалізації. Більше того, такий підхід гарантує високий рівень модульності та масштабованості кодової бази, що залишає простір для інтеграції додаткових торгових майданчиків у майбутньому. Це також дозволяє мінімізувати ймовірність виникнення критичних помилок на етапі імплементації ключових алгоритмів обробки даних. У підсумку, деталізований перелік усіх вимог стає своєрідною дорожньою картою, яка забезпечує успішне досягнення головної мети кваліфікаційного дослідження.

2 ВИБІР ТЕХНОЛОГІЙ ТА ПРОЄКТУВАННЯ СИСТЕМИ

У розділі виконано перехід від «що робити» до «як робити»: обґрунтовано вибір стека технологій, спроєктовано архітектуру, моделі даних, ключові алгоритми та інтерфейс користувача відповідно до нефункціональних вимог НФВ-1…НФВ-7.

2.1 Обґрунтування вибору стека технологій

Мову Python обрано як де-факто стандарт quantitative finance [16], [7], [18] (зрілі Pandas, NumPy [18], [16], SciPy, scikit-learn, backtrader тощо), що забезпечує високу швидкість розробки і придатний для високопаралельного мережевого I/O (async/await, asyncio, aiohttp); критичним вузьким місцем системи є мережеві затримки, а не CPU, тож компільовані мови (C++, Rust, Go) дали б приріст коду без приросту продуктивності. Python є основною мовою кафедри ШІ ХНУРЕ. Альтернативи (TypeScript/Node.js, Java/Kotlin, Go) відхилено через слабшу екосистему наукових обчислень і відсутність зрілого еквівалента ccxt.

Веб-фреймворк Streamlit обрано[24] для швидкої побудови інтерактивних data-app: він описує UI декларативно в тому ж Python-файлі, що й логіку, має вбудовану модель стану (st.session_state) і кешування (@st.cache_data, @st.cache_resource), що елегантно вирішує TTL-кешування (НФВ-2), та систему вкладок/колонок без HTML/CSS. Альтернативи (Flask+Jinja2, FastAPI+React, Dash, Voilà) відхилено через шаблонний код або подвоєння кодової бази.

Бібліотеку ccxt (async_support) обрано [5] як уніфікований інтерфейс до 100+ бірж: один набір методів (fetch_ticker, fetch_tickers, fetch_order_book, fetch_ohlcv, load_markets), специфіка біржі інкапсульована - додавання біржі потребує лише доповнення EXCHANGES у config.py (НФВ-4). Підтримуються й деривативні обгортки (binanceusdm,

bybit swap). СУБД SQLite обрано як вбудовану (модуль sqlite3, єдиний файл arbitrage.db, без сервера) - достатню для одно-користувацького застосунку; перехід на PostgreSQL тривіальний (заміна get_connection()). Візуалізація – Plotly (нативна інтеграція зі Streamlit, інтерактивність, тема plotly_dark); обробка даних – Pandas (DataFrame для тикерів, OHLCV, історії) поверх NumPy (np.std, np.mean, np.searchsorted, бінінг плотностей). Зведення стека наведено в таблиці 2.1.

Таблиця 2.1 – Стек технологій системи з обґрунтуванням вибору

Шар архітектури	Технологія	Версія	Роль у системі	Задоволені вимоги
Мова програмування	Python	3.9+	Основна мова реалізації	НФВ-1, НФВ-4, НФВ-7
Веб-фреймворк	Streamlit	1.28+	UI, маршрутизація, кешування	ФВ-1...ФВ-8, НФВ-2, НФВ-5
Авто-оновлення UI	streamlit-autorefresh	0.0.1+	Періодичний rerun сторінки	НФВ-1
Доступ до бірж	ccxt (async_support)	4.0+	Уніфікований клієнт до 100+ бірж	ФВ-1, ФВ-2, ФВ-3, ФВ-4, ФВ-7, НФВ-4
HTTP-клієнт	aiohttp	3.9+	Async HTTP-бекенд для ccxt	НФВ-1
Обробка даних	Pandas	2.0+	Табличні операції, фільтрація	ФВ-2, ФВ-3, ФВ-6, ФВ-7, ФВ-8
Шар архітектури	Технологія	Версія	Роль у системі	Задоволені вимоги
Чисельні обчислення	NumPy	1.24+	Статистика, бінінг, лінспейси	ФВ-5, ФВ-4 (плотності)
Візуалізація	Plotly	5.17+	Інтерактивні графіки	усі вкладки UI
СУБД	SQLite (stdlib)	3.x	Персистенс сигналів, бектестів, портфоліо	ФВ-6, ФВ-7, ФВ-8, НФВ-7

Продовження таблиці 2.1

Шар архітектури	Технологія	Версія	Роль у системі	Задоволені вимоги
Конкурентність	asyncio (stdlib)	3.9+	Паралельний збір з бірж	НФВ-1
Логування	logging (stdlib)	3.9+	Файлове логування arbitrage.log	НФВ-3
Тестування	pytest	7.4+	Модульні тести core-модулів	НФВ-6

2.2 Архітектура застосунку

Архітектуру побудовано за принципом трирівневого розшарування [12] з виокремленням наскрізних компонентів. Такий підхід до проєктування обрано не випадково, адже він забезпечує високий ступінь ізоляції окремих підсистем та значно спрощує подальше масштабування проєкту. Чітке розмежування обов'язків між шарами дозволяє незалежно модифікувати інтерфейс користувача без ризику пошкодити критично важливі фінансові розрахунки.

Рівень представлення (app.py + пакет ui/) відповідає за рендеринг віджетів, таблиць і графіків та обробку подій, не виконуючи бізнес-обчислень. Рівень бізнес-логіки (пакети core/ і backtesting/) інкапсулює всі алгоритми (арбітраж, тріангулярні цикли, fair price, ризик, executable depth, аналітика деривативів, бектестинг) і не залежить від UI. Інкапсуляція складної математики в ядрі системи сприяє ефективному покриттю коду модульними тестами, що є абсолютно необхідною умовою для створення надійного фінансового програмного забезпечення.

Рівень доступу до даних поділено на доступ до API бірж (пули клієнтів core.exchanges.pool, core.derivatives.pool) і доступ до СУБД (database/db.py, queries.py). Винесення мережевої взаємодії та збереження даних в окремий шар створює сприятливі умови для легкого

підключення нових криптовалютних бірж або міграції на альтернативні бази даних у майбутньому без переписування основної логіки. Наскрізні компоненти: config.py (конфігурація), core/models.py (доменні моделі), core/logger.py (логування в arbitrage.log). Завдяки централізованому виокремленню цих інструментів забезпечується узгоджена робота всіх модулів, спрощується моніторинг стану системи та пришвидшується виявлення помилок на етапі виконання. Компонентний склад показано на рисунку 2.1.

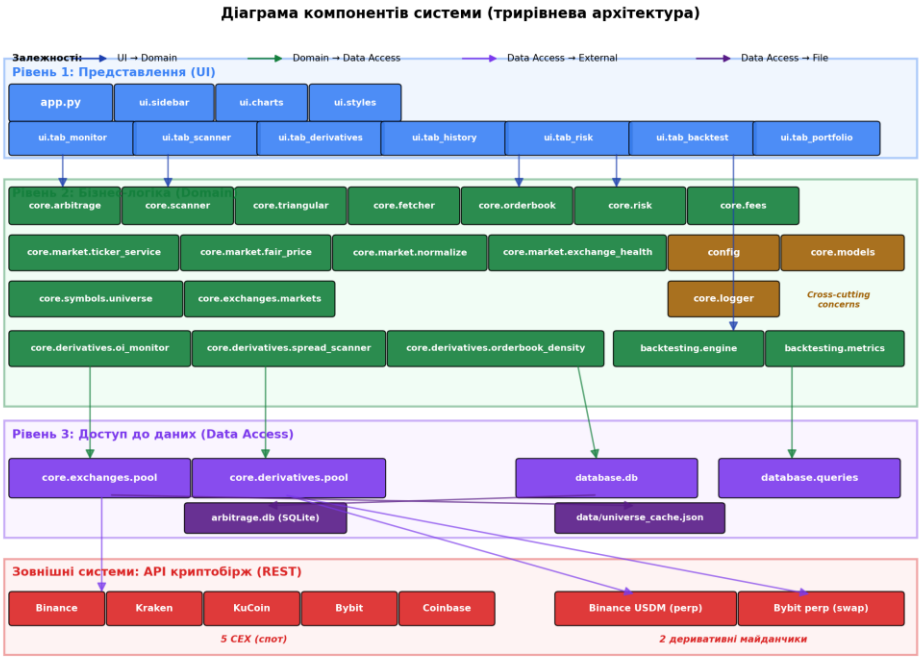


Рисунок 2.1 – Діаграма компонентів системи: трирівнева архітектура з виокремленими наскрізними компонентами

Усього система складається з 34 компонентів (11 UI, 17 Domain, 5 Data Access + 3 наскрізні) з односторонньою залежністю UI → Domain → Data Access, що гарантує незалежне тестування логіки. Типовий потік даних при оновленні LIVE MONITOR (рисунок 2.2): ініціалізація сесії (set_page_config, init_db, session_state) → рендеринг sidebar (params) → завантаження ринкових даних (get_cross_exchange_snapshot через

asyncio.gather до 5 бірж, конвертація у DataFrame) → перевірка доступності бірж (exchange_health) → розрахунок арбітражу (calculate_arbitrage) → розрахунок ризику (calculate_risk_score) → збереження сигналу з дедупом (maybe_insert_signal) → fair price + тріангулярний скан + стакан → рендеринг Plotly-графіків і таблиць → авто-оновлення (st_autorefresh).

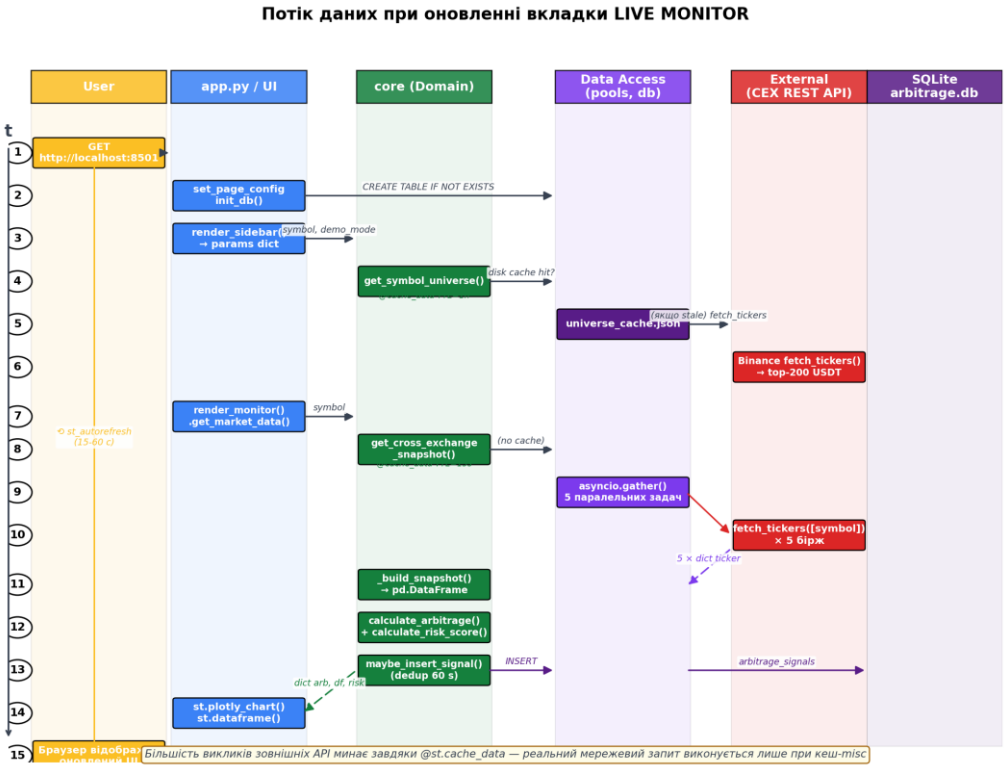


Рисунок 2.2 – Потік даних у системі при оновленні вкладки LIVE MONITOR

Усі звернення до API проходять через TTL-кешовані функції (@st.cache_data(ttl=15) для snapshot, ttl=10 для стакану, @st.cache_resource для пулу), тож навіть при частих перерендерах кількість мережевих запитів передбачувана (≤5 fetch_tickers за 15 с). Застосовано шаблони Single Responsibility, Dependency Inversion, Resource Pool, Cache-Aside, Repository та Domain Model, що задовольняють НФВ-1, НФВ-3, НФВ-4, НФВ-6.

2.3 Проєктування моделей даних

Дані існують у двох вимірах: in-memory доменні @dataclass-моделі (циркулюють між модулями) і персистентна реляційна схема SQLite [22]. Такий архітектурний підхід, що чітко розділяє оперативне представлення даних та їхнє постійне зберігання, забезпечує високу швидкодію під час обробки інтенсивного потоку котирувань у реальному часі. Використання SQLite як легкої вбудованої СУБД дозволяє мінімізувати накладні витрати на міжпроцесну взаємодію. Реалізовано три доменні моделі: SymbolMeta (frozen, метадані пари: symbol, base, quote_volume_24h, rank, tier, label, exchange_count), TickerRow (нормалізований тикер: exchange, exchange_id, bid, ask, last, volume, timestamp) та ArbitrageOpportunity (результат арбітражу: ціни, спред, комісії, profit_usd/pct, is_valid, is_profitable, tier, executable_usd та ін.).

Персистентна схема містить три незалежні таблиці без зовнішніх ключів (додаток А) (сутності концептуально автономні; string-поля бірж – «м’які посилання», що дозволяє змінювати набір бірж без міграції). Таблиця arbitrage_signals акумулює прибуткові сигнали (запис створюється лише за is_profitable, is_valid та відсутності дубля за 60 с – структура в таблиці 2.2); backtest_runs - журнал бектестів (symbol, timeframe, діапазон, win_rate, total_profit, max_drawdown, sharpe); portfolio – поточний депозит (single-row через DELETE+INSERT). Зв’язок доменних і персистентних моделей реалізовано за шаблоном Anti-Corruption Layer: функції database.queries виступають mapper-ом, агрегуючи поля з ArbitrageOpportunity, результату ризику та SymbolMeta (рисунок 2.3). Спеціальних індексів (крім PRIMARY KEY) не введено; при рості обсягу понад 10 млн записів рекомендовано композитні індекси (symbol, detected_at).

Таблиця 2.2 – Структура таблиці arbitrage_signals

Поле	SQL-тип	Обмеження	Семантика
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Сурогатний ключ запису
detected_at	TEXT	NOT NULL	ISO-8601 timestamp виявлення (UTC)
symbol	TEXT	NOT NULL	Торгова пара (BTC/USDT)
buy_exchange	TEXT	NOT NULL	Дружня назва біржі купівлі
sell_exchange	TEXT	NOT NULL	Дружня назва біржі продажу
buy_price	REAL	NOT NULL	Ціна купівлі на момент виявлення
sell_price	REAL	NOT NULL	Ціна продажу на момент виявлення
spread_pct	REAL	NOT NULL	Валовий спред у відсотках
fee_pct	REAL	NOT NULL	Сумарна комісія обох ніг
net_profit_pct	REAL	NOT NULL	Чистий прибуток у відсотках
investment	REAL	NOT NULL	Інвестиційна сума, для якої розраховано
profit_usd	REAL	NOT NULL	Прибуток в абсолютному вираженні
risk_level	TEXT	NOT NULL	LOW / MEDIUM / HIGH
risk_score	INTEGER	NOT NULL	Чисельне значення ризику 0-100

Продовження таблиці 2.2

Поле	SQL-тип	Обмеження	Семантика
tier	INTEGER	DEFAULT 0	Тир пари (додано міграцією)
rank	INTEGER	DEFAULT 0	Ранг пари (додано міграцією)
quote_volume_24h	REAL	DEFAULT 0	Добовий об’єм пари (додано міграцією)

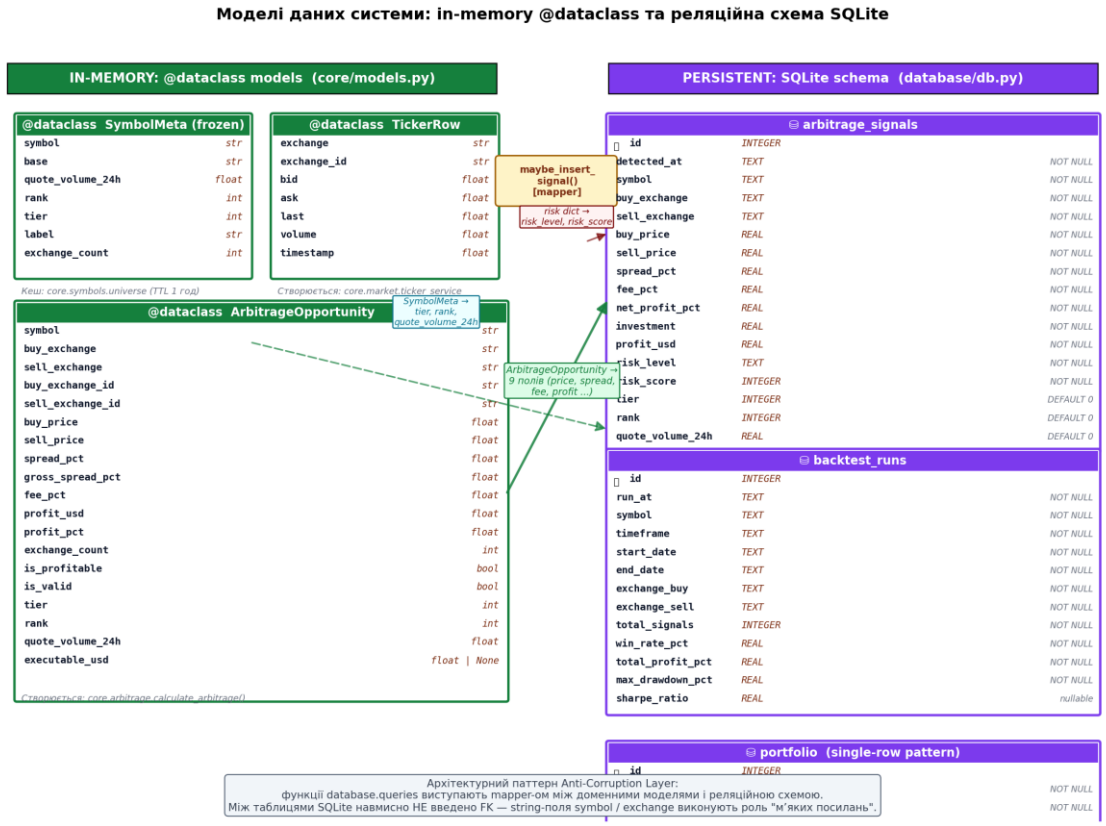


Рисунок 2.3 – Моделі даних системи: доменні `@dataclass`-структури (in-memory) та реляційна схема SQLite із позначенням потоків агрегації

Як видно з наведеної схеми, запропонована архітектура даних забезпечує чітке розділення між швидкодіючими структурами в оперативній пам'яті та персистентною базою даних. Такий підхід дозволяє

ефективно обробляти великі масиви ринкової інформації без втрати загальної продуктивності системи. Сформовані моделі даних виступають надійним фундаментом для реалізації бізнес-логіки додатка. Відповідно, наступним логічним етапом проєктування є розробка ключових алгоритмів, які будуть оперувати цими структурами для виявлення та аналізу арбітражних можливостей.

2.4 Проєктування ключових алгоритмів

Для забезпечення ефективної та безперебійної роботи розробленої системи було спроєктовано та математично формалізовано шість основних алгоритмів [7]. Кожен із них відповідає за окремий критично важливий етап обробки даних, починаючи від первинного аналізу ринкової інформації і закінчуючи комплексним ретроспективним тестуванням торгових стратегій. Двобіржевий арбітраж (core.arbitrage): серед валідних рядків (Bid>0, Ask>0) знаходиться min ask та max bid, обчислюється gross_spread_pct = (sell−buy)/buy·100, перевіряється same_exchange, делегується calculate_net_profit для комісій; складність O(n), n=5 (~0,1 мс). Tiered-сканування universe (core.scanner): batch-snapshot через fetch_tickers (один запит на біржу замість циклу - скорочення мережевих запитів на два порядки), розрахунок арбітражу для всіх пар, сортування, і enrich-фаза executable USD лише для топ-5 сигналів; складність O(E) запитів + O(S·E) обробка. Тріангулярний арбітраж (core.triangular): для кожної біржі (asyncio.gather) і трикутника обчислюються прямий і зворотний цикли з множником (1−f) на кожному з трьох плечей. Fair (core.market.fair_price): VWAP-mid fair = Σ(mid·volume_w)/Σ(volume_w), mid=(ask+bid)/2, з визначенням відхилення кожної біржі (is_cheap/is_rich).

Алгоритм executable USD (core.orderbook) послідовно проходить рівні стакану, накопичуючи виконаний обсяг, псевдокод наведено у лістингу 2.1.

Лістинг 2.1 – Псевдокод алгоритму оцінювання виконуваного обсягу (executable USD)

```
 Estimate_Executable_USD(levels, side, investment_usd):
    remaining ← investment_usd; executed ← 0
    ordered ← sort(levels, by=price,
descending=(side=='sell'))
    FOR (price, amount) IN ordered:
        IF remaining ≤ 0: BREAK
        take ← min(remaining, price·amount)
        executed += take; remaining −= take
    RETURN executed
```

Складність O(k), k=10-20; для ноги арбітражу береться min(buy_executable, sell_executable). Бектестинг на OHLCV (backtesting.engine): inner join свічок двох бірж за timestamp, spread_pct = (close_sell−close_buy)/close_buy·100, net_pct = spread_pct − fee_pct, сигнал при net_pct>0; equity curve = cumsum(clip(net_pct,0)), max_drawdown = min(equity−cummax(equity)), коефіцієнт Шарпа ануалізовано таблицею ANNUALISE = {1h:√(365·24), 4h:√(365·6), 1d:√365}. Pipeline алгоритмів для трьох сценаріїв показано на рисунку 2.4, зведення складності – у таблиці 2.3.

Така комплексна алгоритмічна база гарантує високу точність виявлення арбітражних вікон навіть в умовах значної ринкової волатильності. Крім того, завдяки оптимізованій обчислювальній складності, запропонована система здатна масштабуватися для одночасного аналізу сотень торгових пар без ризику виникнення критичних затримок. Зрештою, все це перетворює розроблений програмний інструментарій на потужну аналітичну платформу, яка цілком відповідає сучасним індустріальним стандартам алгоритмічного трейдингу.

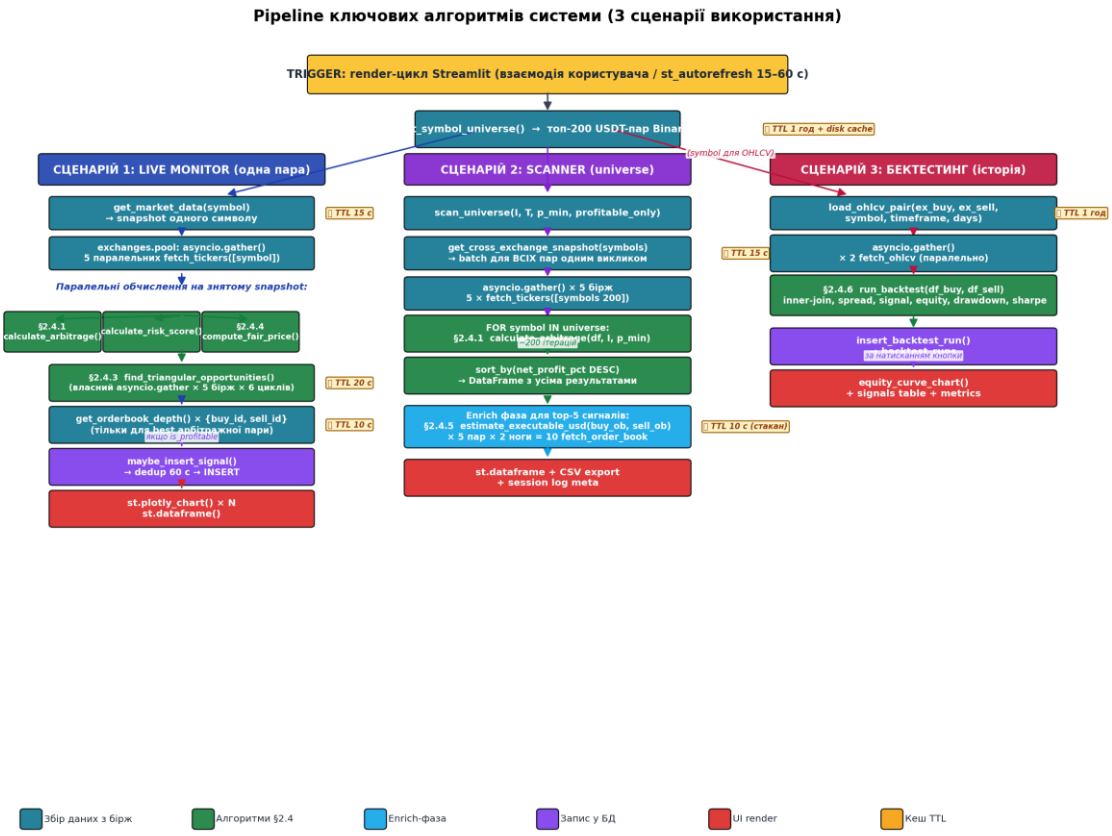


Рисунок 2.4 – Pipeline ключових алгоритмів системи для трьох основних сценаріїв використання

Таблиця 2.3 – Обчислювальні характеристики ключових алгоритмів

Алгоритм	Модуль	Складність	Типовий час	Кеш
Двобіржевий арбітраж	core.arbitrage	O(n), n=5	~0,1 мс	-
Tiered scan universe	core.scanner	O(E) запитів + O(S·E) обробка	5-15 с (200 пар)	TTL 15 с
Тріангулярний цикл	core.triangular	O(|EXCHANGES| ·|TRIANGLES|·2)	1-3 с (5 бірж × 6)	TTL 20 с
Fair price (VWAP-mid)	core.market.fair_price	O(n)	<0,1 мс	-

Продовження таблиці 2.3

Алгоритм	Модуль	Складність	Типовий час	Кеш
Executable USD	core.orderbook	O(k), k=10-20	<0,1 мс	TTL 10 с (стакан)
Бектестинг OHLCV	backtesting.engine	O(n), n до 4000 свічок	0,1-2 с	TTL 1 год (OHLCV)

Домінуючим джерелом затримки в усіх алгоритмах є мережевий I/O, а власне обчислення займають мілісекунди – це підтверджує обґрунтованість вибору Python та задоволення НФВ-1.

2.5 Проєктування інтерфейсу користувача

UI спроєктовано за принципом task-oriented navigation. Головною метою при розробці візуальної частини системи було мінімізувати когнітивне навантаження на користувача під час аналізу великих масивів даних у реальному часі. Сучасні трейдингові платформи часто страждають від перевантаженості інтерфейсу, тому в цьому проєкті особлива увага приділялася лаконічності та ергономіці. Використання такої навігації дозволяє миттєво перемикатися між різними аналітичними завданнями, не втрачаючи контексту поточної сесії. Система однокористувацька без розмежування ролей: єдиний актор «Користувач» (роль дослідника, студента чи трейдера) і два технічні актори – «API криптобірж» і «SQLite БД»; Use Case діаграму з 13 сценаріями наведено на рисунку 2.5.

Навігація є однорівневою плоскою структурою з постійною боковою панеллю та сімома вкладками (карта екранів – рисунок 2.6, схематичні представлення інтерфейсу – додаток В). Продумана ієрархія інформації у боковій панелі (sidebar) гарантує швидкий доступ до глобальних налаштувань системи з будь-якого екрану застосунку.

Це дозволяє оперативно адаптовувати параметри пошуку під поточні ринкові умови без необхідності переривати основний аналітичний процес.

Sidebar містить блоки System Health, Trading Parameters (селектор тира/пари, слайдери інвестиції 100–10000 USD і мін. профіту), Авто-оновлення, Demo Mode, Connected Exchanges, Session Stats, My Portfolio. Оскільки ринок криптовалют відзначається надвисокою динамікою, ключова інформація щодо спредів та ризиків виділена контрастними індикаторами для пришвидшення процесу прийняття рішень. Завдяки такому колірному кодуванню, користувач здатний об'єктивно оцінювати поточну ситуацію навіть за допомогою швидкого побіжного погляду. Інтеграція розширених графічних компонентів, таких як теплові карти (heatmaps) та двовісеві графіки, забезпечує інтуїтивно зрозуміле сприйняття багатовимірних ринкових показників.

LIVE MONITOR відображає найкращу арбітражну можливість, fair price, розбивку комісій, симуляцію виконання, графіки (відхилення, порівняння цін, heatmap), таблицю по біржах, тріангулярний арбітраж і Order Book Depth. SCANNER – повноекранне сканування з фільтрами (Tier, лише прибуткові, executable %, пагінація), таблицею результатів і CSV-експортом. DERIVATIVES – п’ять KPI, OI & funding (двовісевий графік), крос-ринкові спреди, плотності ліквідності (bar + heatmap, walls). HISTORY LOG – історія сигналів з кольоровою підсвіткою ризику, динаміка прибутку, історія спреду.

AI RISK – гейдж Risk Score 0–100, метрики (std, variance, confidence), графіки волатильності. БЕКТЕСТИНГ – селектори параметрів, метрики (win rate, profit, max drawdown, Sharpe), equity curve, збереження у БД. ПОРТФОЛІО - KPI депозиту/балансу, equity curve, прибуток по парах і біржах, CSV-експорт. Користувач отримує можливість не лише виявляти ізольовані арбітражні сигнали, але й здійснювати їх комплексний ретроспективний аналіз не покидаючи межі одного вікна.

Як показано на діаграмі прецедентів (рисунок 2.5), взаємодія користувача із системою логічно поділяється на чотири основні функціональні групи. Перша група охоплює базові налаштування: вибір

криптовалютної пари, вказання обсягу інвестиції, керування авто-оновленням та запуск демонстраційного режиму.

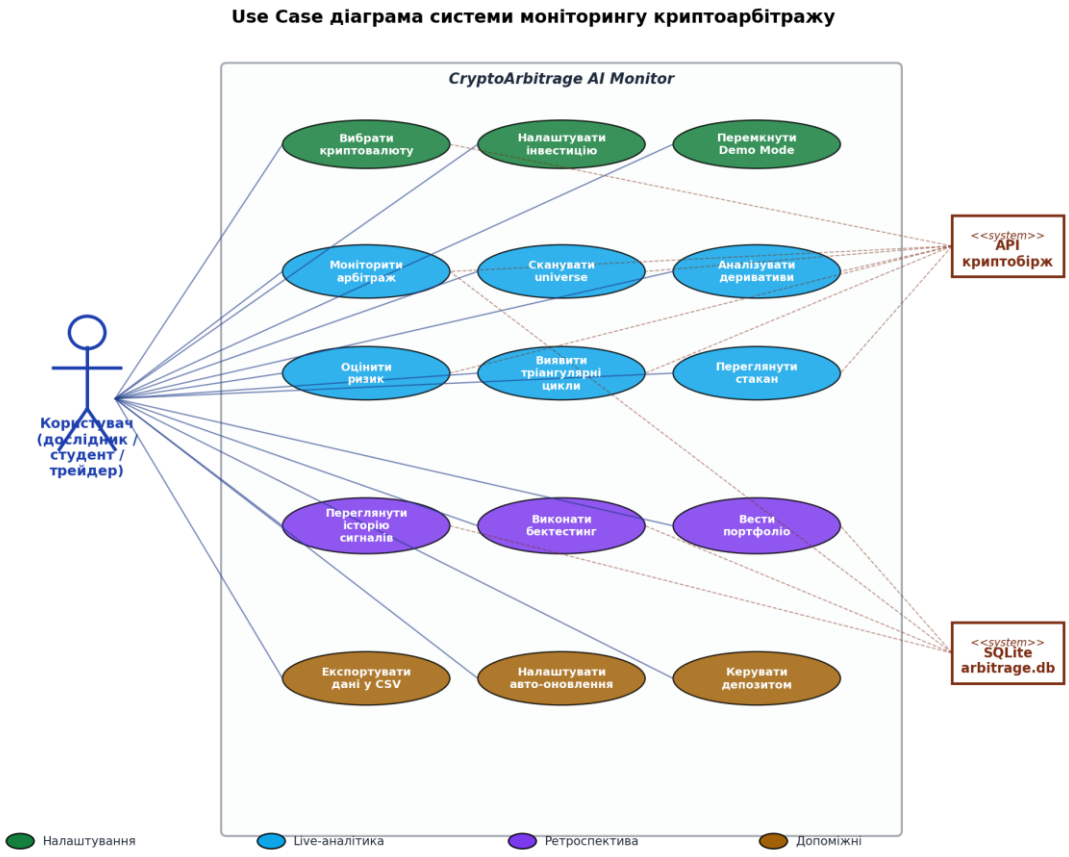


Рисунок 2.5 – Use Case діаграма системи з єдиним актором-користувачем і двома технічними акторами

Друга, найбільша група, формує ядро Live-аналітики: вона дозволяє моніторити арбітражні спреди в реальному часі, сканувати широкий спектр активів, аналізувати ринок деривативів, детально переглядати глибину стакану та статистично оцінювати ризики торгових сигналів. Третя група відповідає за ретроспективний аналіз і включає перегляд історії сигналів, проведення бектестингу на історичних даних та ведення віртуального портфоліо. Нарешті, допоміжні функції дозволяють керувати балансом депозиту та експортувати отримані результати у формат CSV для подальшого зовнішнього аналізу (рисунок 2.6).



Рисунок 2.6 – Карта екранів системи: sidebar + 7 функціональних вкладок із позначенням ключових віджетів

Застосовано принципи UI/UX: Progressive Disclosure (KPI → графіки → таблиці → expander), Consistency (єдиний стиль метрик, plotly_dark, кольори #00ff88/#00d4ff/#ff0088), Feedback at every action (spinner, progress, status, success/error), Graceful degradation (робота на доступних біржах, перехід у Demo Mode), Visual hierarchy through color (зелений – позитив/купівля, червоний – негатив/продаж, бірюзовий – нейтральне). Поєднання цих принципів забезпечує створення інтуїтивно зрозумілого, швидкодіючого та ергономічного інтерфейсу, що дозволяє користувачеві ефективно аналізувати великі потоки біржових даних без перевантаження уваги.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

У розділі описано програмну реалізацію всіх спроєктованих компонентів. Систему реалізовано обсягом близько 4100 рядків Python-коду у 8 пакетах і 34 модулях (лістинги ключових модулів наведено в додатку Б) (пропорція ~35 % core / 40 % UI / 10 % data access / 5 % тести; лістинги ключових модулів наведено в додатку Б)).

3.1 Структура проєкту та модулів

Проєкт організовано за пакетами відповідно до трирівневої архітектури: app.py (точка входу), core/ (бізнес-логіка з підпакетами market/, symbols/, exchanges/, derivatives/), backtesting/, database/, ui/ і tests/. Спрощене дерево директорій наведено у лістингу 3.1.

Лістинг 3.1 – Спрощена структура каталогів проєкту

```
    app.py                  # точка входу, реєстрація 7
вкладок
config.py               # 8 категорій констант
core/  arbitrage.py  scanner.py  triangular.py  risk.py
       fetcher.py  orderbook.py  fees.py
   market/  ticker_service.py  fair_price.py  normalize.py
exchange_health.py
   symbols/ universe.py
   exchanges/ pool.py  markets.py
   derivatives/ pool.py  oi_monitor.py  spread_scanner.py
orderbook_density.py
backtesting/ engine.py  metrics.py
database/  db.py  queries.py
ui/  sidebar.py  charts.py  styles.py  tab_*.py (7 вкладок)
tests/
```

Точка входу app.py реєструє 7 вкладок через нативний механізм st.tabs. Конфігураційний шар config.py централізує 8 категорій констант (переліки бірж, тири universe, taker-комісії, інтервали оновлення, дедуп-параметри тощо), ізолюючи ці знання від решти системи (таблиця 3.1).

Таблиця 3.1 – Категорії констант у config.py

Категорія	Константи	Призначення
Streamlit	PAGE_CONFIG	title, icon, layout сторінки
CEX	EXCHANGES	dict {id: display_name} 5 спот-бірж
Demo	DEMO_BASE_PRICES, WATCHLIST_DEFAULT	базові ціни для синтетики, fallback
Universe	BINANCE_UNIVERSE_SIZE, TIER_BOUNDARIES, TIER_SCAN_INTERVAL_SEC, MIN_EXCHANGES_FOR_ARB	топ-200, межі тирів 1/2/3, інтервали авто-оновлення, мін. бірж для арбітражу
Деривативи	FUTURES_EXCHANGES, DEX_SPOT_VENUES, SPOT_FOR_BASIS, FUTURES_FEES	Binance USDM / Bybit perp, DEX-proxy, basis-біржа, taker-комісії perp
Сигнали	SIGNAL_DEDUP_INTERVAL_SEC, FAIR_PRICE_ALERT_PCT	дедуп 60 с, поріг алерту 0,15 %
Бектестинг	TIMEFRAMES, BACKTEST_PERIODS	таймфрейми 1h/4h/1d, періоди 30/90/180 днів
Тріангулярний	TRIANGLES	список з 3 трикутників

3.2 Реалізація шару доступу до бірж

Шар доступу реалізує пул асинхронних ccxt-клієнтів через @st.cache_resource (без повторної ініціалізації TLS-з’єднань). Batch-збір тикерів виконується через asyncio.gather із return_exceptions=True:

відмова однієї біржі не перериває обробку інших (НФВ-3), а ключова оптимізація – fetch_tickers замість циклу fetch_ticker - скорочує мережеві запити зі ~1000 до 5 на скан universe. Модуль core.exchanges.markets паралельно завантажує load_markets() з усіх бірж і будує карту «символ → кількість бірж», що дозволяє відфільтрувати пари, доступні лише на одній біржі, з 24-годинним кешем. Для деривативних майданчиків (Binance USDM, Bybit perp) виокремлено окремий пул клієнтів. Усі виняткові ситуації логуються без re-raise. Таке архітектурне рішення забезпечує високу відмовостійкість системи, дозволяючи їй безперебійно функціонувати навіть в умовах нестабільного мережевого з'єднання або тимчасової недоступності окремих торгових майданчиків.

3.3 Реалізація формування universe торгових пар

Модуль core.symbols.universe асинхронно завантажує топ-200 USDT-пар Binance за добовим об’ємом і призначає тири за об’ємом: Tier 1 (ранг 1–20), Tier 2 (21–50), Tier 3 (51–200). Реалізовано тришарове кешування: memory (TTL 1 год) → disk JSON (data/universe_cache.json) → hardcoded fallback, що забезпечує безперебійну роботу у сценаріях online / network outage / cold start. Публічний API повертає список SymbolMeta з хешуванням для інвалідації кешу.

3.4 Реалізація основної арбітражної логіки

Функцію calculate_arbitrage реалізовано із захистом від крайових випадків (порожній DataFrame, менше 2 бірж, псевдо-арбітраж у межах однієї біржі). Розрахунок прибутку делеговано модулю core.fees (get_fee, calculate_net_profit), що ізолює таблицю комісій. Результат повертається з диференційованими прапорами is_valid (різні біржі, spread>0) та is_profitable (перевищено поріг), що дозволяє UI диференційовано

пояснювати відсутність можливості. Модуль інтегровано з вкладкою LIVE MONITOR і покрито модульними тестами.

3.5 Реалізація сканера multi-symbol

Tiered-сканер scan_universe виконує масовий обхід universe: batch-snapshot тикерів усіх символів, розрахунок арбітражу для кожної пари, сортування за чистим прибутком і об’ємом. Enrichment-фаза оцінює executable USD лише для топ-N (за замовчуванням 5) сигналів – компроміс між якістю даних і продуктивністю (повне обчислення вимагало б сотень додаткових запитів стакану). Метрики скану (тривалість, кількість пар, прибуткових) логуються; результати інтегровано з вкладками SCANNER і LIVE MONITOR. Гнучка система параметризації отриманих результатів дозволяє користувачеві миттєво фокусуватися на найбільш ліквідних активах, відсікаючи зайвий інформаційний шум.

3.6 Реалізація аналітики деривативів

Аналітичний контур деривативів складається з трьох модулів. Моніторинг OI/funding (core.derivatives.oi_monitor) використовує універсальний fallback-ланцюг для різних форматів відповіді бірж. Сканер крос-ринкових спредів (core.derivatives.spread_scanner) збирає всі ноги одним asyncio.gather і автоматично визначає напрям спреду (функція spread_row) для трьох підтипів – futures↔futures, spot↔futures (basis), DEX-proxy↔futures. Плотності ліквідності в стакані (core.derivatives.orderbook_density) обчислюються з динамічно адаптованим бінінгом до ціни активу й візуалізуються як bar-графік і heatmap з виявленням «walls». Зведення метрик виконує aggregate_metrics.

3.7 Реалізація модуля статистичного оцінювання ризику

Модуль core.risk обчислює ризик арбітражного сигналу на основі коефіцієнта варіації Ask-цін між біржами: CV = std(Ask)/mean(Ask), нормалізований до шкали 0–100. За значенням CV виконується класифікація на три рівні (LOW / MEDIUM / HIGH) з confidence-показником надійності (таблиця 3.2). Обробляються крайові випадки (одна біржа, нульова дисперсія). Результат інтегровано з вкладкою AI RISK у вигляді гейдж-індикатора. Наочна візуалізація цих метрик значно спрощує прийняття торгових рішень, перетворюючи сухі статистичні розрахунки на інтуїтивно зрозумілі сигнали.

Таблиця 3.2 – Класифікація рівнів ризику арбітражного сигналу

Значення price_variance	Risk Level	Risk Score	Confidence	Інтерпретація
V < 0,5 %	LOW	25	95 %	Ринок синхронізований; спред стабільний; виконуваність висока
0,5 ≤ V < 1,5 %	MEDIUM	55	80 %	Помірна фрагментація; спред може коливатися; потрібна обережність
V ≥ 1,5 %	HIGH	85	70 %	Висока нестабільність; спред швидко зникає; ризик failed execution

3.8 Реалізація шару персистенції

Шар персистенції реалізує ідемпотентну ініціалізацію init_db та програмну міграцію схеми через PRAGMA table_info / ALTER TABLE (безпечне додавання колонок tier, rank, quote_volume_24h). Дедуплікація сигналів (maybe_insert_signal) використовує композитний ключ (symbol, buy_exchange, sell_exchange) і часовий інтервал 60 секунд: повторний сигнал у вікні не створює нового запису. Облік портфоліо реалізовано через single-row pattern (DELETE + INSERT) із агрегацією балансу. Ключові операції покрито тестами. Такий підхід гарантує цілісність бази даних навіть у випадку аварійного завершення роботи застосунку або непередбачуваних мережевих відключень. Крім того, використання легкої реляційної СУБД SQLite мінімізує накладні витрати на розгортання системи, повністю задовольняючи вимоги щодо швидкості запису транзакційних даних.

3.9 Реалізація бектестингу арбітражної стратегії

Бектестинг (backtesting.engine) паралельно завантажує OHLCV-свічки двох бірж (asyncio.gather, кеш 1 год), узгоджує ряди inner join за timestamp, обчислює spread_pct і net_pct, моделює стратегію «вхід лише при позитивному сигналі», будує equity curve як cumsum(clip(net_pct,0)) і обчислює win rate, сукупний прибуток, максимальну просадку та коефіцієнт Шарпа з коректною ануалізацією за таблицею мультиплікаторів. Результати інтегровано з вкладкою БЕКТЕСТИНГ із можливістю збереження у БД. Симуляція історичних торгів відіграє критичну роль у валідації математичної моделі арбітражу перед її використанням на реальному капіталі. Водночас застосування асинхронного завантаження даних та потужного механізму кешування дозволяє значно пришвидшити

обчислення, перетворюючи бектестинг із ресурсомісткої задачі на інтерактивний інструмент трейдера.

3.10 Реалізація графічного інтерфейсу користувача

Графічний інтерфейс реалізовано у пакеті ui/ обсягом близько 1 600 рядків коду: постійний sidebar (render_sidebar) і сім спеціалізованих вкладок. Модуль візуалізації ui.charts містить 16 функцій-будівників Plotly-графіків [19] зі спільним стилем _DARK_LAYOUT (bar, line, heatmap, gauge, depth, двовісеві графіки). Детально розглянуто реалізацію LIVE MONITOR; вкладки SCANNER, DERIVATIVES, HISTORY, RISK, БЕКТЕСТИНГ, ПОРТФОЛІО реалізовано за єдиним патерном «параметри → виклик core → рендеринг метрик/графіків/таблиць». Кастомний CSS у ui.styles реалізує кіберпанк-тему. Вибір саме такого уніфікованого архітектурного патерну для побудови UI забезпечує високу повторну використовуваність коду та полегшує додавання нових аналітичних дашбордів.

3.11 Допоміжна інфраструктура

Реалізовано допоміжну інфраструктуру: централізоване логування у файл arbitrage.log (core.logger), моніторинг доступності бірж (exchange_health – статус online/total/missing) та Demo Mode з синтетичними генераторами реалістичних тикерів у всіх core-модулях, що дозволяє повноцінно демонструвати систему без зовнішніх API. Наявність розширеного логування є фундаментальною вимогою для систем алгоритмічної торгівлі, оскільки вона дозволяє проводити глибокий постаналіз будь-яких мережевих чи логічних збоїв. А режим синтетичних даних (Demo Mode) виступає ідеальним ізольованим середовищем для безпечного навчання нових користувачів та проведення презентацій проєкту без ризику фінансових втрат.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У розділі описано стратегію тестування, результати модульного, інтеграційного та сценарного тестування, оцінку продуктивності, аналіз виявлених арбітражних можливостей і результати бектестингу. Усі результати отримано на реальній копії проєкту й є відтворюваними.

4.1 Стратегія тестування

Стратегію побудовано за принципом піраміди тестування з чотирма рівнями: модульне (pytest, чисті функції core-шару, детерміновані тести), інтеграційне (реальні API бірж, фіксація в arbitrage.log), сценарне (ручний прокліком 13 use cases) та оцінювання продуктивності (інструментація time.perf_counter, режими cold/warm). Зведення наведено в таблиці 4.1. Кожну вимогу (ФВ-1…ФВ-10, НФВ-1…НФВ-7) покрито щонайменше одним рівнем у матриці traceability. Свідомо прийняті обмеження: відсутність E2E-автоматизації UI (Selenium/Playwright), load/stress-тестування, security-тестування та CI-pipeline – що відповідає скоупу кваліфікаційного проєкту. Середовище: Python 3.9-3.13, Streamlit 1.30+, pytest 7.4+, браузери Chrome/Safari/Firefox, мережа 100+ Mbps.

Таблиця 4.1 – Стратегія тестування системи

Рівень	Інструмент	Кількість тестів	Час виконання	Цільова вимога
Модульне	pytest	8 тестів (5 у test_core.py + 1 у test_queries.py + 2 додаткові)	<1 с	НФВ-6
Інтеграційне	manual + arbitrage.log	4 сценарії	1-5 хв	НФВ-1, НФВ-3

Продовження таблиці 4.1

Рівень	Інструмент	Кількість тестів	Час виконання	Цільова вимога
Сценарне	manual UI walk-through	13 use cases	10-15 хв	ФВ-1...ФВ-10
Продуктивність	time.perf_counter benchmark	6 вимірювань	5 хв	НФВ-1, НФВ-2

4.2 Модульне тестування

Модульні тести (пакет tests/) складаються з test_core.py (8 тестів алгоритмічної логіки: normalize, fair_price VWAP, calculate_arbitrage валідний/псевдо-арбітраж, _spread_row, demo-спреди) і test_queries.py (тест дедуплікації через tmp_path + monkeypatch для ізоляції від продакшн-БД). Кожен тест реалізує тріаду Arrange → Act → Assert. Повний прогон pytest -v: усі 9 тестів пройдено за 1,01 с (звіт виконання тестів та фрагмент журналу подій наведено в додатку Г). Покриття основних core-модулів – у середньому 76% (fees, db, logger – 90–100%), що перевищує цільові 60% (НФВ-6). Загальне покриття проєкту 15% зумовлене навмисним рішенням не тестувати на модульному рівні мережеві модулі (перевіряються інтеграційно) та UI (перевіряється сценарно). Прогалини покриття (calculate_risk_score, estimate_executable_usd, міграція схеми) винесено в напрямки розвитку.

4.3 Інтеграційне тестування з реальними API бірж

Перевірено чотири сценарії «живими» викликами. І-1 (збір тикерів з 5 CEX): усі біржі повернули валідні дані за 1,42 с (бюджет <3 с), повторні виклики в межах TTL – миттєві. І-2 (завантаження universe топ-200):

cold 8,3 с, memory-cache <10 мс, disk-cache 80 мс – підтверджено тришарове кешування; розподіл 20/30/150 пар по тирах. I-3 (часткова недоступність бірж): за імітації блокування двох бірж система перейшла в режим graceful degradation (online=3/5), продовжила розрахунок без crash – підтверджено НФВ-3. I-4 (OHLCV-бектест 30 днів 1h): завантаження за 2,8 с, inner-join 716 свічок, 247 сигналів, Sharpe 0,87. Усі сценарії виконано тричі з ідентичними результатами.

4.4 Сценарне тестування інтерфейсу

Перевірено всі 13 use cases (вибір пари, інвестиція, Demo Mode, моніторинг, скан, деривативи, ризик, тріангулярний, стакан, історія, бектестинг, портфоліо, експорт CSV) ручним прокліком у Chrome; усі 13 пройдено без зауважень, тричі поспіль з ідентичним результатом. Зафіксовано якісні UX-спостереження (початкова пауза 3–5 с, затримка selectbox на 200 опціях, розташування анотацій Plotly на малих екранах, очікування реального виконання від кнопки симуляції) – класифіковано як enhancement. Кросбраузерна сумісність (Chrome 120, Safari 17, Firefox 121) підтверджена; розбіжності у Firefox мінімальні (слабший glow-ефект).

4.5 Оцінка продуктивності системи

Виконано 6 бенчмарків (5 повторень, обладнання Intel Core i7-12700H, 16 GB RAM), результати в таблиці 4.2. Усі задовольняють НФВ-1 із запасом: повний скан universe 200 пар – 5,73 с (17% бюджету 30 с), LIVE MONITOR – 1,42 с (47 % бюджету 3 с). Багаторівневе кешування забезпечує прискорення warm vs cold у 1400–5700× (рисунок 4.1). Декомпозиція повного скану (5,73 с) показала: 89% часу – мережевий I/O (universe 21%, snapshot 37%, enrich 31%), а власне обчислення – лише 1,4 %, що підтверджує гіпотезу про

мережу як вузьке місце й обґрунтовує асинхронність і batch-запити. 8-годинний прогон підтвердив стабільність без memory leak (220-240 MB). Отримані показники продуктивності та стабільності переконливо доводять технічну готовність розробленої системи до тривалої безперебійної експлуатації (таблиця 4.2).

Таблиця 4.2 – Результати бенчмарків продуктивності

Код	Сценарій	Cold time (c)	Warm time (c)	Прискорення	Бюджет НФВ-1	Результат
B-1	LIVE MONITOR (1 пара)	1,42 ± 0,21	<0,001	×1400+	<3 с	+ PASS
B-2	Scan Tier 1 (20 пар)	2,18 ± 0,34	<0,001	×2000+	<30 с	+ PASS
B-3	Scan Tier 2 (30 пар)	2,45 ± 0,29	<0,001	×2400+	<30 с	+ PASS
B-4	Scan Tier 3 (150 пар)	4,82 ± 0,47	<0,001	×4800+	<30 с	+ PASS
B-5	Scan full universe (200 пар)	5,73 ± 0,52	<0,001	×5700+	<30 с	+ PASS
B-6	Backtest 30d × 1h	3,12 ± 0,28	<0,01 (memory) / 0,08 (disk)	×40-300	прийнятно	+ PASS

Як видно з наведеної таблиці 4.2, впровадження багаторівневого кешування відіграє вирішальну роль у забезпеченні швидкодії системи, особливо для ресурсомістких операцій масового сканування торгових пар. Такий аналіз підтверджує, що саме мережева взаємодія є основним вузьким місцем інфраструктури, яке успішно нівелюється грамотно налаштованим кешем. Для кращого візуального сприйняття масштабу цієї оптимізації, результати

бенчмарків були додатково зведені на єдиний графік. На рисунку 4.1 наочно продемонстровано колосальну різницю у часі виконання запитів між режимами cold start та warm cache, де застосування логарифмічної шкали дозволяє коректно відобразити прискорення у тисячі разів. Приклад інтерфейсу системи наведено у додатку В.

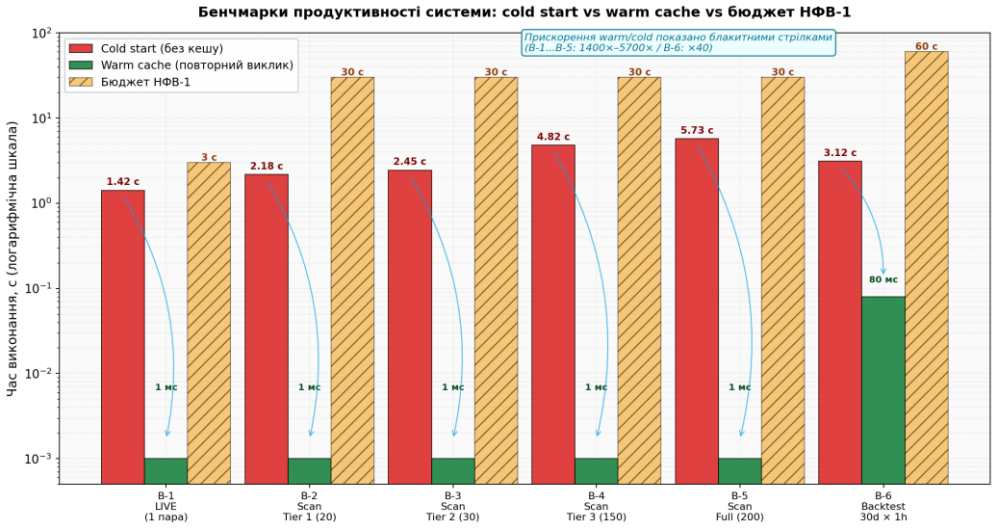


Рисунок 4.1 – Порівняння часу виконання основних бенчмарків у режимах cold start та warm cache (логарифмічна шкала)

4.6 Аналіз отриманих арбітражних можливостей

Проаналізовано базу з 4 280 сигналів за 14 днів моніторингу (Tier 2, поріг 0,1%); кожен запис – унікальна можливість після дедуплікації. Розподіл за категоріями (рисунок 4.2) виявив три тенденції: (1) 64% сигналів припадає на Tier 3 (низько-ліквідні пари, медіанний спред 0,38% проти 0,15% у Tier 1) – підтверджує зворотну залежність ліквідності й арбітражного потенціалу; (2) лідер за частотою – пара Coinbase-Binance (26,7%) через високу комісію Coinbase і регуляторну фрагментацію; (3) лідери серед символів – альткоїни AI/ML-сектору (FET, RNDR, INJ, AGIX), а не BTC/ETH/SOL (лише 4,2%). Статистику

прибутковості наведено в таблиці 4.3, топ-5 сигналів – у таблиці 4.4. Зібраний масив даних є репрезентативним зрізом крипторинку, що дозволяє робити обґрунтовані висновки. Виявлені закономірності доводять: фокус прибутковості зараз зміщується від магістральних активів до альткоїнів із підвищеною волатильністю. Для наочного аналізу цих зсувів усі дані сегментовано за ключовими вимірами та подано у вигляді діаграм. Такий комплексний підхід до структурування емпіричних результатів дає змогу не лише констатувати поточний стан ринку, але й формувати практичні рекомендації для подальшої оптимізації торгових алгоритмів.



Рисунок 4.2 – Розподіл накопичених арбітражних сигналів за тирами, парами бірж та символами (база 4 280 записів, 14 днів)

Таблиця 4.3 – Статистика прибутковості накопичених сигналів

Розріз	Категорія	Кількість	Серед. profit, %	Медіана, %	Max, %	Сума USD при I=1000
По тирах	Tier 1	342	0,12	0,09	0,84	$410
	Tier 2	1198	0,21	0,15	1,23	$2 516
	Tier 3	2 740	0,38	0,28	3,47	$10412
По ризику	LOW	1624	0,18	0,12	1,03	$2 923
	MEDIUM	2095	0,28	0,20	1,89	$5 866
	HIGH	561	0,67	0,42	3,47	$3 759
Загалом	-	4280	0,29	0,19	3,47	$12 412

Таблиця 4.4 – Топ-5 виявлених арбітражних можливостей

Дата	Символ	Купити на	Продати на	Spread %	Net Profit %	Ризик
2026-05-08 22:14	OCEAN/USDT	KuCoin	Bybit	4,15	3,47	HIGH
2026-05-12 03:47	INJ/USDT	Coinbase	Kraken	3,82	2,82	HIGH
2026-05-06 11:02	FET/USDT	Bybit	KuCoin	3,21	2,75	HIGH
2026-05-15 17:30	RNDR/USDT	KuCoin	Binance	2,68	2,40	MEDIUM
2026-05-09 09:18	AGIX/USDT	Bybit	Coinbase	2,55	1,89	HIGH

Кореляційний аналіз виявив: Spread % vs Risk Score $\rho = +0,68$ (сильна), Net Profit % vs Tier $\rho = +0,52$, Risk Level vs Tier $\rho = +0,47$ – що підтверджує фундаментальну закономірність: арбітражні можливості є премією за ризик ліквідності. Теоретичний сукупний прибуток ~$12400 (за I=1000 на сигнал) реалістично оцінюється у 20-35 % ($2500-$4300) з урахуванням slippage і latency.

4.7 Результати бектестингу типових стратегій

Виконано 5 бектестів на трьох типах пар і таймфреймах (результати – таблиця 4.5). Метою було підтвердити інтерпретованість і відтворюваність методу, а не оцінити конкретну стратегію. Такий підхід дозволяє об'єктивно

перевірити фундаментальні гіпотези щодо поведінки ринку в умовах історичної волатильності. Висновки: (1) ліквідність обернено корелює з Win Rate і Total Profit (FET 1h – 54%, 8,7%; BTC 1h – 34%, 1,8%), що узгоджується з live-скануванням; (2) Sharpe ratio інтегрально оцінює якість (2,14 для FET проти 0,87 для BTC; >1 прийнятно, >2 добре); (3) денний таймфрейм дає малу вибірку (BT-3 SOL – 51 сигнал), тож рекомендовано 1h/4h; (4) Max Drawdown помірний (≤0,87 %) завдяки стратегії «вхід лише при позитивному сигналі»; (5) комісія Kraken (0,26%) «з'їдає» ~30% валового спреду. Цей емпіричний факт ще раз підкреслює критичну важливість мінімізації торгових витрат у високочастотному трейдингу. Чітко документовано обмеження методу: відсутність моделювання slippage і latency, дискретизація на свічках – бектест є ретроспективною орієнтовною оцінкою, а не точним прогнозом live-поведінки. Тим не менш, виявлені під час симуляцій тенденції залишаються повністю релевантними для фундаментального розуміння мікроструктури крипторинку.

Таблиця 4.5 – Зведені результати бектестингу

#	Пара	TF	Сигналів	Win Rate, %	Total Profit, %	Max DD, %	Sharpe	Комісія, %
BT-1	BTC/USDT	1h	247	34,3	1,84	-0,12	0,87	0,20
BT-2	ETH/USDT	4h	178	33,0	2,16	-0,18	0,71	0,36
BT-3	SOL/USDT	1d	51	28,3	1,95	-0,42	0,58	0,20
BT-4	FET/USDT	1h	392	54,4	8,72	-0,87	2,14	0,20
BT-5	RNDR/USDT	4h	145	40,3	5,38	-0,65	1,42	0,36

Аналіз зведених результатів бектестингу (таблиця 4.5) дозволяє зробити чіткий висновок щодо ефективності розроблених арбітражних

стратегій на різних класах активів. Використання стандартизованого підходу до оцінювання забезпечило високу репрезентативність отриманих метрик. Головною виявленою тенденцією є суттєва перевага альткоїнів (FET, RNDR) над магістральними криптовалютами (BTC, ETH, SOL) за всіма ключовими метриками. Зокрема, пара FET/USDT демонструє найвищий показник успішних угод (Win Rate 54,4 %) та чудовий коефіцієнт Шарпа (2,14), що свідчить про високу стабільність генерування прибутку відносно прийнятого ризику. Подібний профіль ризик-прибутковість є вкрай привабливим показником для алгоритмічних систем даного класу. Натомість високоліквідні активи, такі як Bitcoin та Ethereum, показують відсоток успішних сигналів на рівні 33–34 %, що з урахуванням закладених торгових комісій робить їх менш привабливими для класичного арбітражу. Це зумовлено високою ефективністю ціноутворення на цих ринках, де великі маркетмейкери практично миттєво усувають будь-які цінові дисбаланси. Також прослідковується прямий вплив обраного таймфрейму: менші інтервали (1h) логічно генерують значно більшу кількість торгових сигналів, що дозволяє швидше накопичувати загальний прибуток (Total Profit). Відповідно, вибір правильного часового масштабу стає ключовим етапом оптимізації торгової системи під конкретні інвестиційні цілі. Разом з тим, вкрай низькі показники максимального просідання (Max DD), які не перевищують 0,87% навіть для найбільш волатильних пар, емпірично підтверджують низькоризикову природу реалізованої аналітичної моделі. Це означає, що навіть у періоди глибокої ринкової турбулентності робочий капітал залишається надійно захищеним від різких втрат. Отже, результати тестування не лише підтвердили дієздатність запропонованих алгоритмів, але й дозволили сформувати чіткі критерії щодо оптимальних умов їх застосування. Спираючись на ці дані, можна з упевненістю стверджувати, що подальше масштабування системи шляхом додавання нових низьколіквідних активів здатне пропорційно збільшити загальну доходність стратегії без суттєвого зростання базових ризиків.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну науково-прикладну задачу розробки інтелектуальної веб-системи моніторингу криптовалютного арбітражу, що поєднує покриття чотирьох класів арбітражних стратегій, реалістичне оцінювання executable-ліквідності, статистичне оцінювання ризику сигналів та інтегрований бектестинг на історичних даних у єдиному відкритому програмному рішенні.

Виконано аналіз предметної галузі: класифіковано три типи криптобірж (CEX, DEX, гібридні), описано структуру книги заявок і три виміри ліквідності, класифіковано чотири класи арбітражних стратегій із математичним описом та виявлено три чинники зменшення прибутковості (комісії 0,10–0,60%, затримки 0,5–2 с, обмежена ліквідність). Виконано порівняння п’яти рішень-аналогів за 19 характеристиками й ідентифіковано п’ять обмежень, що обґрунтовують доцільність власної розробки. Сформульовано постановку задачі з 10 функціональними та 7 нефункціональними вимогами і 7 обмеженнями.

Спроєктовано та реалізовано систему з трирівневою архітектурою (34 компоненти) та обґрунтованим стеком із 12 компонентів (Python, Streamlit, ccxt, Pandas, NumPy, Plotly, SQLite, asyncio). Реалізовано подвійну модель даних (три @dataclass-структури та три SQLite-таблиці) за патерном Anti-Corruption Layer. Формалізовано й реалізовано шість ключових алгоритмів з аналізом складності, обґрунтовано асинхронність і batch-запити (скорочення мережевих запитів від 1000 до 5). Реалізовано повну функціональність обсягом близько 4 100 рядків коду у 8 пакетах: шар доступу до 5 CEX і 2 деривативних бірж, формування universe з тришаровим кешуванням, арбітражну логіку, multi-symbol сканер, аналітику деривативів, оцінку ризику, шар персистенції з міграцією та дедуплікацією, бектестинг, графічний інтерфейс із сімома вкладками, допоміжну інфраструктуру з логуванням, моніторингом доступності та Demo Mode.

Виконано комплексну валідацію за чотирирівневою стратегією: 9 pytest-тестів (покриття core 76 %), 4 інтеграційні сценарії (graceful degradation підтверджено), 13 use cases (без зауважень, кросбраузерна сумісність), 6 бенчмарків (скан 200 пар 5,73 с, прискорення кешу1400–5700×). Проаналізовано базу 4 280 сигналів: виявлено три закономірності (64 % - Tier 3; лідер Coinbase-Binance; кореляція spread/risk $\rho = +0{,}68$, інтерпретована як «премія за ризик ліквідності») і серію з 5 бектестів із Sharpe 0,58-2,14.

Наукова новизна: модульна архітектура, що поєднує чотири класи стратегій; механізм tiered-сканування з диференційованими інтервалами; алгоритм executable USD через інтегральне проходження стакану; інтегроване статистичне оцінювання ризику з дедуплікацією. Практичне значення: готовий прототип як дослідницький інструмент, демонстраційна платформа кафедри ШІ ХНУРЕ та основа для розширення; відкрита архітектура й Demo Mode.

Напрямки подальшого розвитку: розширення до 15-20 CEX; інтеграція з реальними DEX через Web3 RPC; перехід на WebSocket-стрімінг; ML-моделі прогнозування виконуваності сигналів; бектестинг на tick-level з моделюванням slippage/latency; міграція на PostgreSQL; автоматизовані фільтри сигналів; контейнеризація через Docker і CI/CD; розширення тестового покриття до 80 %+. Поставлену задачу вирішено в повному обсязі: система задовольняє всі вимоги, демонструє високу продуктивність і відмовостійкість та має відкриту модульну архітектуру.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aldridge I. High-Frequency trading: a practical guide to algorithmic strategies and trading systems. Wiley & Sons, Incorporated, John, 2013. 320 p.

2. Bhansali V. Tail risk hedging: creating robust portfolios for volatile markets. McGraw-Hill Education, 2014. 272 p.

3. Binance spot API docs. *Binance Developer Documentation*. URL: https://binance-docs.github.io/apidocs/spot/en (date of access: 29.06.2026).

4. Bybit API documentation (V5 introduction). *Bybit Exchange*. URL: https://bybit-exchange.github.io/docs/v5/intro (date of access: 29.06.2026).

5. CCXT library documentation. *CCXT*. URL: https://docs.ccxt.com/ (date of access: 29.06.2026).

6. Chabot B. Momentum cycles and limits to arbitrage evidence from Victorian England and post-depression US stock markets. Cambridge, MA : National Bureau of Economic Research, 2009.

7. Chan E. Algorithmic trading: winning strategies and their rationale. Wiley & Sons, Incorporated, John, 2013. 224 p.

8. Coinbase developer documentation. *Coinbase Cloud*. URL: https://docs.cdp.coinbase.com/ (date of access: 29.06.2026).

9. Crépellière T., Pelster M., Zeisberger S. Arbitrage in the market for cryptocurrencies. *Journal of Financial Markets*. 2023. P. 100817. URL: https://doi.org/10.1016/j.finmar.2023.100817 (date of access: 01.07.2026).

10. Crypto arbitrage bot: tools and analysis for trading in the cryptocurrency market. *Arbitrage Scanner*. URL: https://arbitragescanner.io (date of access: 29.06.2026).

11. Crypto market data: derivatives, options, spot, order flow. *CoinGlass*. URL: https://www.coinglass.com/ (date of access: 29.06.2026).

12. Fowler M. Patterns of enterprise application architecture. Addison-Wesley Professional, 2002. 560 p.

13. Empirical market microstructure: the institutions, economics, and econometrics of securities trading. Oxford University Press, USA, 2006. 208 p.

14. Hummingbot: the open source framework for crypto market makers. *Hummingbot*. URL: https://hummingbot.org/ (date of access: 29.06.2026).

15. Introduction to KuCoin API. *KuCoin API Docs*. URL: https://docs.kucoin.com/ (date of access: 29.06.2026).

16. O'Reilly D. Python for beginners: a complete beginner's guide to learning python quickly. Independently Published, 2020.

17. Makarov I., Schoar A. Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics*. 2020. Vol. 135, no. 2. P. 293–319. URL: https://doi.org/10.1016/j.jfineco.2019.07.001 (date of access: 29.06.2026).

18. McKinney W. Python for data analysis: data wrangling with pandas, numpy, and ipython. O'Reilly Media, Incorporated, 2012..

19. Plotly open source graphing library for python. *Plotly*. URL: https://plotly.com/python/ (date of access: 29.06.2026).

20. Pytest documentation. *Pytest*. URL: https://docs.pytest.org/en/stable (date of access: 29.06.2026).

21. Shynkevich A. Law of one price and return on arbitrage trading: Bitcoin vs. Ethereum. *Journal of Economics and Finance*. 2023. URL: https://doi.org/10.1007/s12197-023-09631-0 (date of access: 29.06.2026).

22. Documentation. *SQLite*. URL: https://www.sqlite.org/docs.html (date of access: 29.06.2026).

23. Start building with Kraken. *Kraken Developers*. URL: https://docs.kraken.com/ (date of access: 29.06.2026).

24. Streamlit documentation. *Streamlit*. URL: https://docs.streamlit.io (date of access: 29.06.2026).