Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Марченко Нікіти Вячеславовича*
	(ПІБ)
академічної групи	*122М-21-1*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Розробка та дослідження ефективності впровадження криптовалютного бота для біржі Binance.*

Н.В. Марченко

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Бердник М.Г.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент	*Доц. Шедловський І.А.*			
Нормоконтролер	*Проф. Лактіонов І.С.*	78	добре	

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 20 22 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-21-1* *Марченко Нікіти Вячеславовича*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження ефективності впровадження криптовалютного бота для біржі Binance.*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 10.12.2022 р. № 1036 -с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – динаміка курсів криптовалют.

Предмет досліджень – методи атоматичних торгів на біржі Binance за допомогою криптовалютного бота.

Мета НДР – створення програми бота для атоматичних торгів на біржі Binance з використанням алгоритму на основі причинно-наслідкових зв'язків із ключовими індикаторами.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що розроблено новий

оригінальний алгоритм для автоматичної торгівлі криптовалюти на біржі Binance.

Практична цінність полягає у тому, що автоматична торгівля потребує набагато менше уваги з боку людини. Все, що потрібно зробити трейдеру - налаштувати бота відповідно до своєї стратегії, після чого той буде автономно моніторити пропозиції на ринку і купувати/продавати валюту відповідно до заданих умов. Боти негайно реагують на найменші коливання курсів і автоматично проводять угоди відповідно до заданої стратегії. Людина ж фізично не зможе розмістити сотні ордерів на секунду, якщо це вимагатиме ринкова ситуація.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити використання алгоритму торгівлі. В результаті роботи повинен бути розроблений бот для вирішення задачі автоматичної торгівлі криптовалюти на біржі Binance.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2022-30.09.2022
Побудова моделі та розробка алгоритму	01.10.2022-31.10.2022
Розробка, тестування та дослідження ефективності програмного забезпечення для автоматичної торгівлі криптовалюти на біржі Binance	01.11.2022-16.12.2022

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки успішній торгівлі боту, що принисе прибуток.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки завдяки полегшенню роботи на біржі.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Бердник М.Г.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Марченко Н.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2022 р.*

Термін подання кваліфікаційної роботи до ЕК *16.12.2022*

РЕФЕРАТ

Пояснювальна записка: 92 с., 21 рис., 3 дод., 35 джерел.

Об’єкт дослідження – динаміка курсів криптовалют.

Предмет дослідження – дослідження ефективності впровадження криптовалютного бота для біржі Binance.

Мета роботи – створення програми бота для атоматичних торгів на біржі Binance з використанням алгоритму на основі причинно-наслідкових зв’язків із ключовими індикаторами.

Методи дослідження. У відповідності до поставлених завдань було здійснено дослідження існуючих методів та моделей прогнозування криптовалют, а саме лінії Боллінжера та індикатор SMA.

Новизна отриманих результатів визначається тим, що вперше розроблено новий оригінальний алгоритм для автоматичної торгівлі криптовалюти на біржі Binance.

Практична цінність полягає у тому, що автоматична торгівля потребує набагато менше уваги з боку людини. Все, що потрібно зробити трейдеру - налаштувати бота відповідно до своєї стратегії, після чого той буде автономно моніторити пропозиції на ринку і купувати/продавати валюту відповідно до заданих умов. Боти негайно реагують на найменші коливання курсів і автоматично проводять угоди відповідно до заданої стратегії. Людина ж фізично не зможе розмістити сотні ордерів на секунду, якщо це вимагатиме ринкова ситуація.

Область застосування. Розроблена інформаційна система може застосовуватися для атоматичних торгів на біржі Binance.

Значення роботи та висновки. Результатом роботи виступає бот який на основі причинно-наслідкових зв’язків із ключовими індикаторами відкриває ордера на купівлю або продаж криптовалюти у потрібний момент.

Прогнози щодо розвитку досліджень. Покращити інформаційну систему, додавши новий метод аналізу зміни цін , з метою зменшення відсотка похибки при торгівлі.

Список ключових слів: ПРОГРАМА, БОТ, КРИПТОВАЛЮТА, BINANCE, SMA, BOLLINGER, .NET, VISUAL STUDIO.

ABSTRACT

Explanatory note: 92 p., 21 figures, 3 appendices, 35 sources.

The object of the study is the dynamics of cryptocurrency exchange rates.

The subject of the study is the study of the effectiveness of the introduction of a cryptocurrency bot for the Binance exchange.

The purpose of the work is to create a bot program for automatic trading on the Binance exchange using an algorithm based on causal relationships with key indicators.

Research methods. In accordance with the assigned tasks, a study of the essence of the task and subject area, a study of existing methods and models of cryptocurrency forecasting, namely Bollinger lines and the SMA indicator, was carried out.

The novelty of the obtained results is determined by the fact that for the first time a new original algorithm was developed for the automatic trading of cryptocurrency on the Binance exchange.

The practical value is that automated trading requires much less human attention. All the trader needs to do is configure the bot according to his strategy, after which it will autonomously monitor market offers and buy/sell currency according to the given conditions. Bots immediately react to the slightest fluctuations in rates and automatically execute transactions in accordance with the given strategy. A person physically will not be able to place hundreds of orders per second, if the market situation requires it.

Field of application. The developed information system can be used for automatic trading on the Binance exchange.

Value of work and conclusions. The result of the work is a bot that, on the basis of cause-and-effect relationships with key indicators, opens orders for the purchase or sale of cryptocurrency at the right time.

Forecasts regarding the development of research. Improve the information system by adding a new method of analyzing price changes in order to reduce the percentage of error in trading.

List of keywords: PROGRAM, BOT, CRYPTOCURRENCY, BINANCE, SMA, BOLLINGER, .NET, VISUAL STUDIO.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

OS — Operating System;

IDE — Integrated Development Environment;

IS — Information System;

OOP — Object-Oriented Programming;

IT — Information Technologies;

`SMA` — Simple Moving Average;

SDK — Software Development Kit;

XML — Extensible Markup Language;

GUI — Graphical User Interface;

JSON — JavaScript Object Notation;

IO — Input Output;

API — Application Programming Interface;

PC — Personal Computer;

RAM — Random Access Memory;

CLR — Common Language Runtime;

AI — Artificial Intelligence;

ML — Machine Learning;

IL — Intermediate Language;

JIT — Just-In-Time;

FCL — Framework Class Library;

LINQ — Language Integrated Query;

NLP — Natural Language Processing;

ВСТУП

В сьогодення багато людей у всьому світі щодня використовують комп'ютери, ноутбуки та смартфони. Вони використовують їх для різних сфер життєдіяльності. Також майже кожен чув про криптовалюти, біржі на якких можливо купити криптовалюту. З кожним днем ця сфера зацікавлює все більше людей, надаючи їм багато можливостей у різних напрямках, майже миттєві перекази з мінімальною комісією, трейдінг, зберігання унікальних невзаємозамінних токенів. В розвинених країнах деякі криптомонети починають заміщувати звичайні паперові гроші в окремих сферах людського життя, їх признають на загальнодержавному рівні та встановлюють спеціальні термінали для роботи з даним видом валют[1]. Таким чином, з'явилась можливість користуватись девайсами для трейдінгу.

Безсумнівно, Трейдінг – це галузь, що стрімко розвивається, та надає можливість високого заробітку. Є дуже багато торгівельних стратегій, кожна з яких має унікальний набір алгоритмів для аналізу стану ринку. Досвідчені трейдери користуються одразу декількома індикаторами які надають змогу приймати вигідні рішення.

Автоматична торгівля потребує набагато менше уваги з боку людини. Все, що потрібно зробити трейдеру - налаштувати бота відповідно до своєї стратегії, після чого той буде автономно моніторити пропозиції на ринку і купувати/продавати валюту відповідно до заданих умов. Боти негайно реагують на найменші коливання курсів і автоматично проводять угоди відповідно до заданої стратегії. Людина ж фізично не зможе розмістити сотні ордерів на секунду, якщо це вимагатиме ринкова ситуація.

Ідея торгівельного-бота з користувацьким налаштуванням не зовсім нова ідея для ринку. Про неї багато говорять представники різних крипто-ком'юніті, але отримати її у відкритому доступі фактично неможливо. В даний час можливо створювати ботів з API, наданих біржами, саме мною

обрана біржа Binance, яка є однією з найпопулярніших бірж у світі.

У проекті таке програмне забезпечення буде проаналізовано та обговорено його переваги та недоліки. Завдання даного проекту та його предмет діяльності безпосередньо пов’язані з напрямком підготовки «Комп’ютерні науки» та відповідають узагальненій тематиці кваліфікаційної роботи, типовим завданням діяльності, навичок та компетенцій, які повинні виконуватися магістрам відповідно до освітнього кваліфікації. З Виконання кваліфікаційної роботи дає можливість автору отримати кваліфікацію спеціаліста з розробки та тестування програмного забезпечення.

Актуальність кваліфікаційної магістерської роботи визначена стрімким розвитком криптовалют та технологій, які вони використовують. За умов появи нових видів криптовалют та адаптації існуючих до різноманітних фінансових інститутів, виникає необхідність в детальному вивченні даної тематики, визначення ряду факторів, що впливають на динаміку курсів криптовалют. Завдяки динамічним змінам цін на різні монети і виникає зацікавлениість охочих заробляти завдяки трейдінгу.

У першому розділі проаналізовано предметну область завдання, визначено актуальність направлення, сформульовано постановку проблеми, зазначено вимоги до реалізації боту, технології та програмні засоби.

У другому розділі розглянуто існуючі рішення, обрано платформу розробки, описано розробку застосунка, названо основні алгоритми, індикатори та структуру, наведено довідкову інформацію та функціональні можливості програми, описано інтерфейс користувача системи.

У третьому розділі продемонстровано роботу ПЗ, усі реалізовані налаштування та можливості. Також представлені результати торгівлі.

Результатом цього проекту є віконний додаток, який надає користувачеві унікальний інтерактивний інтерфейс для користування всіма можливостями торгівельного боту та аналізу графіку ринку.

РОЗДІЛ 1. АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальна інформація про предметну область

Криптовалюта. Визначення та опис:

Криптовалюта – це будь-який вид валюти у цифровій чи віртуальній формі; для захисту транзакцій у криптовалюті використовується шифрування (криптографія). Немає центрального органу з випуску чи регулювання криптовалют. Для запису транзакцій та випуску нових одиниць використовується децентралізована система

Криптовалюта – це цифрова платіжна система, під час перевірки транзакцій у якій не беруть участь банки. Це система з рівноправними учасниками, що дозволяє будь-якому користувачеві, що перебуває в будь-якому місці, надсилати та отримувати платежі. Криптовалютні платежі існують виключно в цифровому вигляді в онлайн базі даних, що описує конкретні транзакції. Вони не мають на увазі операцій з фізичними грошима, що мають ходіння та можливості обміну обмін у реальному світі. При переказі коштів у криптовалюті, транзакції записуються до реєстру. Криптовалюта зберігається у цифрових гаманцях.

Термін Криптовалюта узвичаївся завдяки тому, що для перевірки транзакцій використовується шифрування (криптографія): для зберігання та передачі даних про криптовалюту між гаманцями та в загальнодоступні реєстри використовується розширене кодування. Мета шифрування – забезпечити надійність та безпеку.

Першою криптовалютою став біткойн, створений у 2009 році та найвідоміший на сьогоднішній день. Торгівля криптовалютами цікава з погляду отримання прибутку. Внаслідок спекулятивних дій періодично спостерігаються стрибки цін на криптовалюти.

Застосування криптовалюти:

Криптовалюти обробляються в розподіленому публічному реєстрі – блокчейні, де зберігаються записи про всі транзакції, які оновлюють власники

валюти.

Одиниці криптовалюти (монети) створюються у процесі майнінгу. Цей процес, у якому обчислювальна потужність комп'ютера використовується на вирішення складних математичних завдань, у результаті генеруються монети. Користувачі також можуть купувати валюту у брокерів, а потім зберігати та витрачати її за допомогою криптографічних гаманців.

Криптовалюта – це не матеріальний об'єкт, це ключ, який дозволяє переміщати запис чи одиницю виміру від однієї особи до іншої без довіреної третьої сторони.

Біткойн існує з 2009 року, проте у фінансовому плані криптовалюти та застосування технології блокчейн все ще перебувають на етапі становлення.

У майбутньому очікується їх бурхливий розвиток. У перспективі криптовалюти можуть використовуватись у торговельних угодах з акціями, облігаціями та іншими фінансовими активами.

Приклади криптовалют:

- Bitcoin:

Біткойн (BTC), створений у 2009 році, став першою криптовалютою і досі зберігає найвищу популярність. Валюта була розроблена Сатоші Накамото – вважається, що це псевдонім людини чи групи людей, а точна особистість розробника залишається невідомою.

- Ethereum (Ефіріум):

Блокчейн-платформа Ethereum була розроблена у 2015 році. Вона має власну криптовалюту Ether (ETH) чи Ethereum. Це найпопулярніша криптовалюта після биткойна.

- Litecoin:

Ця валюта найбільше схожа на біткойн, але в ній оперативніше розвиваються нововведення, такі як швидкі платежі та процеси, що дозволяють проводити більше транзакцій.

- Ripple:

Ripple - це система з розподіленим реєстром, заснована у 2012 році. Ripple можна використовувати для відстеження різних видів транзакцій, не лише криптовалютних. Компанія-розробник платформи Ripple працювала з різними банками та фінансовими установами.

Криптовалюти, відмінні від біткойна, називають загальним терміном "альткойни", щоб відрізняти від оригіналу.

Як купити криптовалюту:

Може виникнути питання, як безпечно купити криптовалюту. Зазвичай це відбувається у три етапи.

Перший крок – вибрати платформу для використання. Як правило, можна вибрати між традиційним брокером чи спеціалізованою біржею криптовалют.

Традиційні брокерів. Це онлайн-брокери, що пропонують купівлю та продаж криптовалюти, а також інших фінансових активів: акцій, облігацій, ETF. Ці платформи, як правило, пропонують нижчі торгові комісії, але менше криптовалютних функцій.

Криптовалютні біржі. Існує безліч криптовалютних бірж, кожна з яких пропонує різні криптовалюти, сховище гаманців, варіанти відсоткових рахунків та багато іншого. Багато бірж стягують комісію залежно від активів, що торгуються.

При порівнянні платформ рекомендується звернути увагу на криптовалюти, що торгуються, комісії, функції безпеки, варіанти зберігання та виведення коштів, а також освітні ресурси.

Наступний крок після вибору платформи – це поповнення рахунку, щоб з'явилася можливість почати торгувати. Більшість криптовалютних бірж, залежно від платформи, дають змогу користувачам купувати криптовалюту за випущені державою валюти, такі як долар США, британський фунт, євро, а також при оплаті дебетовими або кредитними картками.

Покупки криптовалюти з оплатою кредитних карток вважаються ризикованими, тому підтримуються не всіма біржами. Деякі компанії, що

випускають кредитні картки, також не дозволяють здійснювати криптовалютні транзакції. Це пов'язано з вкрай високою волатильністю криптовалют – при торгівлі певними активами не рекомендується ризикувати, роблячи угоди в борг або потенційно виплачуючи високі комісії за транзакції за кредитними картками.

Деякі платформи також приймають ACH-перекази (перекази через автоматизовану клірингову палату) та банківські перекази. Допустимі способи оплати та час на введення та виведення коштів залежать від платформи, а час клірингу депозитів залежить від способу оплати.

Важливим фактором, на який слід звернути увагу, є розмір комісій, що включають потенційні комісії за введення та виведення коштів, а також торгові комісії. Розмір комісій варіюється в залежності від способу оплати та платформи. Це питання рекомендується досліджувати на етапі вибору платформи.

Замовлення можна розміщувати через веб-сайт або мобільний додаток вибраного брокера чи біржі. Щоб купити криптовалюту, потрібно вибрати варіант «Купити», тип замовлення, вказати суму криптовалюти, що купується, і підтвердити замовлення. Аналогічно розміщується замовлення на продаж.

Існують інші способи інвестувати в криптовалюту. До них відносяться платіжні сервіси, такі як PayPal, Cash App та Venmo, які дозволяють купувати, продавати та зберігати криптовалюти. Крім того, існують такі інвестиційні інструменти:

Біткойн-трасти. Акції біткойн-траст можна придбати на звичайний брокерський рахунок. Такі інструменти надають індивідуальним інвесторам доступ до криптовалюти через фондовий ринок.

Фонди взаємних інвестицій у біткойни. Існують ETF, прив'язані до біткойнів, а також фонди взаємних інвестицій у біткойни.

Блокчейн-акції та ETF. Непрямо інвестувати в криптовалюту можна через блокчейн-компанії, що спеціалізуються на технологіях, що лежать в основі майнінгу криптовалют та криптовалютних транзакцій. В якості альтернативи можна купити акції або ETF компаній, які використовують технологію

блокчейну.

Оптимальний варіант залежить від інвестиційних цілей та схильності до ризику.

Зберігання криптовалюти:

Після придбання необхідно забезпечити надійне зберігання криптовалюти, що гарантує захист від злому та крадіжки. Зазвичай криптовалюта зберігається у криптогаманцях. Це фізичні пристрої або онлайн-програми, які використовуються для безпечного зберігання закритих ключів до криптовалют. Деякі біржі надають послуги гаманця, завдяки чому зберігання криптовалютних засобів здійснюється безпосередньо самою платформою, проте автоматично такі послуги надають не всі біржі та брокери.

Існують також різні провайдери гаманців. Існують два типи зберігання засобів: «гарячий гаманець» та «холодний гаманець».

Гарячий гаманець є криптографічне сховище, що використовує онлайн-програми для захисту закритих ключів до активів.

Холодний гаманець (також званий апаратним гаманцем) на відміну від гарячого гаманця використовує автономні електронні пристрої для безпечного зберігання закритих ключів [28].

Як правило, за використання холодних гаманців стягується комісія, а використання гарячих не стягується.

Криптобіржа Binance:

Binance — екосистема, що включає централізовану біржу криптовалют із найбільшим у світі обсягом торгів, блокчейн-платформу BNB Chain, навчальний центр, інвестиційний та благодійний фонди, NFT-маркетплейс та інші продукти.

Першим та головним продуктом Binance є біржа криптовалют, яка почала працювати у 2017 році.

На ній представлені різні інструменти: спотова торгівля цифровими активами та угоди з плечем, деривативи, секція P2P-обміну. Крім того, користувачам доступна можливість пасивного заробітку на криптовалютах

шляхом стейкінгу та фармінгу, функція випуску дебетової картки та інші сервіси.

Згідно з офіційним сайтом, станом на квітень 2022 року Binance підтримує понад 600 криптоактивів та понад 40 фіатних валют. На платформі зареєстровано 90 млн користувачів.

За підсумками опитування, проведеного наприкінці 2021 року, ця біржа стала найпопулярнішою серед читачів ForkLog – на ній торгують понад 70% респондентів.

Binance не розкриває фінансових даних. За різними оцінками, її виручка у 2021 році склала від $14,6 млрд до $20 млрд. За інформацією ЗМІ, у 2021 році власна оцінка компанії дорівнювала $200 млрд.

Точна кількість співробітників Binance невідома, але на початку 2022 року гендиректор проекту Чанпен Чжао розповів, що віддалений штат працівників налічує близько 4000 осіб.

Як працюють криптовалютні боти:

Торговий бот аналізує ситуацію на криптобіржах і автоматично закриває угоди, відповідно до сценарію, закладеного в нього розробником або самим кріпотрейдером. Алгоритм процесів сервісу називається стратегією.

Наприклад, трейдер може встановити умову, щоб бот закуповував певну криптовалюту при зниженні її курсу на 5-7% і стабілізації на цій відмітці, і продажу, коли вартість монети знову почне зростати. Враховуючи волатильність цифрових валют, угоди з купівлі та продажу можуть відбутися в один день і згенерувати прибуток трейдеру.

1.2. Огляд та аналіз існуючих аналогів ботів

У міру розвитку криптоіндустрії на ринку з'являлися різні криптоботи, що полегшують роботу трейдерів. Ми виділили кілька основних типів, залежно від їхнього функціоналу, а саме боти для торгівлі на криптобіржі поділяються на:

- Торгові боти, які заробляють на продажі цифрової валюти. Вони відстежують тенденції, роблять прогнози на основі ринкових індикаторів, таких як RSI, MACD та лінії Болліндженра, а потім закривають угоди у

найбільш вдалий момент.

- Арбітражні боти, що працюють одразу на кількох біржах і переслідують завдання купити в одному місці дешевше, і тут же продати в іншому — дорожче. Справа в тому, що криптобіржі не встановлюють курс валют, а отже, він може значно відрізнятися на різних платформах в однаковий момент часу.
- Маркет-мейкінг боти, такі роботи формують лімітні ордери на купівлю/продаж криптовалют і дозволяють отримати прибуток за рахунок коливань курсів. Це підвищує ліквідність обраної цифрової валюти на біржі, за рахунок чого трейдер може розраховувати на бонус від платформи у вигляді зниження комісій на торгівлю.
- Сигнальні боти, що аналізують дані у ЗМІ та блогах провідних трейдерів, і на основі цієї інформації дають трейдеру поради щодо купівлі або продажу тієї чи іншої монети.
- Кредитні боти, які відкривають можливість позичати свою криптовалюту з метою отримання прибутку. При цьому користувач може сам встановити умови, обсяг та строк позики, залежно від своїх переваг. У роботі з такими ботами важливо моніторити ринок, щоб сформувати пропозицію, яка буде цікавою для позичальників та вигідною для трейдера [31].

Популярні боти для криптобірж:

На ринку є готові рішення — вже розроблені торгові роботи, які можна налаштовувати та інтегрувати в роботу. Розглянемо найпопулярніші варіанти, що здобули хорошу репутацію серед трейдерів.

CRYPTOHOPPER (Рис. 1.1).

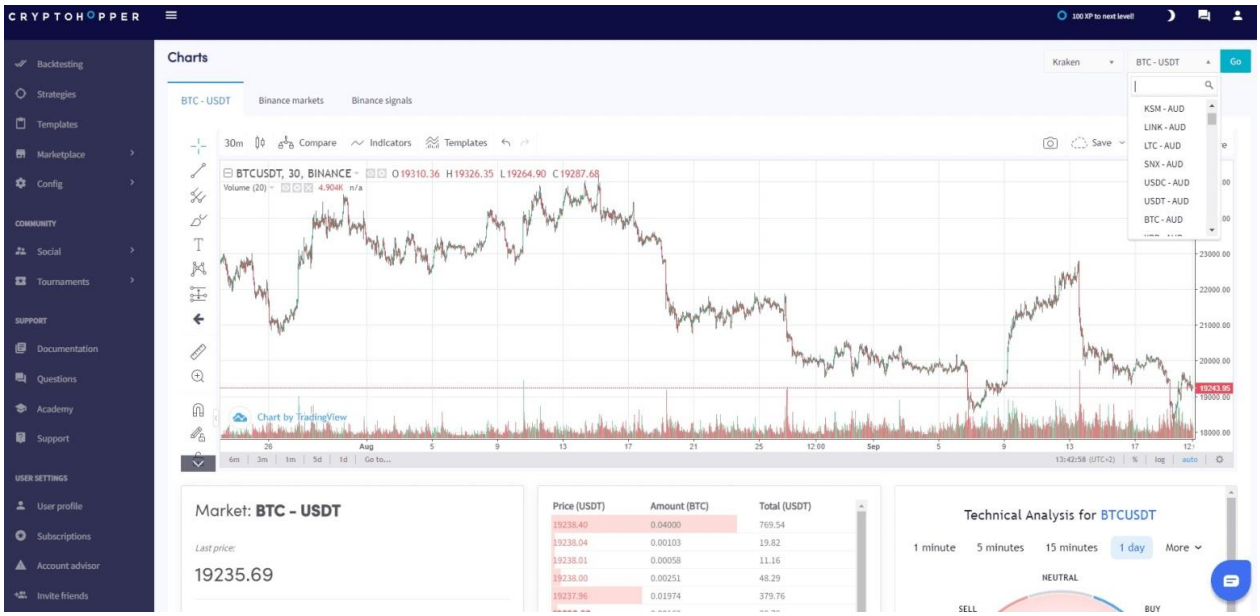
Вартість: до $99 на місяць.

Кількість підтримуваних бірж: 10.

Рис. 1.1. CRYPTOHOPPER

Можливості Cryptohopper

Сервіс Cryptohopper є набір інструментів для автоматизації торгівлі. Орієнтований на торгівлю через ботів: доступно чотири стратегії, що настроюються. Навички програмування для налаштування ботів Cryptohopper не потрібні: установка параметрів реалізована через інтерфейс. Доступно створення



шаблонів та збереження налаштувань бота.

При використанні базового облікового запису користувачі Cryptohopper можуть запускати до 20 ботів одночасно. Також можливе виставлення ордерів безпосередньо через біржову склянку.

У Cryptohopper реалізовано підключення до 15 криптовалютних бірж через ключі API. Статистичні дані транслюються через TradingView.

Додаткові інструменти Cryptohopper: статистика рахунків, опрацювання сигналів та стратегій, тестування на історичних даних у режимі Paper Trading. У маркетплейсі Cryptohopper доступні користувальницькі сигнали, боти з встановленими конфігураціями та підключення Mirror Trading – готової торгової моделі трейдерів(Рис.1.2).

Торгівля із ботами

Торгові боти Cryptohopper називаються "Hopper". Створення бота можливе за трьома алгоритмами:

- автоматична торгівля – класичний бот з можливістю налаштування параметрів, продажу, покупки, Stop Loss та Take Profit.
- арбітраж – бот, який торгує між різними біржами та отримує прибуток за рахунок різниці курсів.
- маркет-мейкер – бот аналізує склянку та стрічку заявок, дозволяючи торгувати через спред.

Маркет-мейкер та арбітраж у Cryptohopper доступні лише за платною підпискою. До діючого роботу можна підключити «Алгоритмічний Інтелект» (AI), який автоматично генеруватиме сигнали на купівлю та продаж. Перед запуском, AI можна «навчити» та налаштувати у тестовому середовищі.

Плюси: вбудовано багато індикаторів, включаючи RSI, EMA, Parabolic Sar. Можна вибрати Рис.1.2. Приклад роботи CRYPTOHOPPER .

BITSGAP (Рис. 1.3).

Вартість: $19 на місяць

Кількість підтримуваних бірж: 30

Рис. 1.3. BITSGAP

Торгова платформа пропонує зручність купівлі та продажу криптоактивів на кількох біржах.

Ви можете миттєво перемикатися між парами та біржами, не відкриваючи нову вкладку браузера. Користувачі мають доступ до графіків TradingView, типів графіків, що настроюються, і більш ніж 100 технічним індикаторам.

Ви також можете переглядати історію торгівлі, відстежувати відкриті позиції та керувати своїм балансом на кожній із зв'язаних бірж.

Bitsgap також поставляється з набором аналітичних торгових інструментів, щоб оцінити продуктивність всіх популярних торгових пар.

Можна легко інтегрувати можливість вибору стилю графіка та встановлення таймфрейму.

Щоб дати вам більше контролю над вашою торговою практикою, існують різні торгові ордери, в тому числі:

- Stop-Limit. Встановлена межа ціни, за якою здійснюється угода;
- Ринкові заявки. Найкраща доступна ринкова ціна, за якою ви можете

миттєво відкрити позицію;

- Тейк-профіт/Стоп-лосс. Обмежте збиток за позицією, якщо ринок розгорнеться не на вашу користь;
- Тіньові замовлення. Обмін біржової книги ордерів з виконанням ордера лише за встановленою ціною.

Торгові Боти

Bitsgap Торговий бот - це ексклюзивна функція, яка дозволяє вам отримувати невеликий, але частий прибуток при кожному русі ринку.

Він заснований на популярній сітковій стратегії, яка включає розміщення серії ордерів на купівлю і продаж в межах заданого цінового інтервалу.

Ваші інвестиції розподіляються за декількома рівнями, на яких виконуються замовлення та розміщуються нові. Деякі стратегії роботів, доступні на платформі Bitsgap, включають:

- Класичний бот

Він включає в себе купівлю та продаж фіксованої суми базової валюти за замовлення, щоб отримати прибуток від кожної завершеної угоди.

Ця стратегія довела свою ефективність на ринку, що росте, завдяки своїй логіці розподілу інвестицій. Наприклад, якщо у вас 10 ETH в якості базової валюти бот буде купувати та продавати тільки цю суму на кожному рівні.

- SBot

Це модернізована версія класичного робота, що відрізняється логікою розподілу інвестицій. SBot купує та продає фіксований обсяг котированої валюти за ордер.

Наприклад, якщо у вас є 100 доларів як основна сума, Sbot купує ETH на 100 доларів, коли ціна падає. Навпаки, коли ціна зростає.

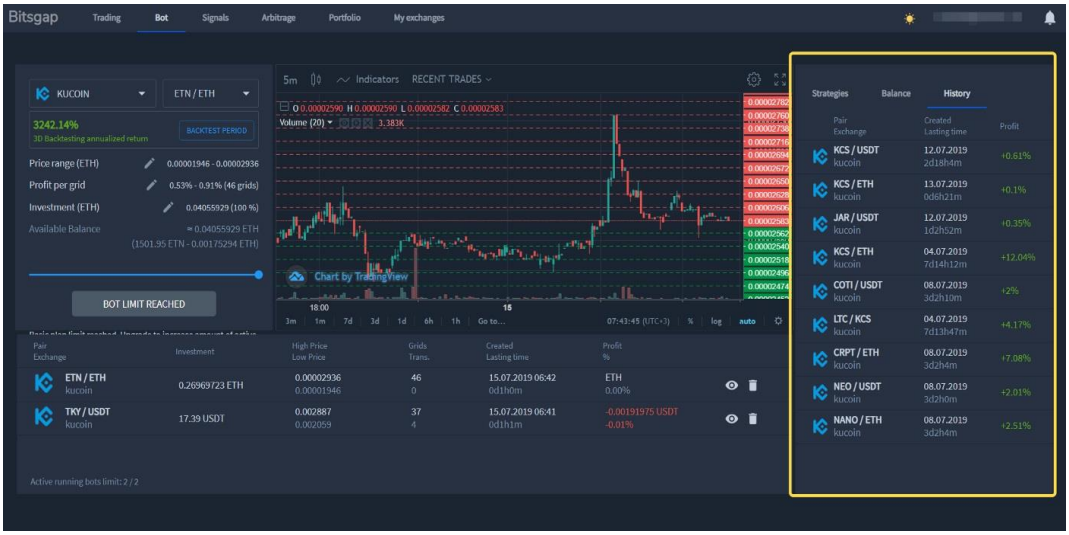
Таким чином, ви накопичуєте більше монет за нижчими цінами і продаєте менше монет за вищими цінами. Ця стратегія є синонімом методу DCA та може успішно використовуватись у бічних ринкових умовах.

- Комбо-бот

Якщо ви займаєтеся торгівлею ф'ючерсами, цей бот може допомогти вам отримати прибуток як на ринках, що ростуть, так і на падаючих. Він поєднує в собі алгоритми DCA та сітки для здійснення операцій при кожному русі ринку.

Вбудована функція трейлінгу бота автоматично розміщує рівні сітки, так і рівні DCA, щоб нескінченно генерувати дохід(Рис. 1.4).

Плюси: є тріальна версія, і демо-режим, у якому можна попрактикуватись. Можна інтегрувати додаткові торгові функції: тейк-профіт, трейлінг-стоп тощо



[34].

Рис. 1.4. Приклад роботи BITSGAP

REVENUEBOT (Рис. 1.5).

Вартість: 20% від прибутку, але не дорожче за $50 на місяць.

Кількість підтримуваних бірж: 10

Рис. 1.5. REVENUEBOT

Особливості та функції RevenueBOT:

Боти RenevueBOT працюють на спотових та ф'ючерсних ринках криптовалютних бірж. З'єднання реалізовано через API-ключі. На платформі є два готових торгових робота і більше 400 конфігурацій для налаштування власного робота.

У робочому просторі RevenueBOT немає графіків, склянок та інших звичних біржових інструментів. Взаємодія з ринком здійснено лише через торгових роботів. Також реалізовано розширені протоколи для збирання статистики(Рис.1.6).

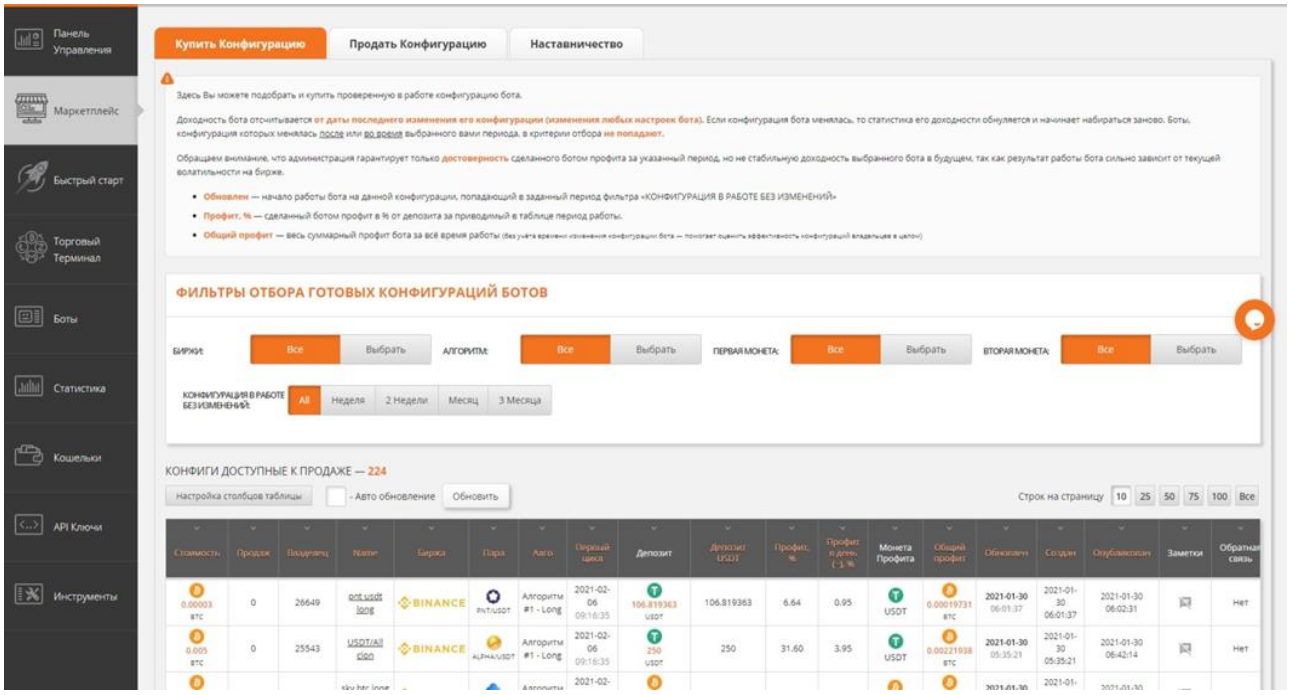


Рис. 1.6. Приклад роботи REVENUEBOT

Боти торгують через сітку ордерів. Користувачі можуть настроювати обсяги від загального депозиту та відсоток перекриття зміни ціни. Одночасно можна використовувати ботів для кількох валютних пар. Доступно перемикання шаблонів налаштувань бота між ринками.

Крім роботів, у RevenueBOT інтегровані інструменти технічного аналізу: індикатори волатильності та докладна статистика за циклом кожного алгоритму. Для тестування стратегій розроблено симулятор із можливістю торгівлі на історичних графіках за останні два місяці.

Створення та запуск бота

Для використання RevenueBOT.io необхідно зареєструватися та надати сервісу API-ключі. Також потрібно створити гаманець для визначення депозиту, з яким працюватиме бот. Кошти, що використовуються, не залишають торговий рахунок користувача на біржі.

Під час створення бота потрібно вибрати його тип:

Long – купівля частинами при падінні ціни на каналі та продаж на верхніх рівнях опору;

Short – продаж частинами при зростанні ціни та купівля під час падіння до нижнього рівня опору.

Після створення бота необхідно налаштувати відсотки та умови купівлі/продажу активу. Додатково конфігурації RevenueBOT дозволяють встановити тимчасові та канальні фільтри, щоб обмежити збитки та автоматично зупиняти роботу алгоритму під час корекції ринку.

У головній панелі управління доступна докладна статистика: за ордерами, заробітком, доходом від реферальних програм. У налаштуваннях RevenueBOT можна підключити сповіщення через e-mail, SMS або Telegram.

Із плюсів: працює на хмарі, тобто встановлювати на ПК нічого не доведеться, бере за використання відсоток угод, а отже — зацікавлений у позитивному результаті трейдера.

Що краще: готове рішення чи власний торгівельний бот?

Функціональний і перевірений бот коштує чимало, що на довгій дистанції виявляється дорожчим за розробку власного рішення. У той же час використання сумнівних і більш дешевих варіантів пов'язане з ризиком встановити шкідливе програмне забезпечення, яке обнулює підключені гаманці.

Власний торговий бот може мати необмежений функціонал, ув'язнений на вирішення саме ваших завдань

1.3. Мета розробки та сфера застосування бота

Програмним додатком будуть користуватися користувачі, які мають настільний ПК під управлінням ОС Windows з доступом до Інтернету.

Мета бота — надати користувачеві можливість автоматично відкривати та закривати угоди, згідно з алгоритмом.

Операційна мета бота — реагувати на найменші коливання курсів, і автоматично проводити операції, відповідно до заданої стратегії, зручний, простий та інтуїтивно зрозумілий користувальницький інтерфейс, який не вимагає особливих навичок та умінь користуватися додатком.

Ідея створення програми такого типу полягала в тому, щоб створити

багатофункціональне і зручне ПЗ, який допоможе користувачеві займатися трейдингом і приділяти цьому мінімум часу.

Підсумовуючи, цей додаток може бути дійсно необхідним користувачам які хочуть менше нервувати.

1.4. Постановка задачі

Метою роботи є створення ПЗ — криптовалютного бота. Додаток розроблено для ОС Windows. У IDE Visual Studio 2022 з використанням мови програмування C#.

Простота інтерфейсу повинна дати можливість користуватися утилітою людям різного віку.

Додаток складається з графічної частини та налаштування. Структура графічної частини складається з наступного:

- Свічки на графіку.
- Показник SMA на графіку.
- Показник Bollinger на графіку.
- Показник SAR на графіку.

Структура частини налаштування складається з наступного:

- Обрати монету.
- Обрати суму ставки.
- Налаштувати розмір Take profit.
- Налаштувати розмір Stop loss [16].

Програму потрібно виконати в Visual Studio IDE для ОС Windows. Додаток Crypto Indicator має мати вікно помилок, де буде відображатися помилки які можуть з'являтися поки працює бот. Крім того, інтерфейс має бути інтуїтивно зрозумілим.

Кнопка відкриття угоди, якщо людина хоче сама відкрити угоду.

Кнопка закриття угоди, якщо людина хоче сама закрити угоду.

Чекбокс онлайн-графік, якщо людина хоче бачити графіки онлайн або ні.

Чекбокс угода, якщо людина хоче щоб працювала стратегія.

1.4.1. Вимоги до програмного забезпечення

Основна вимога – негайна реакція на найменші коливання курсів, та автоматично проведення угод на біржі Binance, відповідно до заданої стратегії.

1.4.2. Функціональні вимоги

Додаток Crypto Indicator повинен забезпечувати можливість виконання наступних функцій і можливостей:

− Відкриття угоди.

− Закриття угоди.

− Автоматичне відкриття угоди.

− Автоматичне закриття угоди.

Надійну (стабільну) роботу Crypto Indicator має забезпечити користувач, виконавши комплекс організаційно-технічних заходів, перерахованих нижче:

− організація безперебійного електропостачання технічного обладнання;

− використання ліцензійного програмного забезпечення;

− відсутність сторонніх або шкідливих програм, які можуть призвести до вимкнення цієї програми [4].

Необхідно враховувати доступність інтерфейсу для користувачів під час його проектування та розробки. Додатком можуть користуватися люди, які погано розбираються в технологіях та пошуку інформації.

Тому при розробці інтерфейсу важливо дотримуватися наступних правил:

1. Інтерфейс повинен бути візуальним, тобто розташування елементів має бути зрозумілим і зрозумілим користувачам.

2. Інтерфейс повинен бути виконаний у спокійних, ненав’язливих тонах і бути читабельним [18].

1.4.3. Вимоги інформаційної безпеки

Вимоги до інформаційних структур і методів вирішення програмного забезпечення та інформаційної безпеки полягають у тому, що системне програмне забезпечення, яке використовується програмою, має бути представлене ліцензованою локалізованою версією операційної системи для конкретного пристрою, сумісної з компонентами програми.

Щоб уникнути некоректної роботи програми, необхідно реалізувати контроль надходження даних, обробку виняткових ситуацій, а також бути впевненим у незмінному стані даних, що зберігаються в конфігураційних файлах, у разі збою програми чи екстремальних обставин [24].

1.4.4. Вимоги до апаратного середовища

Для стабільної роботи даного програмного забезпечення необхідний пристрій з наступними характеристиками.

Настільний або мобільний пристрій під ОС Windows, який може отримати доступ до інтернету.

1.4.5. Вимоги до сумісності

Програмний продукт розроблено в середовищі Visual Studio 2022.

Програмний код написаний мовою програмування C# за допомогою бібліотек Binance.Net [3], Newtonsoft.Json, CryptoExchange.Net, ScottPlot, EntityFramework.

1.5. Висновок

У данному розділі проаналізовано предметну область завдання, визначено актуальність направлення, сформульовано постановку задачі, зазначено вимоги до реалізації боту, технології та програмні засоби.

.

РОЗДІЛ 2. ДИЗАЙН ТА РОЗРОБКА ПЗ

2.1. Функціональні цілі ПЗ

Аналіз вимог – це визначення потреб і умов, які розглядає новий або оновлений продукт, беручи до уваги можливі суперечливі вимоги різних клієнтів, наприклад користувачів або бенефіціарів [20].

Аналіз вимог має вирішальне значення для успішного розвитку проекту. Вимоги повинні бути задокументовані, піддані вимірюванню, тестуванню та описані достатньо детально, щоб розробити програмну систему.

Функціональні вимоги перераховані як функції та послуги, які повинні надаватися системою, а також обмеження системи на основі даних і поведінки під час їх виконання та взаємодії с компонентами [31].

Специфікації функціональних вимог — опис функцій та їх властивостей, які не містять винятків і будь-яких протиріч [13].

Додаток Crypto Indicator — в першу чергу сервіс для автоматичної торгівлі криптовалюти на біржі Binance. Для системи, що розробляється, були сформульовані наступні функціональні вимоги, наведені нижче.

Розроблена програма повинна працювати з біржею Binace та мати необхідну адаптивність для реалізації в рамках інших сервісах даного типу, які надають східний типовий функціонал користувача.

Користувач може самостійно відкривати, або закривати ордер на купівлю чи продаж певної монети.

Є можливість дивитись графік зміни цін для обранного токену у режимі онлайн з різними часовими інтервалами.

Користувачі можуть змінювати налаштування бота, щоб увімкнути автоматичну торгівлю певної монети на обрану суму.

Функціональна мета програми Crypto Indicator для користувачів – це можливість автоматичної торгівлі криптовалюти на біржі Binance.

2.2. Опис використаних математичних методів

Реалізація автоматичної торгівлі розроблена спираючись на математичні методи алгоритму «Лінії Боллінжера» та «SMA».

2.2.1. Bollinger Bands

Формули розрахунку індикатора Bollinger Bands на прикладі простої Сковзної середньої (Simple Moving Average):

Основна лінія (середня)

ML = SUM (CLOSE N) / N = SMA (CLOSE N), де:

SUM сума за N періодів

CLOSE ціна закриття свічки

N кількість періодів, що використовуються для розрахунку

SMA проста Ковзна середня.

Верхня лінія розраховується на підставі середньої

TL = ML + (D * StdDev), де D - число стандартних відхилень.

Нижня лінія розраховується так само на підставі середньої

BL = ML – (D* StdDev), де D – число стандартних відхилень.

При цьому StdDev розраховується за такою формулою:

StdDev = SQRT (SUM ((CLOSE - SMA (CLOSE, N))^2, N)/N), де SQRT квадратний корінь [21].

В даний час при поширеності комп'ютерних технологій розраховувати індикатор Смуги Болліджера самостійно немає жодної необхідності, всі розрахунки комп'ютер проведе автоматично, достатньо в налаштуваннях ввести необхідний параметр центральної лінії, який залежить від переваг трейдера та конкретних торгових умов (у класичному налаштуванні застосовується період 20, але на практиці можна використовувати період від 12 до 24) [19].

2.2.2. SMA

SMA - просте арифметичне ковзне, яке відрізняється від інших МА тим, що має інший ваговий коефіцієнт, який тут присвоюється його останнім

показникам - це величини вартості якогось тимчасового відрізка, вони вивчаються з рівною вагою [22].

У такого інструмента розрахунок даних грунтується на підсумовуванні вартості закриття активів за число відрізків (наприклад, за 10, 15 год.). Далі показник ділиться на кількість відрізків часу. Таким чином, отримуємо таку формулу:

SMA = SUM (CLOSE (i), N)/N, де:

SUM - сума;

CLOSE (i) - ціна закриття періоду;

N - Число періодів [26].

2.3. Опис використаних шаблонів архітектури та дизайну

2.3.1. JSON

JSON (JavaScript Object Notation) — це простий у використанні формат і метод зберігання даних. Основною особливістю JSON є текстова структура, яка дозволяє легко читати та редагувати об’єкт. Файли даних легко обробляти та створювати машинами. Структура заснована на підмножині стандарту мови програмування JavaScript ECMA-262 3rd Edition — грудень 1999. Формат не залежить від мови програмування та використовує принципи, знайомі розробникам сімейства мов C, наприклад C#, Perl, Java, C, C++, JavaScript та інші. Ці функції роблять JSON найкращим форматом зберігання даних для невеликих файлів, таких як файли конфігурації та налаштувань.

JSON заснований на двох основних структурах. Колекції пар імен і значень. У різних мовах програмування він реалізується як структура, об'єкт, словник, списки, запис, хеш-таблиця або асоціативний масив [12].

2.3.2. Binance.Net

Binance.Net — це бібліотека, заснована на платформі .NET і повністю написана на C# [3].

Сокети надає доступ до API веб-сокету біржі. Потоки пропозицій Websocket API, які надсилаються оновлення, які клієнт може прослуховувати. Деякі біржі також пропонують деякий рівень функціональності, дозволяючи клієнтам давати команди через веб-сокет, але більшість бірж дозволяють це тільки через API. Так само, як клієнт Rest поділено на клієнтів Rest Api, клієнт Socket поділено на клієнтів Socket Api, кожен із яких має власний набір функцій API. Клієнти Socket Api зазвичай не діляться на теми, оскільки кількість методів не така велика, як у клієнта Rest.

Клієнт надає доступ до кінцевої точки Rest API. Доступ до інших кінцевих точок здійснюється шляхом надсилання HTTP-запиту та отримання відповіді. Клієнт поділено на кілька субклієнтів, які називаються клієнтами

API. Потім ці клієнти API знову поділяються на різні теми. Зазвичай клієнт Rest виглядає так:

- SpotApi
  - Рахунок
  - Обмін даними
  - Трейдинг
- FuturesApi
  - Рахунок
  - Обмін даними
  - Трейдинг

Бібліотека заснована на гнучкої архітектурі, що дозволяє без особливих зусиль адаптувати її до будь-яких потреб.

Основні особливості Binance.Net включають наступне (як зазначено на офіційному сайті):

- ExchangeData
  - ExchangeData.GetAggregatedTradeHistoryAsync - Отримує стислі, сукупні угоди. Угоди, які виконуються в той час, з того самого замовлення, з тією самою ціною, матимуть агреговану кількість.
  - ExchangeData.GetAssetIndexAsync - Отримайте індекс активів для режиму Multi-Assets для символу.
  - ExchangeData.GetBookPriceAsync - Отримує найкращу ціну/кількість у книзі замовлень для символу.
  - ExchangeData.GetBookPricesAsync - Отримує найкращу ціну/кількість у книзі

замовлень.

- ExchangeData.GetCompositeIndexInfoAsync - Отримує інформацію про складений індекс.
- ExchangeData.GetContinuousContractKlinesAsync - Отримайте дані свічок для наданої пари.
- ExchangeData.GetExchangeInfoAsync - Отримайте інформацію про обмін, включаючи обмеження курсу та список символів.
- ExchangeData.GetFundingRatesAsync - Отримати історію ставок фінансування для наданого символу.
- ExchangeData.GetTickersAsync - отримання тікера.
- ExchangeData.GetOrderBookAsync - отримання книги ордерів символу.
- ExchangeData.GetTradeHistoryAsync - отримання останніх угод за символом.
- ExchangeData.GetKlinesAsync - Отримайте свічки для символу.
- ExchangeData.GetMarkPriceKlinesAsync - Свічкові бари для ціни марки символу.
- ExchangeData.GetPriceAsync - Отримує ціну символу.
- ExchangeData.GetTradeHistoryAsync - Отримати історію торгівлі для символу

• Account

- Account.GetBalancesAsync - Отримує залишки на рахунку.
- Account.StartUserStreamAsync - Підписка на оновлення замовлення.
- Account.GetAccountInfoAsync - Отримує інформацію про рахунки, включаючи баланси.

- Account.GetBracketsAsync - Отримує умовні дужки та кредитне плече.
- Account.GetDownloadIdForTransactionHistoryAsync - Отримайте ідентифікатор завантаження для завантаження історії транзакцій.
- Account.GetMarginChangeHistoryAsync - Запитує історію змін маржі для певного символу.
- Account.GetPositionInformationAsync - Отримує інформацію про обліковий запис.
- Account.GetPositionModeAsync - Отримати режим позиції користувача (режим хеджування або односторонній режим).
- Account.GetTradingStatusAsync - Отримує поточний статус правил торгівлі для облікового запису.
- Account.GetUserCommissionRateAsync - Отримує комісійні ставки облікового запису.
- Account.KeepAliveUserStreamAsync - Підтримуйте потік користувачів. Це слід викликати кожні 30 хвилин, щоб запобігти зупинці потоку користувача
- Account.ModifyPositionMarginAsync - Змінити маржу на відкритій позиції
- Account.StartUserStreamAsync - Отримати ключ слухання можна використовувати для підписки на потік користувача за допомогою клієнта сокета
- Account.StopUserStreamAsync - Зупинка потоку користувача, оновлення більше не надсилатимуться

• Trading

- Trading.PlaceOrderAsync - розміщення замовлення.

- Trading.GetOrderAsync - Запит конкретного замовлення.
- Trading.GetOrdersAsync - Запит історії замовлень.
- Trading.CancelOrderAsync - Скасувати замовлення.
- Trading.GetUserTradesAsync - Отримувати угоди користувача.

• UsdFuturesStreams

- UsdFuturesStreams.SubscribeToAllTickerUpdatesAsync - підписка на оновлення ринкових даних.
- UsdFuturesStreams.SubscribeToUserDataUpdatesAsync - підписка на оновлення замовлення.
- UsdFuturesStreams.SubscribeToOrderBookUpdatesAsync - Підписується на потік оновлення глибини для наданих символів.
- UsdFuturesStreams.SubscribeToMarkPriceUpdatesAsync - Підписується на потік оновлення цін для одного символу
- UsdFuturesStreams.SubscribeToKlineUpdatesAsync - Підписується на потік оновлень свічок для наданого символу [3] .

2.4. Шаблони дизайну

Стилізація та використання шаблонів Windows Presentation Foundation (WPF) належать до набору можливостей, які дозволяють розробникам та дизайнерам створювати візуально привабливі ефекти та узгоджений зовнішній вигляд своїх продуктів. При налаштуванні зовнішнього вигляду програми необхідна строга модель стилізації та шаблонів, що забезпечує обслуговування та спільне використання зовнішнього вигляду у програмах та

між ними. WPF надає таку модель.

Ще однією можливістю моделі стилізації WPF є поділ уявлення та логіки. Дизайнери можуть створювати зовнішній вигляд програми тільки за допомогою XAML в той же час, коли розробники працюють над логікою програми, використовуючи мови C# або Visual Basic [7].

Стилі

Елемент Style можна розглядати як зручний спосіб застосування набору значень властивостей кількох елементів. Стиль можна використовувати для будь-якого елемента, похідного від FrameworkElement або FrameworkContentElement, наприклад, Window або Button [6].

Найчастіше стиль оголошується як ресурс у розділі Resources XAML. Оскільки стилі є ресурсами, їм діють самі правила визначення області, як і всіх інших ресурсів. Простіше кажучи, те, де ви оголошує стиль, впливає на те, де цей стиль може бути застосований. Наприклад, якщо оголосити стиль у кореневому елементі файлу XAML визначення програми, стиль може використовуватись у будь-якому місці програми.

Візуальні стани

Елементи управління завжди знаходяться у певному стані. Наприклад, коли вказівник миша переміщається над елементом управління, то вважається, що елемент управління знаходиться у звичайному стані MouseOver. Елемент керування без певного стану сприймається як елемент керування зі звичайним станом Normal. Стани розбиваються на групи, а згадані вище стани є частиною групи станів CommonStates. Більшість елементів управління мають дві групи станів: CommonStates і FocusStates. Для кожної групи станів, що застосовується до елемента управління, елемент управління завжди знаходиться в одному зі станів кожної групи, наприклад CommonStates.MouseOver та FocusStates.Unfocused. Елемент керування не може перебувати у двох різних станах у межах однієї групи, наприклад CommonStates.Normal та CommonStates.Disabled [8].

Загальні ресурси та теми

Типова програма WPF може мати кілька ресурсів інтерфейсу користувача, які застосовуються в рамках всього додатка. У сукупності цей набір ресурсів можна як тему докладання. WPF підтримує упаковку ресурсів інтерфейсу користувача у вигляді теми, використовуючи словник ресурсів, який інкапсулюється як клас ResourceDictionary.

Теми WPF задаються за допомогою механізмів стилізації та використання шаблонів, які WPF надає для налаштування відображення будь-якого елемента [25].

Ресурси теми WPF зберігаються у словниках запроваджених ресурсів. Ці словники ресурсів повинні бути впроваджені в підписане складання і можуть бути впроваджені або в ті ж зборки, що і сам код, або в паралельну складання. Для бібліотеки PresentationFramework.dll (складання, що містить елементи управління WPF) ресурси тем знаходяться в ряді паралельних збірок.

Тема стає останнім місцем для пошуку стилю елемента. Як правило, процес пошуку починається з проходу вгору по дереву елементів у пошуках відповідного ресурсу, потім виконується пошук у колекції ресурсів програми і, нарешті, в останню чергу здійснюється запит до системи. Це дає розробникам додатків можливість перевизначити стиль для будь-якого об'єкта на рівні дерева або додатка до досягнення теми [15].

Словники ресурсів, оформлені у вигляді окремих файлів, дозволяють повторно використовувати тему в кількох програмах. Також можна створити змінні теми, визначивши кілька словників ресурсів, які забезпечують одні й самі типи ресурсів, але з різними значеннями. Перевизначення цих стилів або інших ресурсів на рівні програми є рекомендованим способом зміни теми програми.

2.5. Методологія розробки

Модель життєвого циклу забезпечує парадигму розробки програмного

забезпечення, яка допомагає програмістам визначити прийнятні стратегії розробки. Методологія розробки програмного забезпечення має свій власний набір інструментів, процедур та методів для явного визначення та визначення життєвого циклу розробки програмного забезпечення [23].

Життєвий цикл розробки програмного забезпечення зазвичай включає такі етапи:

Етап 1. Збір та аналіз вимог:

Цей етап дає більш чітке уявлення про загальні масштаби проекту, розкриваючи передбачувані проблеми, можливості та директиви, які поклали початок проекту. На цьому етапі здійснюється додаткове планування стандартних вимог до впевненості та обізнаності про ризики.

Етап 2. ТЕО:

Повинне існувати техніко-економічне обґрунтування, оскільки важливо визначити та задокументувати потреби у програмному забезпеченні. Це означає всі частини, які необхідно запрограмувати та спроектувати протягом життєвого циклу проекту. В основному існує п'ять типів техніко-економічних обґрунтувань: технічні, практичні, юридичні, економічні та графічні.

Етап 3. Проектування:

На цьому етапі повинен бути підготовлений документ специфікації вимог разом з проектною документацією системи та програмного забезпечення.

Це допомагає визначити загальну архітектуру системи. Цей етап проектування є входом для наступного етапу моделі.

Етап 4. Кодування:

На етапі кодування завдання розбиваються на блоки або модулі та призначаються різним розробникам. Це найдовший етап життєвого циклу розробки програмного забезпечення.

Розробники починають будувати всю систему з написання коду на

штучній мові на свій вибір. Програмний код має бути отриманий і реалізований за допомогою інструментів програмування, таких як компілятори, інтерпретатори та налагоджувачі, дотримуючись певних попередньо визначених інструкцій щодо кодування.

Етап 5. Тест:

Етап тестування дуже важливий, щоб переконатися, що вимоги замовника виконуються, а отже, вся програма працює відповідно.

В ідеалі цей процес триває, доки програма не стане безпомилковою, стабільною та працюватиме відповідно до потреб системи.

Крок 6. Встановлення/розгортання:

На основі відгуків, наданих керівником проекту, випускається остаточна версія програми та додатково тестується на наявність проблем із розгортанням, якщо це необхідно в процесі.

Етап 7. Технічне обслуговування:

Метою цього етапу SDLC є переконатися, що вимоги все ще виконуються, і тому система працює відповідно до специфікацій, наданих на першому етапі. Якість вашого дизайну визначає ефективність вашої системи. Тому кожен етап може бути навмисно розбитий на різні етапи, і для цього необхідно підготувати документацію [5].

Відображення результату роботи програми.

Використання життєвого циклу розробки програмного забезпечення (SDLC) під час розробки пакетів є дуже важливим. По-перше, він забезпечує основу для планування проекту, планування та оцінки, підвищуючи видимість проекту [32].

Для всіх зацікавлених сторін у процесі події. По-друге, SDLC збільшує швидкість розробки, запроваджуючи механізм відстеження та управління проектом. Крім того, SDLC забезпечує основу для типового набору дій і результатів, зменшує ризики проекту та накладні витрати на планування управління проектами, а також покращує критично важливі для бізнесу

відносини з клієнтами.

Тому найпоширенішими основними методологіями розробки програмного забезпечення є модель водоспаду, інкрементна модель, V-модель, спіральна модель, гнучка модель і Scrum.

Додаткові методології були застосовані під час розробки настільного додатку. Це включає як подію, так і подальшу підтримку продукту. Вважається завершеним, коли виконано всі вимоги.

Інкрементні моделі не є автономними моделями. По суті, це серія водоспадних циклів. За допомогою цього методу кожен цикл є дійсним, оскільки це етап обслуговування попереднього випуску програмного забезпечення. Поступові зміни моделі дозволяють дублювати цикли розробки. У цьому випадку наступний цикл може початися раніше, ніж завершиться попередній цикл [19].

На початку проекту матеріали поділяються на різні групи. Під час заходу кожна група буде дотримуватися моделі SDLC. Метод SDLC є ітеративним, додаючи нові функції з кожним новим випуском, доки не буде досягнуто всіх необхідних вимог. Покрокова стратегія SDLC для планування, впровадження та тестування проекту поетапно (з невеликими доповненнями кожного разу) до початку циклу подій. Ця модель поєднує в собі елементи каскадної моделі з прототипуванням [11].

Інкрементні моделі працюють за принципом каскадних моделей з перекриттям. Це призводить до раннього виробництва робочих функцій продукту. Це може включати цілий набір попередньо сформованих вимог, виконаних у вигляді серії менших проектів. Крім того, реалізація проекту може початися з формулювання спільних цілей, які потім реалізуються командою розробників.

Це вдосконалення в рамках каскадної моделі однаково ефективно при використанні для дуже великих і малих проектів.

Ця модель служить для того, щоб «згладити кути», а не одразу

розгортати користувачам абсолютно нову систему. Проект розділений на кілька компонентів, кожен з яких розроблений і побудований незалежно від іншої сторони (складання). Кожен компонент доставляється клієнту, як тільки він буде готовий, що дозволяє клієнту негайно почати використовувати продукт і уникнути тривалої розробки. Це також скорочує час очікування результатів за рахунок великих інвестиційних витрат. Перевагами інкрементальної моделі є гнучкість і випуск робочих програм на ранніх стадіях життєвого циклу продукту. Змінити обсяг проекту та вимоги відносно дешево. З невеликими ітераціями тестування та редагування стають легшими. Ви можете визначити ризики. Кожна ітерація є контрольним показником для конкретного проекту. Недоліком інкрементальної моделі є те, що кожна фаза ітерації є фіксованою. Також можуть бути проблеми з архітектурою системи, оскільки не всі вимоги для всього життєвого циклу програмного забезпечення були зібрані заздалегідь [33].

Інкрементальні моделі використовуються з чіткими та зрозумілими вимогами, які реалізуються поетапно.

2.6. Опис використовуваних технологій та мов програмування

C# (вимовляється як «сі шарп» або «сі шарп») — це мова програмування сімейства мов .NET. Він може створювати широкий спектр програм і є об’єктно-орієнтованим. C# була представлена корпорацією Майкрософт 26 червня 2000 року і стала популярною мовою програмування загального призначення в 21 столітті[9].

Ця мова є еволюцією сімейства мов C. Він також містить функції альтернативних мов програмування, таких як Java і Delphi. Отже, базовий синтаксис Java і C# виглядає дуже знайомим. Код навмисно нагадує мову C++. C# бере свій початок у спадщині C++, тому він підтримує певні функції або поводиться певним чином. Сучасні особливості мови (наприклад, LINQ -

Language Integrated Query) і асинхронне програмування (asynchronous) не тільки присутні в C#, але й роблять мову відносно унікальною [27].

Станом на 2021 рік поточною стабільною версією мови є C# 10.0, випущена в 2021 році як частина фреймворку .NET 6.0.

C# — це керована мова, для якої потрібна реалізація .NET CLR. Під час виконання програми CLR відповідає за керування пам'яттю, створення винятків, функції GC тощо. Компілятор C# компілює код не безпосередньо до машинної мови, а до проміжної мови (IL). Цей IL розуміє CLR. JIT-компілятор CLR компілює та виконує код IL у машинний код метод за методом. Для запуску C# потрібен CLR. Усі нові операційні системи Windows постачаються з попередньо встановленою версією CLR, яка доступна на старих системах через оновлення операційної системи [10].

Незважаючи на ці обмеження, C# має доступ до FCL, тому він може зробити багато. Наприклад, FCL дозволяє створювати настільні програми за допомогою WPF, WinForms і консольні програми. В Інтернеті ви можете використовувати програми ASP.NET. Він має ADO.NET Entity Framework і LINQ для обробки даних. Останні функції .NET включають підтримку Windows Store і Windows Phone. Масштабовані хмарні програми можна створювати за допомогою Windows Azure. Це лише деякі з доступних функцій. Універсальна мова програмування загального призначення, така як C#, дозволяє вам робити більше, ніж звичайна мова програмування [29].

2.7. Розроблений опис ПЗ

2.7.1. Використані апаратні ресурси

Для розробки цього програмного забезпечення використовувався ПК з наступними характеристиками:

- 1,8 GHz 2-ядерний процесор Intel Core i5;
- 8 ГБ 1600 МГц DDR3;
- Intel HD Graphics 6000 1536 МБ;
- OS Microsoft Windows 10;
- 13" monitor;
- keyboard, mouse;
- access to the Internet.

2.7.2. Використані програмні ресурси

Додаток Bores написаний об’єктно-орієнтованою мовою програмування C# в Windows Studio IDE.

Система сумісна з серверами Windows 10.

Для роботи серверної програми необхідна інсталяція платформи .NET Framework v4.7.2.

2.8. Висновок

У другому розділі розглянуто існуючі рішення, обрано платформу розробки, описано розробку застосунка, представлені основні алгоритми, індикатори та структуру, наведено довідкову інформацію та функціональні можливості програми, описано інтерфейс користувача системи.

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ ПРОГРАМИ

3.1 Авторизція та налаштування всіх покажчиків для роботи

На початку роботи потрібно авторизуватися. Для авторизації вводимо ApiKey, SecretKey та User name(Рис.3.1).

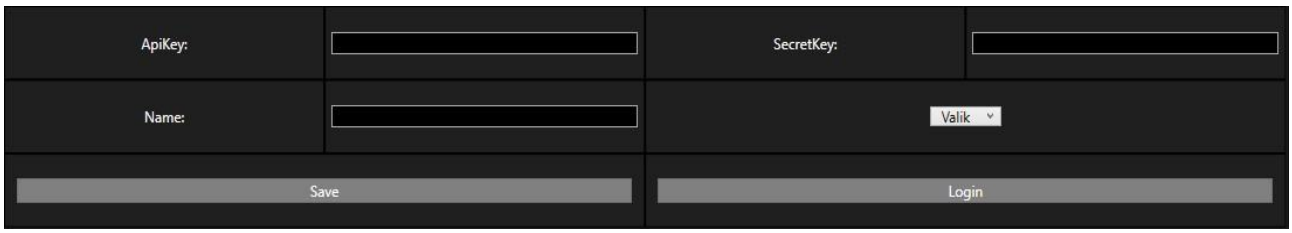


Рис .3.1. Авторизация

Реалізована можливість виставляти кількість свічок на графіку(Рис.3.2).

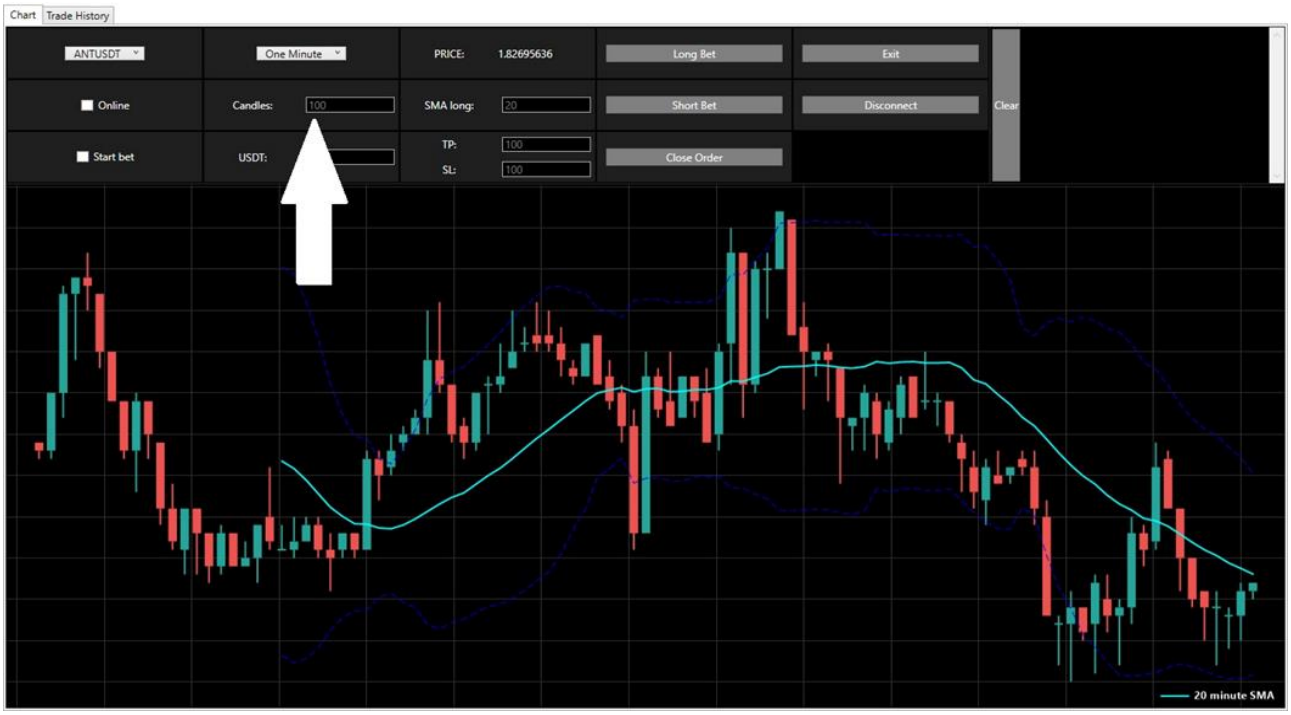


Рис.3.2. Кількість свічок

Задаємо інтервал для індекатора SMA(Рис.3.3).

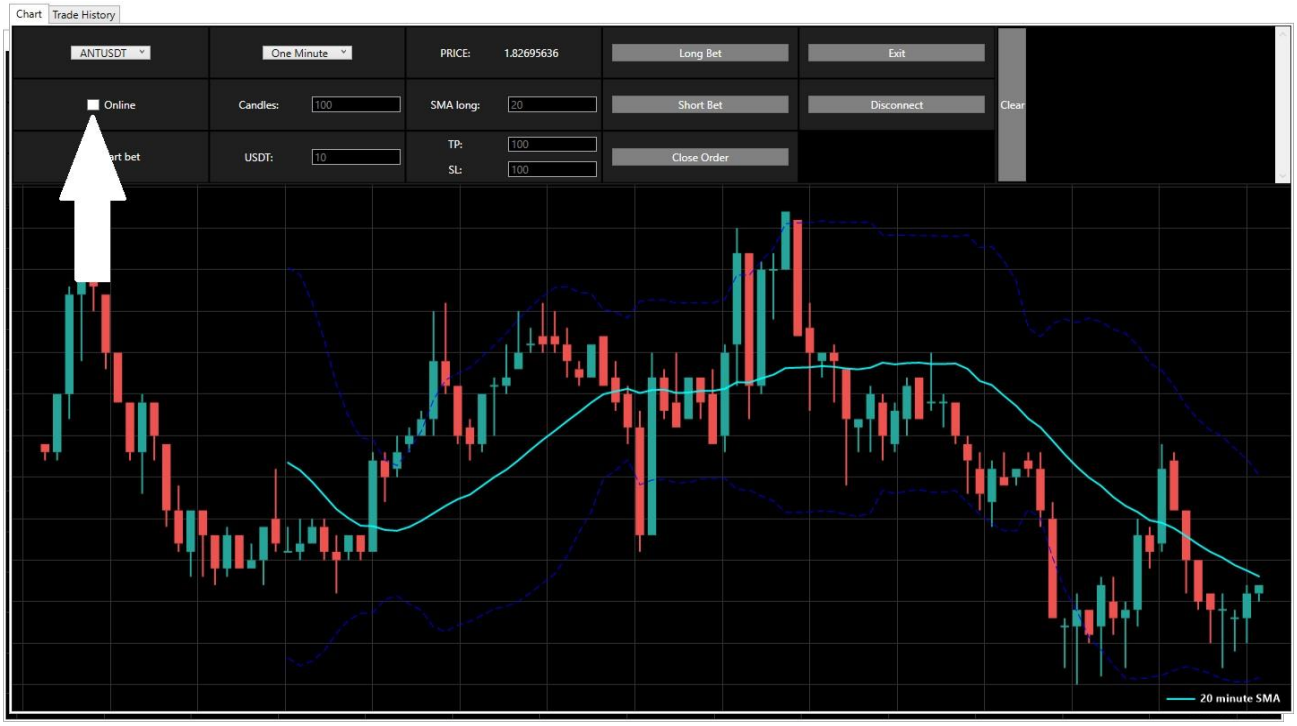


Рис.3.3. Задача інтервала

У полі USDT вказуємо суму ордеру, який ми відкриваємо(Рис. 3.4).

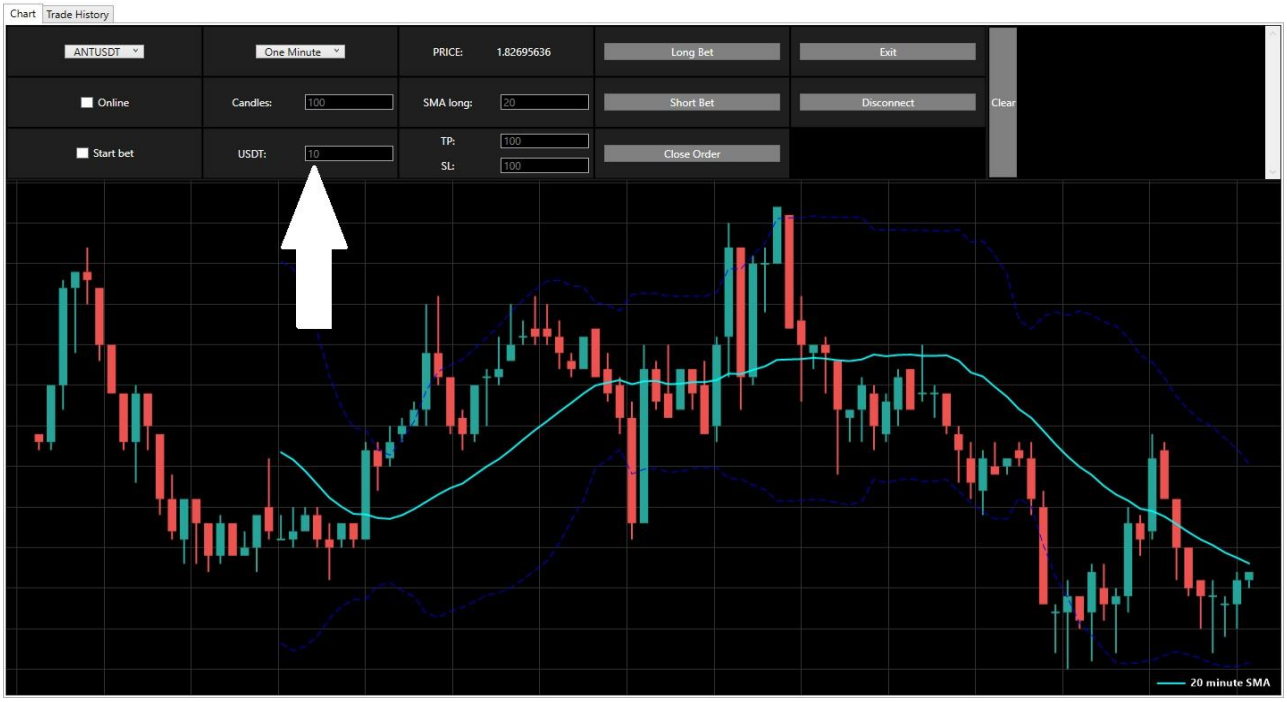


Рис. 3.4. Вказуємо суму ордеру

Можливість вибору роботи з графіком online та offline(Рис.3.4).

Обираємо валютну пару для торгів біржою(Рис.3.6).

Обираємо певний інтервал свічок за нашим рішенням(Рис.3.7).

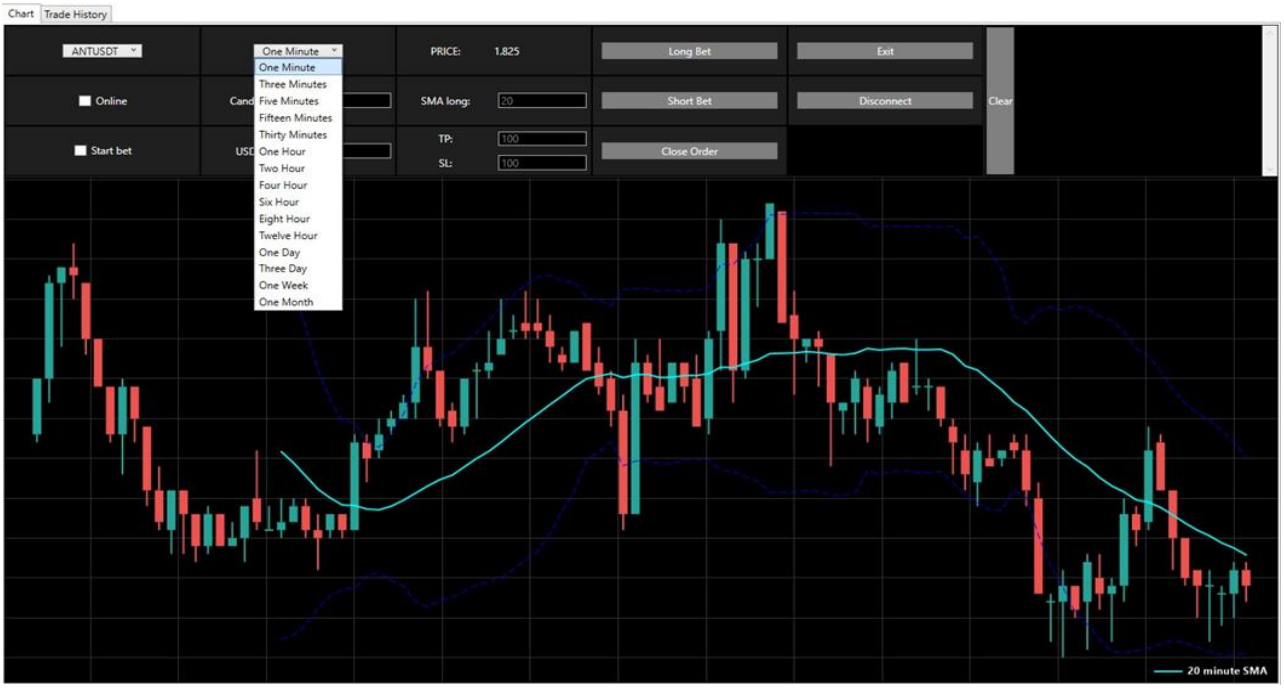


Рис.3.6. Вибір валютної пари

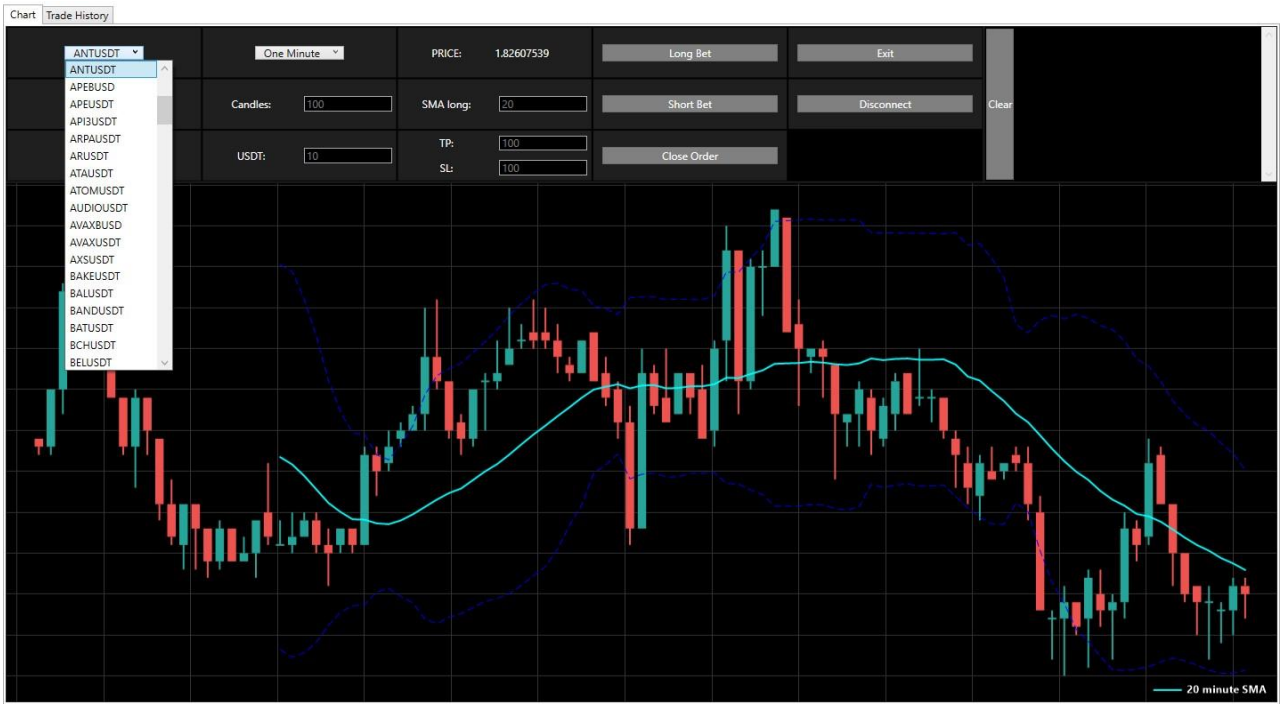


Рис. 3.7 Задаємо take profit та stop loss

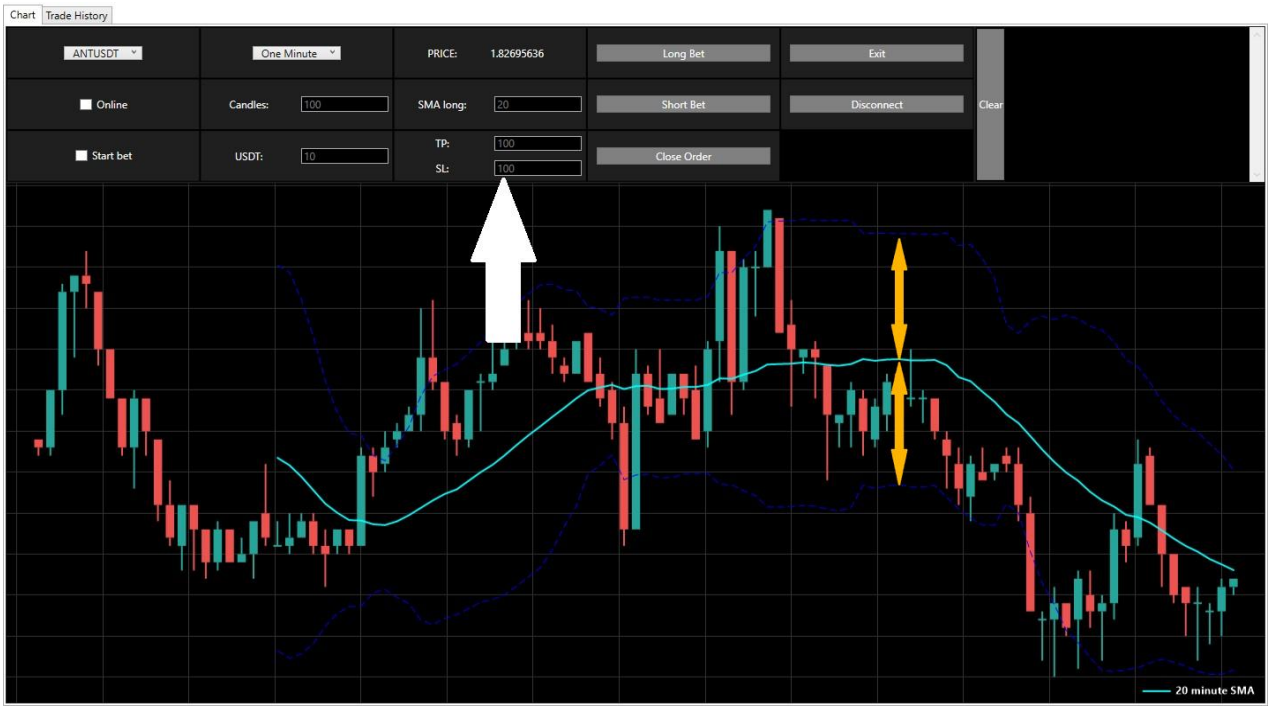


Рис.3.8. Задаем take profit и stop loss

Задається відсоткове співвідношення цін від лінії індикатора SMA до ліній верхнього індикатора Bolinger і нижнього індикатора Bolinger (Відображено на графіку помаранчевими стрілками) (Рис.3.8)

3.2. Старт бота та пояснення алгоритму дій бота

Запускаємо роботу алгоритму боту (Рис. 3.9).

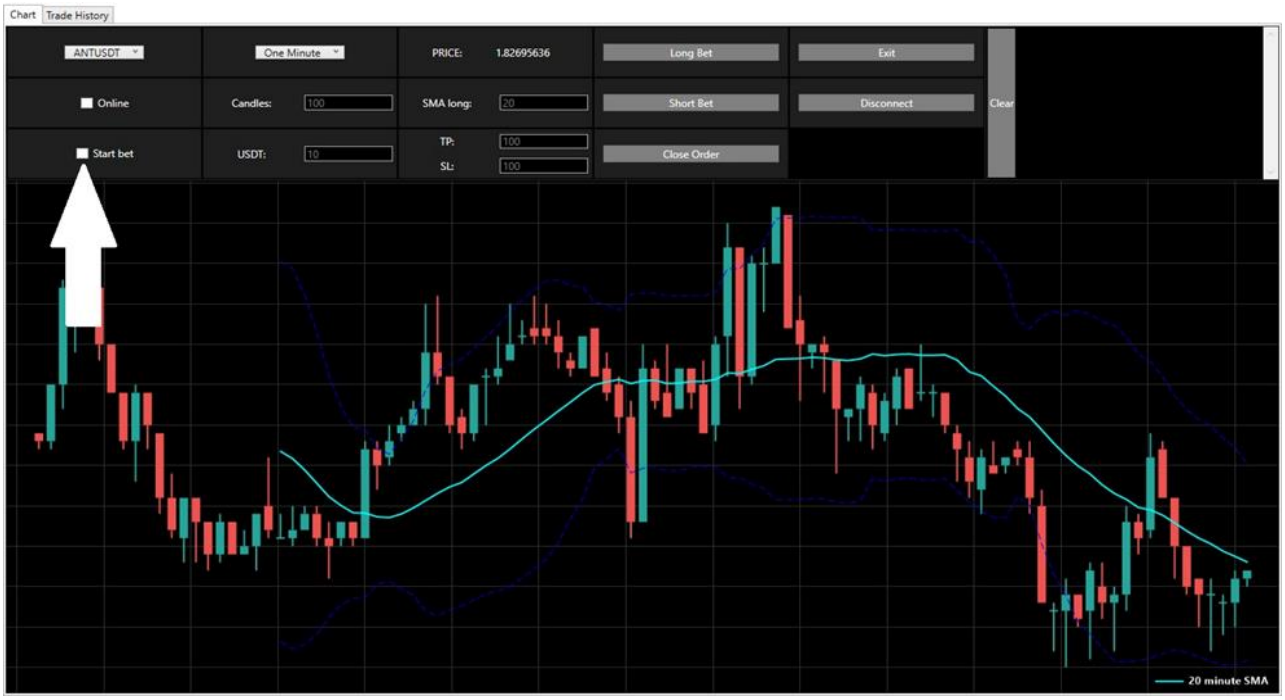


Рис. 3.9. Запускаємо дію бота

Якщо поточна ціна монети перетинає лінію індикатора SMA відкривається ордер на покупку (Рис.3.10).

Рис.3.10. Ордер на покупку

Якщо поточна ціна монети перетинає значення take profit і stop loss (лініяї Болліджера)(Рис.3.11) відкривається ордер на продаж [2] (Рис.3.12).

Рис.3.11. Лінії Болліджера

Рис.3.12. Відображення відкриття та закриття ордерів на графіку

Меню для самостійного укладання угод користувачем без алгоритму(Рис.3.13). Також маємо статистику торгівлі (Рис.3.14).

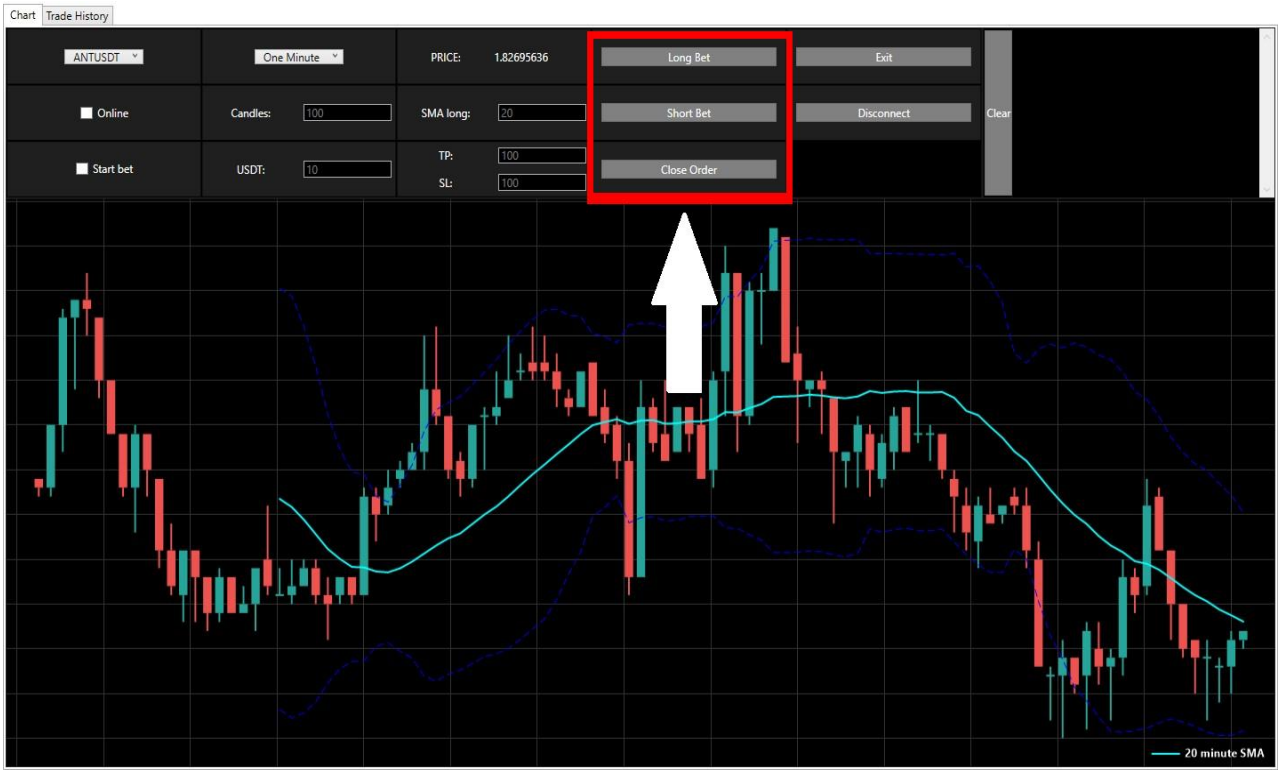


Рис.3.13. Меню для
самостійного укладання угод

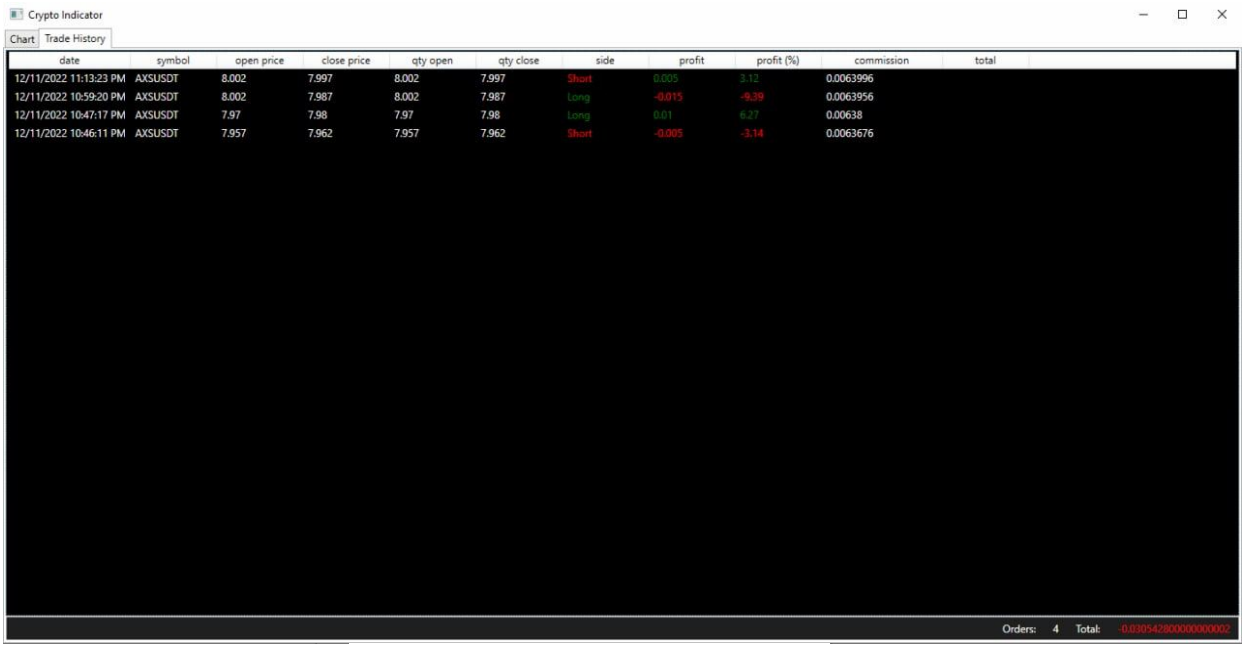


Рис.3.14. Статистика торгів
для певної монети

Всі помилки виводяться в окремий блок програми, де їх можна копіювати(Рис.3.15).

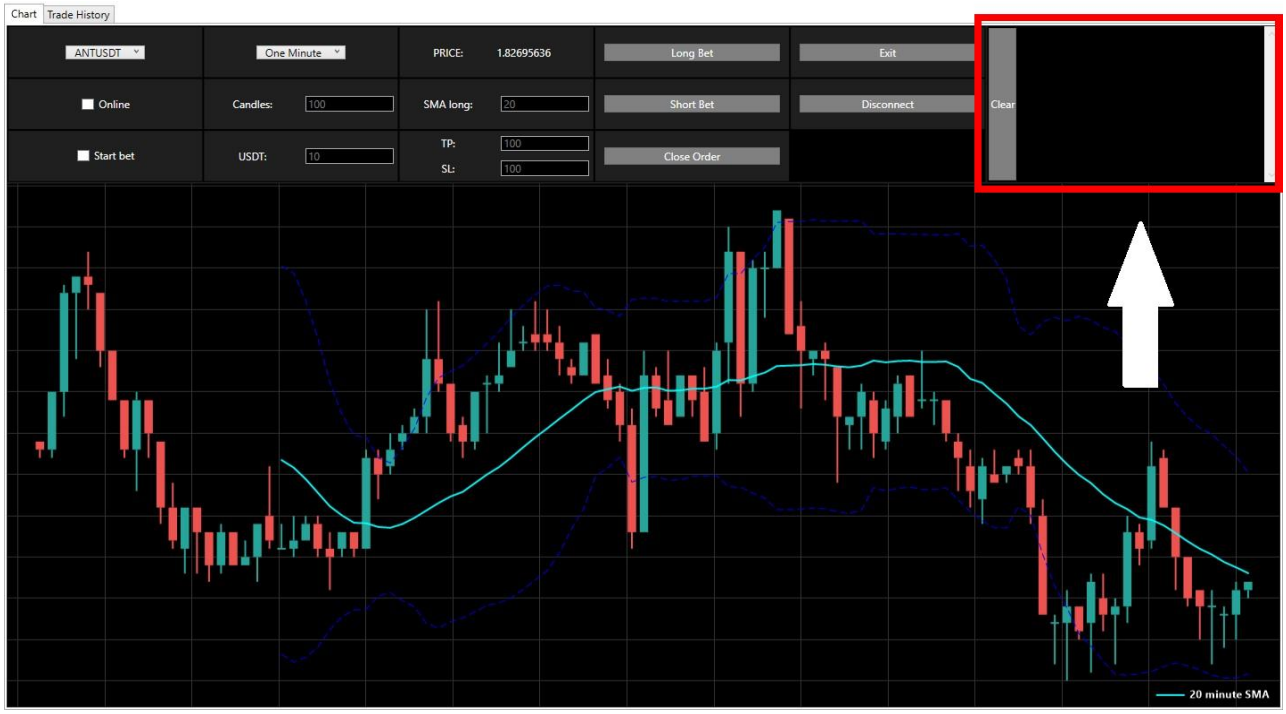


Рис.3.15. Відображення помилок

3.3. Висновок

У відповідності до поставлених завдань було здійснено: дослідження поставленої задачі та предметної області, здійснено дослідження існуючих методів та моделей прогнозування криптовалют, а саме лінії Боллінжера та індикатор SMA. Реалізоване ПЗ згідно всіх вимог.

ВИСНОВКИ

Метою кваліфікаційної роботи є створення програми бота для атоматичних торгів на біржі Binance з використанням алгоритму на основі причинно-наслідкових зв’язків із ключовими індикаторами.

ПЗ Crypto Indicator являє собою систему, яка дозволяє автоматично відкривати та закривати ордери на купівлю, або продаж криптовалюти на біржі Binance під ОС Windows. Середовище розробки — Visual Studio з мовою програмування C#.

Віконна програма Crypto Indicator написана об’єктно-орієнтованою мовою програмування C# в Visual Studio IDE. Система сумісна з Windows 10 і пізнішими версіями.

Crypto Indicator має конфігураційні файли JSON для зберігання інформації для входу до акаунту на біржі. Для всіх конкретних даних користувача було вирішено використовувати JSON, який зберігає інформацію в спеціальних файлах та папках.

Додаток Crypto Indicator складається з кількох основних частин, які мають різні цілі та функції. Простота інтерфейсу дає можливість користуватися утилітою, як молоді, так і людям старшого віку. Додаток разом з Binance утворює цілісну працездатну систему.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Binance. https://www.binance.com

2. Bollinger. https://blog.roboforex.com/ru/blog/2019/08/07/polosy-bollindzhera-opisanie-indikatora-bollinger-bands/#:~:text=ML%20%3D%20SUM(CLOSE%20N)%2F,CLOSE%20цена%20закрытия%20свечи

3. Binance.Net. https://jkorf.github.io/Binance.Net/Examples.html

4. Mark J. Price. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition. 2019

5. John Sharp. Microsoft Visual C# Step by Step (8th Edition) (Developer Reference). 2018

6. Tony Gaddis. Starting out with Visual C# (4th Edition). 2017

7. Code Quickly. Learn C# Quickly: A Complete Beginner’s Guide to Learning C#, Even If You’re New to Programming. 2016

8. Stephen Cleary. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. 2018

9. Jon Skeet. C# in Depth. 2016

10. C.P.A. Inc, CyberPunk Architects. The Programmers Code: A Deep Dive Into Mastering Computer Programming Including Python, C, C++, C#, Html Coding, Raspberry Pi3, And Black Hat Hacking. 2020

11. Ian Griffiths. Programming C# 8.0: Build Cloud, Web, and Desktop Applications. 2019

12. Huw Collingbourne. The Little Book Of C# Programming: Learn To Program C-Sharp For Beginners. 2019

13. Joyce Farrell. Microsoft Visual C#: An Introduction to Object-Oriented Programming (MindTap Course List). 2017

14. Joseph Albahari. C# 8.0 in a Nutshell: The Definitive Reference. 2020

15. Ockert J. du Preez, Sunny Sharma. Visual Studio 2019 In Depth: Discover

and make use of the powerful features of the Visual Studio 2019 IDE to develop better and faster mobile, web, and desktop applications. 2019

16. Benjamin Smith. C#: A Comprehensive Beginner's Guide to Learn about the Realms of C# from A-Z. 2020

17. Jon Skeet. C# in Depth. 2008

18. Gabriel Baptista, Francesco Abbruzzese. Hands-On Software Architecture with C# 8 and .NET Core 3: Architecting software solutions using microservices, DevOps, and design patterns for Azure Cloud. 2019

19. Sean Burns. Hands-On Network Programming with C# and .NET Core: Build robust network applications with C# and .NET Core. 2019

20. John Paul Mueller, Bill Sempf, Chuck Sphar. C# 7.0 All-in-One For Dummies. 2017

21. Joydip Kanjilal. Mastering C# 8.0: Master C# skills with plentiful code examples (English Edition). 2019

22. Jeffrey Richter. CLR via C# (Developer Reference). 2012

23. Joyce Farrell. Microsoft Visual C#: An Introduction to Object-Oriented Programming (MindTap Course List). 2017

24. Gary Mclean. Adaptive Code via C#: Agile coding with design patterns and SOLID principles (Developer Reference). 2014

25. Benjamin Perkins, Jacob Vibe Hammer, Jon D. Reid. Beginning C# 7 Programming with Visual Studio 2017. 2018

26. DENNIS SHARP. C# Advanced Topics, Features and Programming Techniques: Take Your C# Skills and Expertise to the Next Level (Advanced Level). 2019

27. Life -Style Academy. C#: C# CRASH COURSE - Beginner's Course To Learn The Basics Of C# Programming Language: (c#, c programming, c, java, python, angularjs, c++ programming). 2016

28. Wally Parsons. C# Programming: Ultimate Guide For Advanced Users

To Learn C# Programming (3 books in 1). 2019

29. Aristides S. Bouras. C# for Tweens and Teens (Black & White Edition): Learn Computational and Algorithmic Thinking. 2017

30. Capers Jones. Software Methodologies: A Quantitative Guide. 2017

31. David Harned. Hands-On Agile Software Development with JIRA: Design and manage software projects using the Agile methodology. 2018

32. David Kung. Object-Oriented Software Engineering: An Agile Unified Methodology. 2013

33. Michael E. Bays. Software Release Methodology. 1999

34. Обзор Bitsgap 2022. https://www.bloggersideas.com/ru/bitsgap-review

35. Обзор Cryptohopper. https://tradingbot.info/ua/obzor-cryptohopper/

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
using CryptoIndicator.Binance;

using CryptoIndicator.Errors;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using Newtonsoft.Json;

using Binance.Net.Enums;

using ScottPlot;

using System.Drawing;

using Binance.Net.Objects.Models.Spot;

using ScottPlot.Plottable;

using CryptoIndicator.ConnectDB;

using System.Windows.Threading;

using Binance.Net.Objects.Models.Futures;

using CryptoIndicator.Interval;

using CryptoIndicator.Model;

using CryptoIndicator.Objects;

using System.Data.Entity;

using System.Runtime.InteropServices;

using Binance.Net.Clients;

using Binance.Net.Objects;

using CryptoExchange.Net.Interfaces;

using System.Threading.Tasks;

using System.Windows.Documents;

using CryptoIndicator.Indicators;

using static CryptoIndicator.Indicators.SAR;
```

```
namespace CryptoIndicator
{
    public partial class MainWindow : Window
    {
        public List<Candle> Candles= new List<Candle>();
        public List<HistoryOrder> HistoryOrders = new List<HistoryOrder>();
        public List<BinanceFuturesOrder> BinanceFuturesOrders = new List<BinanceFuturesOrder>();
        public List<Symbol> Symbols = new List<Symbol>();
        public Variables variables { get; set; } = new Variables();
        public int LINE { get; set; } = 2;
        public string API_KEY { get; set; } = "";
        public string SECRET_KEY { get; set; } = "";
        public string CLIENT_NAME { get; set; } = "";
        public int COUNT_CANDLES { get; set; } = 200;
        public int SMA_LONG { get; set; } = 20;
        public decimal USDT_BET { get; set; } = 11;
        public double BOLINGER_TP { get; set; } = 100;
        public double BOLINGER_SL { get; set; } = 100;
        public Socket socket;
        public List<string> list_sumbols_name = new List<string>();
        public FinancePlot candlePlot;
        public ScatterPlot sma_long_plot;
        public ScatterPlot bolinger_lower;
        public ScatterPlot bolinger_upper;
        public ScatterPlot line_scatter;
        public ScatterPlot order_long_open_plot;
        public ScatterPlot order_long_close_plot;
        public ScatterPlot order_short_open_plot;
        public ScatterPlot order_short_close_plot;
        public ScatterPlot indicator_sar_long_plot;
        public ScatterPlot indicator_sar_short_plot;
        public List<ScatterPlot> order_long_lines_vertical = new List<ScatterPlot>();
        public List<ScatterPlot> order_long_lines_horisontal = new List<ScatterPlot>();
```

```
        public List<ScatterPlot> order_short_lines_vertical = new List<ScatterPlot>();
        public List<ScatterPlot> order_short_lines_horisontal = new List<ScatterPlot>();
        public KlineInterval interval_time = KlineInterval.OneMinute;
        public TimeSpan timeSpan = new TimeSpan(TimeSpan.TicksPerMinute);
        public List<HistoryOrder> history_order = new List<HistoryOrder>();

        public MainWindow()
        {
            InitializeComponent();
            Chart();
            Loaded += MainWindow_Loaded;
        }

        #region - Loaded -
        private void MainWindow_Loaded(object sender, RoutedEventArgs e)
        {
            ErrorWatcher();
            Clients();
            HISTORY_ORDER.ItemsSource = history_order;
            INTERVAL_TIME.ItemsSource = IntervalCandles.Intervals();
            INTERVAL_TIME.SelectedIndex = 0;
            LIST_SYMBOLS.ItemsSource = list_sumbols_name;
            this.DataContext = this;
            variables.PropertyChanged += Variables_PropertyChanged;
        }

        private void Variables_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            if(e.PropertyName == "IsDataBase")
            {
                if (variables.IsDataBase)
                {
                    //Create Table BinanceFuturesOrders
                    using (ModelBinanceFuturesOrder context = new ModelBinanceFuturesOrder())
```

```
                {
                    context.BinanceFuturesOrders.Create();
                }
                //Create Table HistoryOrders
                using (ModelHistoryOrder context = new ModelHistoryOrder())
                {
                    context.HistoryOrders.Create();
                }
                //Create Table Candles
                using (ModelCandle context = new ModelCandle())
                {
                    context.Candles.Create();
                }
            }
        }
    }
    #endregion

    #region - Open order, close order -

    public decimal quantity_bet;
    public long bet_order_id = 0;
    private void LongBet_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            string symbol = LIST_SYMBOLS.Text;
            if (bet_order_id == 0)
            {
                if (USDT_BET > 0m && variables.PRICE_SYMBOL > 0m)
                {
                    quantity_bet = RoundQuantity(USDT_BET / variables.PRICE_SYMBOL);
                    bet_order_id = Algorithm.Algorithm.OpenOrder(socket, symbol, quantity_bet,
PositionSide.Long);
                }
```

```
                }
            }
            catch (Exception ex)
            {
                ErrorText.Add(ex.ToString());
            }
        }
        private void ShortBet_Click(object sender, RoutedEventArgs e)
        {
            string symbol = LIST_SYMBOLS.Text;
            if (bet_order_id == 0)
            {
                if (USDT_BET > 0m && variables.PRICE_SYMBOL > 0m)
                {
                    quantity_bet = RoundQuantity(USDT_BET / variables.PRICE_SYMBOL);
                    bet_order_id = Algorithm.Algorithm.OpenOrder(socket, symbol, quantity_bet,
PositionSide.Short);
                }
            }
        }
        private void CloseOrder_Click(object sender, RoutedEventArgs e)
        {
            string symbol = LIST_SYMBOLS.Text;
            if (bet_order_id != 0) bet_order_id = Algorithm.Algorithm.CloseOrder(socket, symbol,
bet_order_id, quantity_bet);
        }

        private decimal RoundQuantity(decimal quantity)
        {
            Symbol symbol =  Symbols.First(item=>item.Name == LIST_SYMBOLS.Text);
            decimal quantity_final = 0m;
            if (symbol.StepSize == 0.001m) quantity_final = Math.Round(quantity, 3);
            else if (symbol.StepSize == 0.01m) quantity_final = Math.Round(quantity, 2);
            else if (symbol.StepSize == 0.1m) quantity_final = Math.Round(quantity, 1);
            else if (symbol.StepSize == 1m) quantity_final = Math.Round(quantity, 0);
```

```
    if (quantity_final < symbol.MinQuantity) return symbol.MinQuantity;
    return quantity_final;
}
#endregion

#region - Trede History -
private void TAB_CONTROL_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    History();
    double sum_total = 0;
    int count_orders = 0;
    if (variables.IsDataBase)
    {
        foreach (var it in ConnectHistoryOrder.Get())
        {
            history_order.Insert(0, it);
            sum_total += it.total;
            count_orders++;
        }
    }
    else
    {
        foreach (var it in HistoryOrders)
        {
            history_order.Insert(0, it);
            sum_total += it.total;
            count_orders++;
        }
    }
    COUNT_ORDERS.Content = count_orders;
    SUM_TOTAL.Content = sum_total;
    if (sum_total > 0) SUM_TOTAL.Foreground = System.Windows.Media.Brushes.Green;
    else if (sum_total < 0) SUM_TOTAL.Foreground = System.Windows.Media.Brushes.Red;
    HISTORY_ORDER.Items.Refresh();
```

```
        }

        private void History()
        {
            try
            {
                history_order.Clear();
                if (variables.IsDataBase)
                {
                    ConnectHistoryOrder.DeleteAll();
                }
                else
                {
                    HistoryOrders.Clear();
                }
                List<BinanceFuturesOrder> orders = new List<BinanceFuturesOrder>();
                if(variables.IsDataBase)
                {
                    orders = ConnectOrder.Get();
                }
                else
                {
                    orders = BinanceFuturesOrders;
                }
                int i = 0;
                foreach (var it in orders)
                {
                    if (it.PositionSide == PositionSide.Long && it.Side == OrderSide.Sell)
                    {
                        if(variables.IsDataBase)
                        {
                            ConnectHistoryOrder.Insert(new HistoryOrder(it.CreateTime, it.Symbol,
Convert.ToDouble(orders[i - 1].AvgPrice), Convert.ToDouble(it.AvgPrice), Convert.ToDouble(orders[i -
1].QuoteQuantityFilled), Convert.ToDouble(it.QuoteQuantityFilled), it.PositionSide));
                        }
```

```
                        else

                        {

                            HistoryOrders.Add(new HistoryOrder(it.CreateTime, it.Symbol,
Convert.ToDouble(orders[i - 1].AvgPrice), Convert.ToDouble(it.AvgPrice), Convert.ToDouble(orders[i -
1].QuoteQuantityFilled), Convert.ToDouble(it.QuoteQuantityFilled), it.PositionSide));

                        }

                    }

                    else if (it.PositionSide == PositionSide.Short && it.Side == OrderSide.Buy)

                    {

                        if (variables.IsDataBase)

                        {

                            ConnectHistoryOrder.Insert(new HistoryOrder(it.CreateTime, it.Symbol,
Convert.ToDouble(orders[i - 1].AvgPrice), Convert.ToDouble(it.AvgPrice), Convert.ToDouble(orders[i -
1].QuoteQuantityFilled), Convert.ToDouble(it.QuoteQuantityFilled), it.PositionSide));

                        }

                        else

                        {

                            HistoryOrders.Add(new HistoryOrder(it.CreateTime, it.Symbol,
Convert.ToDouble(orders[i - 1].AvgPrice), Convert.ToDouble(it.AvgPrice), Convert.ToDouble(orders[i -
1].QuoteQuantityFilled), Convert.ToDouble(it.QuoteQuantityFilled), it.PositionSide));

                        }

                    }

                    i++;

                }

            }

            catch (Exception c)

            {

                ErrorText.Add($"History {c.Message}");

            }

        }

        #endregion


        #region - Coordinate Orders -


        List<double> long_open_order_x = new List<double>();

        List<double> long_open_order_y = new List<double>();

        List<double> long_close_order_x = new List<double>();
```

```
List<double> long_close_order_y = new List<double>();
List<double> short_open_order_x = new List<double>();
List<double> short_open_order_y = new List<double>();
List<double> short_close_order_x = new List<double>();
List<double> short_close_order_y = new List<double>();
private void InfoOrderAsunc(DateTime start_time)
{
    try
    {
        long_open_order_x.Clear();
        long_open_order_y.Clear();
        long_close_order_x.Clear();
        long_close_order_y.Clear();
        short_open_order_x.Clear();
        short_open_order_y.Clear();
        short_close_order_x.Clear();
        short_close_order_y.Clear();
        bool check_one = false;
        string symbol = LIST_SYMBOLS.Text;
        if (symbol != "")
        {
            if (variables.IsDataBase) {
                ConnectOrder.DeleteAll();
            }
            else
            {
                BinanceFuturesOrders.Clear();
            }
            foreach (var it in Algorithm.AlgorithmBet.InfoOrder(socket, symbol, start_time))
            {

                if (it.PositionSide == PositionSide.Long && it.Side == OrderSide.Buy)
                {
                    long_open_order_x.Add(it.CreateTime.ToOADate());
```

```
                    long_open_order_y.Add(Decimal.ToDouble(it.AvgPrice));
                    check_one = true;
                    if (variables.IsDataBase) {
                        ConnectOrder.Insert(it);
                    }
                    else
                    {
                        BinanceFuturesOrders.Add(it);
                    }
                }
                else if (it.PositionSide == PositionSide.Long && it.Side == OrderSide.Sell
&& check_one)
                {
                    long_close_order_x.Add(it.CreateTime.ToOADate());
                    long_close_order_y.Add(Decimal.ToDouble(it.AvgPrice));
                    if (variables.IsDataBase)
                    {
                        ConnectOrder.Insert(it);
                    }
                    else
                    {
                        BinanceFuturesOrders.Add(it);
                    }
                }
                else if (it.PositionSide == PositionSide.Short && it.Side == OrderSide.Sell)
                {
                    short_open_order_x.Add(it.CreateTime.ToOADate());
                    short_open_order_y.Add(Decimal.ToDouble(it.AvgPrice));
                    check_one = true;
                    if (variables.IsDataBase)
                    {
                        ConnectOrder.Insert(it);
                    }
                    else
                    {
```

```
                                BinanceFuturesOrders.Add(it);
                            }
                        }
                        else if (it.PositionSide == PositionSide.Short && it.Side == OrderSide.Buy
&& check_one)
                        {
                            short_close_order_x.Add(it.CreateTime.ToOADate());
                            short_close_order_y.Add(Decimal.ToDouble(it.AvgPrice));
                            if (variables.IsDataBase)
                            {
                                ConnectOrder.Insert(it);
                            }
                            else
                            {
                                BinanceFuturesOrders.Add(it);
                            }
                        }
                    }
                }
            }
            catch (Exception c)
            {
                ErrorText.Add($"InfoOrderAsunc {c.Message}");
            }
        }
        #endregion

        #region - Event Text Changed -
        private void COUNT_CANDLES_TextChanged(object sender, TextChangedEventArgs e)
        {
            ReloadChart();
        }

        private void INTERVAL_TIME_DropDownClosed(object sender, EventArgs e)
        {
```

```
    int index = INTERVAL_TIME.SelectedIndex;
    interval_time = IntervalCandles.Intervals()[index].interval;
    timeSpan = new TimeSpan(IntervalCandles.Intervals()[index].timespan);
    ReloadChart();
}
#endregion

#region - Event SMA -

private void SMA_LONG_TextChanged(object sender, TextChangedEventArgs e)
{
    ReloadSmaChart();
}
#endregion

#region - Load Chart -

public List<OHLC> list_candle_ohlc = new List<OHLC>();
private void LIST_SYMBOLS_DropDownClosed(object sender, EventArgs e)
{
    ReloadChart();
}
private void LoadingCandlesToChart()
{
    try
    {
        string symbol = LIST_SYMBOLS.Text;
        if (symbol != "")
        {
            list_candle_ohlc.Clear();
            List<Candle> list_candles = new List<Candle>();
            if (variables.IsDataBase) list_candles = ConnectCandle.Get();
            else list_candles = Candles;
            foreach (Candle it in list_candles)
```

```
                    {
                        list_candle_ohlc.Add(new OHLC(it.Open, it.High, it.Low, it.Close,
it.DateTime, TimeSpan.FromTicks(it.TimeSpan)));
                    }
                    InfoOrderAsunc(list_candle_ohlc[0].DateTime);
                }
            }
            catch (Exception c)
            {
                ErrorText.Add($"LoadingCandlesToChart {c.Message}");
            }
        }
        private void ReloadChart()
        {
            if (socket != null && SMA_LONG > 1 && COUNT_CANDLES > 0 && COUNT_CANDLES > SMA_LONG &&
COUNT_CANDLES < 500)
            {
                StopAsync();
                if (variables.IsDataBase)
                {
                    ConnectCandle.DeleteAll();
                }
                else {
                    Candles.Clear();
                }
                LoadingCandlesToDB();
                if (variables.ONLINE_CHART) StartKlineAsync();
                LoadingCandlesToChart();
                LoadingChart();
                plt.Plot.AxisAuto();
                plt.Refresh();
            }
        }
        private void ReloadSmaChart()
        {
```

```
            if (socket != null && SMA_LONG > 1 && COUNT_CANDLES > 0 && COUNT_CANDLES > SMA_LONG &&
COUNT_CANDLES < 500 && SMA_LONG < list_candle_ohlc.Count - 1)

            {

                plt.Plot.Remove(sma_long_plot);

                plt.Plot.Remove(bolinger_lower);

                plt.Plot.Remove(bolinger_upper);

                sma_long = candlePlot.GetBollingerBands(SMA_LONG);

                sma_long_plot = plt.Plot.AddScatterLines(sma_long.xs, sma_long.ys, Color.Cyan, 2,
label: SMA_LONG + " minute SMA");

                sma_long_plot.YAxisIndex = 1;

                bolinger_lower = plt.Plot.AddScatterLines(sma_long.xs, sma_long.lower,
Color.LightGreen, lineStyle: LineStyle.Dash);

                bolinger_lower.YAxisIndex = 1;

                bolinger_upper = plt.Plot.AddScatterLines(sma_long.xs, sma_long.upper,
Color.LightGreen, lineStyle: LineStyle.Dash);

                bolinger_upper.YAxisIndex = 1;

                plt.Refresh();

            }

        }

        public (double[] xs, double[] ys, double[] lower, double[] upper) sma_long;

        private void LoadingChart()

        {

            if (COUNT_CANDLES > 0)

            {

                if (SMA_LONG > 1 && SMA_LONG < list_candle_ohlc.Count - 1)

                {



                    plt.Plot.Remove(candlePlot);

                    plt.Plot.Remove(sma_long_plot);

                    plt.Plot.Remove(bolinger_lower);

                    plt.Plot.Remove(bolinger_upper);

                    plt.Plot.Remove(order_long_open_plot);

                    plt.Plot.Remove(order_long_close_plot);

                    plt.Plot.Remove(order_short_open_plot);

                    plt.Plot.Remove(order_short_close_plot);

                    //plt.Plot.Remove(indicator_sar_long_plot);
```

```
                    //plt.Plot.Remove(indicator_sar_short_plot);

                    // Candles

                    candlePlot = plt.Plot.AddCandlesticks(list_candle_ohlc.ToArray());

                    //candlePlot.YAxisIndex = 1;

                    //// Sma

                    sma_long = candlePlot.GetBollingerBands(SMA_LONG);

                    sma_long_plot = plt.Plot.AddScatterLines(sma_long.xs, sma_long.ys, Color.Orange,
2, label: SMA_LONG + " candles SMA");

                    sma_long_plot.YAxisIndex = 1;

                    //// Bolinger lower

                    bolinger_lower = plt.Plot.AddScatterLines(sma_long.xs, sma_long.lower,
Color.LightGreen, lineStyle: LineStyle.Dash);

                    bolinger_lower.YAxisIndex = 1;

                    //// Bolinger upper

                    bolinger_upper = plt.Plot.AddScatterLines(sma_long.xs, sma_long.upper,
Color.LightGreen, lineStyle: LineStyle.Dash);

                    bolinger_upper.YAxisIndex = 1;

                    //// Orders

                    if (order_long_lines_vertical.Count > 0) foreach (var it in
order_long_lines_vertical) plt.Plot.Remove(it);

                    if (order_long_lines_horisontal.Count > 0) foreach (var it in
order_long_lines_horisontal) plt.Plot.Remove(it);

                    if (order_short_lines_vertical.Count > 0) foreach (var it in
order_short_lines_vertical) plt.Plot.Remove(it);

                    if (order_short_lines_horisontal.Count > 0) foreach (var it in
order_short_lines_horisontal) plt.Plot.Remove(it);




                    if (long_close_order_x.Count != 0 && long_close_order_y.Count != 0)

                    {

                        order_long_close_plot = plt.Plot.AddScatter(long_close_order_x.ToArray(),
long_close_order_y.ToArray(), color: Color.Orange, lineWidth: 0, markerSize: 10, markerShape:
MarkerShape.eks);

                        order_long_close_plot.YAxisIndex = 1;

                        order_long_lines_vertical.Clear();

                        for (int i = 0; i < long_close_order_x.Count; i++)

                        {

                            double[] x = { long_open_order_x[i], long_open_order_x[i] };

                            double[] y = { long_open_order_y[i], long_close_order_y[i] };
```

```
                            ScatterPlot scatter = plt.Plot.AddScatterLines(x, y, Color.Orange,
lineStyle: LineStyle.Dash);

                            scatter.YAxisIndex = 1;

                            order_long_lines_vertical.Add(scatter);

                        }

                        order_long_lines_horisontal.Clear();

                        for (int i = 0; i < long_close_order_x.Count; i++)

                        {

                            double[] x = { long_open_order_x[i], long_close_order_x[i] };

                            double[] y = { long_close_order_y[i], long_close_order_y[i] };

                            ScatterPlot scatter = plt.Plot.AddScatterLines(x, y, Color.Orange,
lineStyle: LineStyle.Dash);

                            scatter.YAxisIndex = 1;

                            order_long_lines_horisontal.Add(scatter);

                        }

                    }

                    if (long_open_order_x.Count != 0 && long_open_order_y.Count != 0)

                    {

                        order_long_open_plot = plt.Plot.AddScatter(long_open_order_x.ToArray(),
long_open_order_y.ToArray(), color: Color.Green, lineWidth: 0, markerSize: 8);

                        order_long_open_plot.YAxisIndex = 1;

                    }

                    if (short_close_order_x.Count != 0 && short_close_order_y.Count != 0)

                    {

                        order_short_close_plot = plt.Plot.AddScatter(short_close_order_x.ToArray(),
short_close_order_y.ToArray(), color: Color.Orange, lineWidth: 0, markerSize: 10, markerShape:
MarkerShape.eks);

                        order_short_close_plot.YAxisIndex = 1;

                        order_short_lines_vertical.Clear();

                        for (int i = 0; i < short_close_order_x.Count; i++)

                        {

                            double[] x = { short_close_order_x[i], short_close_order_x[i] };

                            double[] y = { short_open_order_y[i], short_close_order_y[i] };

                            ScatterPlot scatter = plt.Plot.AddScatterLines(x, y, Color.Orange,
lineStyle: LineStyle.Dash);

                            scatter.YAxisIndex = 1;

                            order_short_lines_vertical.Add(scatter);
```

```
                        }

                        order_short_lines_horisontal.Clear();

                        for (int i = 0; i < short_close_order_x.Count; i++)

                        {

                            double[] x = { short_open_order_x[i], short_close_order_x[i] };

                            double[] y = { short_open_order_y[i], short_open_order_y[i] };

                            ScatterPlot scatter = plt.Plot.AddScatterLines(x, y, Color.Orange,
lineStyle: LineStyle.Dash);

                            scatter.YAxisIndex = 1;

                            order_short_lines_horisontal.Add(scatter);

                        }

                    }

                    if (short_open_order_x.Count != 0 && short_open_order_y.Count != 0)

                    {

                        order_short_open_plot = plt.Plot.AddScatter(short_open_order_x.ToArray(),
short_open_order_y.ToArray(), color: Color.DarkRed, lineWidth: 0, markerSize: 8);

                        order_short_open_plot.YAxisIndex = 1;

                    }



                    //SAR Sar = new SAR();

                    //(List < SarInfo > SarLong, List<SarInfo> SarShort) =
Sar.Calculate(list_candle_ohlc);



                    //List<double> Xlong = SarLong.Select(item => item.X).ToList();

                    //List<double> Ylong = SarLong.Select(item => item.Y).ToList();

                    //List<double> Xshort = SarShort.Select(item => item.X).ToList();

                    //List<double> Yshort = SarShort.Select(item => item.Y).ToList();



                    //indicator_sar_long_plot = plt.Plot.AddScatter(Xlong.ToArray(),
Ylong.ToArray(), color: Color.LightPink, lineWidth: 0, markerSize: 5);

                    //indicator_sar_long_plot.YAxisIndex = 1;

                    //indicator_sar_short_plot = plt.Plot.AddScatter(Xshort.ToArray(),
Yshort.ToArray(), color: Color.LightYellow, lineWidth: 0, markerSize: 5);

                    //indicator_sar_short_plot.YAxisIndex = 1;



                    StartAlgorithm();
```

```
                }
            }
        }

        #endregion

        #region - Algorithm -
        public long order_id = 0;
        decimal quantity;
        public bool start = false;
        public bool position;
        public bool temp_position;
        public bool start_programm = true;

        #region - Check change position (Long, Short) -
        private void Position()
        {
            try
            {
                if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close <
sma_long.ys[sma_long.ys.Length - 1]) position = false;
                else position = true;
            }
            catch (Exception c)
            {
                ErrorText.Add($"Position {c.Message}");
            }
        }
        private void TempPosition()
        {
            try
            {
                if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close <
sma_long.ys[sma_long.ys.Length - 1]) temp_position = false;
                else temp_position = true;
```

```
    }
    catch (Exception c)
    {
        ErrorText.Add($"TempPosition {c.Message}");
    }
}

#endregion

#region - Check SL TP -
private bool PriceBolingerLongTP()
{
    try
    {
        if (BOLINGER_TP > 0)
        {
            double price_bolinger = sma_long.upper[sma_long.upper.Length - 1];
            double price_sma = sma_long.ys[sma_long.ys.Length - 1];
            double price = (price_bolinger - price_sma) / 100 * BOLINGER_TP + price_sma;
            if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close > price) return true;
            else return false;
        }
        else return false;
    }
    catch (Exception c)
    {
        ErrorText.Add($"PriceBolingerLongTP {c.Message}");
        return false;
    }
}
private bool PriceBolingerLongSL()
{
    try
    {
```

```
        if (BOLINGER_SL > 0)
        {
            double price_bolinger = sma_long.lower[sma_long.lower.Length - 1];
            double price_sma = sma_long.ys[sma_long.ys.Length - 1];
            double price = price_sma - (price_sma - price_bolinger) / 100 * BOLINGER_SL;
            if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close < price) return true;
            else return false;
        }
        else return false;

    }
    catch (Exception c)
    {
        ErrorText.Add($"PriceBolingerLongSL {c.Message}");
        return false;
    }
}
private bool PriceBolingerShortTP()
{
    try
    {
        if (BOLINGER_TP > 0)
        {
            double price_bolinger = sma_long.lower[sma_long.lower.Length - 1];
            double price_sma = sma_long.ys[sma_long.ys.Length - 1];
            double price = price_sma - (price_sma - price_bolinger) / 100 * BOLINGER_TP;
            if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close < price) return true;
            else return false;
        }
        else return false;
    }
    catch (Exception c)
    {
        ErrorText.Add($"PriceBolingerShortTP {c.Message}");
```

```
        return false;
    }
}
private bool PriceBolingerShortSL()
{
    try
    {
        if (BOLINGER_SL > 0)
        {
            double price_bolinger = sma_long.upper[sma_long.upper.Length - 1];
            double price_sma = sma_long.ys[sma_long.ys.Length - 1];
            double price = (price_bolinger - price_sma) / 100 * BOLINGER_SL + price_sma;
            if (list_candle_ohlc[list_candle_ohlc.Count - 1].Close > price) return true;
            else return false;
        }
        else return false;
    }
    catch (Exception c)
    {
        ErrorText.Add($"PriceBolingerShortSL {c.Message}");
        return false;
    }
}
#endregion



private void StartAlgorithm()
{
    try
    {
        if (variables.START_BET && variables.ONLINE_CHART && order_id == 0)
        {
            Position();
```

```
                    if (start_programm)
                    {
                        TempPosition();
                        start_programm = false;
                    }

                    if (position == true && temp_position == false) start = true;
                    else if (position == false && temp_position == true) start = true;

                    TempPosition();
                }
                string symbol = LIST_SYMBOLS.Text;


                if (variables.START_BET && variables.ONLINE_CHART && order_id != 0)
                {
                    bool sl = false;
                    bool tp = false;
                    PositionSide position_side =
Algorithm.AlgorithmBet.InfoOrderPositionSide(socket, symbol, order_id);
                    if (position_side == PositionSide.Long)
                    {
                        tp = PriceBolingerLongTP();
                        sl = PriceBolingerLongSL();
                    }
                    else if (position_side == PositionSide.Short)
                    {
                        tp = PriceBolingerShortTP();
                        sl = PriceBolingerShortSL();
                    }
                    if (tp || sl)
                    {
                        order_id = Algorithm.AlgorithmBet.CloseOrder(socket, symbol, order_id,
quantity);
                        if (order_id == 0) start_programm = true;
```

```
                    }
                }
                if (variables.START_BET && variables.ONLINE_CHART && start && order_id == 0)
                {
                    if (USDT_BET > 0m && variables.PRICE_SYMBOL > 0m)
                    {
                        quantity = RoundQuantity(USDT_BET / variables.PRICE_SYMBOL);

                        order_id = Algorithm.AlgorithmBet.OpenOrder(socket, symbol, quantity,
list_candle_ohlc[list_candle_ohlc.Count - 1].Close, sma_long.ys[sma_long.ys.Length - 1]);

                        start = false;
                    }

                }
            }
            catch (Exception c)
            {
                ErrorText.Add($"StartAlgorithm {c.Message}");
            }
        }

        #endregion

        #region - Load Candles -
        private void LoadingCandlesToDB()
        {
            try
            {
                string symbol = LIST_SYMBOLS.Text;
                if (symbol != "")
                {
                    Klines(symbol, klines_count: COUNT_CANDLES);
                }
            }
```

```
    catch (Exception c)
    {
        ErrorText.Add($"LoadingCandlesToDB {c.Message}");
    }
}

#endregion

#region - List Sumbols -
private void GetSumbolName()
{
    Symbols = ListSymbols();
    foreach (var it in Symbols)
    {
        list_sumbols_name.Add(it.Name);
    }
    list_sumbols_name.Sort();
    LIST_SYMBOLS.Items.Refresh();
    LIST_SYMBOLS.SelectedIndex = 0;
}
public List<Symbol> ListSymbols()
{
    List<Symbol> list = new List<Symbol>();
    try
    {
        var result = socket.futures.ExchangeData.GetExchangeInfoAsync().Result;
        if (!result.Success) ErrorText.Add($"Failed ListSymbols {result.Error?.Message}");
        else
        {
            foreach (var it in result.Data.Symbols.ToList())
            {
                list.Add(new Symbol()
                {
                    Name = it.Name,
```

```
                    MinQuantity = it.LotSizeFilter.MinQuantity,
                    StepSize = it.LotSizeFilter.StepSize,
                    TickSize = it.PriceFilter.TickSize
                });
            }
        }
    }
    catch (Exception e)
    {
        ErrorText.Add($"ListSymbols {e.Message}");
    }
    return list;
}

#endregion

#region - Chart -
private void Chart()
{
    plt.Plot.Layout(padding: 12);
    plt.Plot.Style(figureBackground: Color.Black, dataBackground: Color.Black);
    plt.Plot.Frameless();
    plt.Plot.XAxis.TickLabelStyle(color: Color.White);
    plt.Plot.XAxis.TickMarkColor(ColorTranslator.FromHtml("#333333"));
    plt.Plot.XAxis.MajorGrid(color: ColorTranslator.FromHtml("#333333"));

    plt.Plot.YAxis.Ticks(false);
    plt.Plot.YAxis.Grid(false);
    plt.Plot.YAxis2.Ticks(true);
    plt.Plot.YAxis2.Grid(true);
    plt.Plot.YAxis2.TickLabelStyle(color: ColorTranslator.FromHtml("#00FF00"));
    plt.Plot.YAxis2.TickMarkColor(ColorTranslator.FromHtml("#333333"));
    plt.Plot.YAxis2.MajorGrid(color: ColorTranslator.FromHtml("#333333"));
```

```
            var legend = plt.Plot.Legend();
            legend.FillColor = Color.Transparent;
            legend.OutlineColor = Color.Transparent;
            legend.Font.Color = Color.White;
            legend.Font.Bold = true;
        }
        #endregion

        #region - Async klines -

        private void STOP_ASYNC_Click(object sender, RoutedEventArgs e)
        {
            StopAsync();
        }
        private void StopAsync()
        {
            try
            {
                socket.socketClient.UnsubscribeAllAsync();
            }
            catch (Exception c)
            {
                ErrorText.Add($"STOP_ASYNC_Click {c.Message}");
            }
        }
        //public Candle candle = new Candle();
        public void StartKlineAsync()
        {
            StartPriceAsync();
            socket.socketClient.UsdFuturesStreams.SubscribeToKlineUpdatesAsync(LIST_SYMBOLS.Text,
interval_time, Message =>
            {
                Dispatcher.Invoke(new Action(() =>
                {
                    Candle candle = new Candle();
```

```
                candle.DateTime = Message.Data.Data.OpenTime;
                candle.Open = Decimal.ToDouble(Message.Data.Data.OpenPrice);
                candle.High = Decimal.ToDouble(Message.Data.Data.HighPrice);
                candle.Low = Decimal.ToDouble(Message.Data.Data.LowPrice);
                candle.Close = Decimal.ToDouble(Message.Data.Data.ClosePrice);
                candle.TimeSpan = timeSpan.Ticks;
                variables.PRICE_SYMBOL = Message.Data.Data.ClosePrice;
                if(variables.IsDataBase)
                {
                    ConnectCandle.Update(candle);
                }
                else
                {
                    if (Candles[Candles.Count - 1].DateTime == candle.DateTime)
                    {
                        Candles[Candles.Count - 1] = candle;
                    }
                    else
                    {
                        Candles.Add(candle);
                    }
                }

                LoadingCandlesToChart();
                LoadingChart();
                plt.Refresh();
            }));
        });

    }

    private void StartPriceAsync()
    {
        socket.socketClient.UsdFuturesStreams.SubscribeToMarkPriceUpdatesAsync(symbol:
LIST_SYMBOLS.Text, updateInterval: 1000, Message =>
```

```
            {
                Dispatcher.Invoke(new Action(() =>
                {
                    variables.PRICE_SYMBOL = Message.Data.MarkPrice;
                }));
            });
        }
        #endregion

        #region - Candles Save -
        public void Klines(string Symbol, DateTime? start_time = null, DateTime? end_time = null,
int? klines_count = null)
        {
            try
            {
                var result = socket.futures.ExchangeData.GetKlinesAsync(symbol: Symbol, interval:
interval_time, startTime: start_time, endTime: end_time, limit: klines_count).Result;
                if (!result.Success) ErrorText.Add("Error GetKlinesAsync");
                else
                {
                    List<Candle> list = new List<Candle>();
                    foreach (var it in result.Data.ToList())
                    {
                        Candle candle = new Candle();
                        candle.DateTime = it.OpenTime;
                        candle.Open = Decimal.ToDouble(it.OpenPrice);
                        candle.High = Decimal.ToDouble(it.HighPrice);
                        candle.Low = Decimal.ToDouble(it.LowPrice);
                        candle.Close = Decimal.ToDouble(it.ClosePrice);
                        candle.TimeSpan = timeSpan.Ticks;
                        list.Add(candle);
                    }
                    if (variables.IsDataBase)
                    {
                        ConnectCandle.InsertRange(list);
```

```
                    }
                    else
                    {
                        Candles = list;
                    }
                    variables.PRICE_SYMBOL = result.Data.ToList()[result.Data.ToList().Count -
1].ClosePrice;
                }
            }
            catch (Exception e)
            {
                ErrorText.Add($"Klines {e.Message}");
            }
        }

        #endregion

        #region - Login -
        private void Button_Save(object sender, RoutedEventArgs e)
        {
            try
            {
                if (CLIENT_NAME != "" && API_KEY != "" && SECRET_KEY != "")
                {
                    string path = System.IO.Path.Combine(Environment.CurrentDirectory, "clients");
                    if (!Directory.Exists(path)) Directory.CreateDirectory(path);
                    if (!File.Exists(path + "/" + CLIENT_NAME))
                    {

                        Client client = new Client(CLIENT_NAME, API_KEY, SECRET_KEY);
                        string json = JsonConvert.SerializeObject(client);
                        File.WriteAllText(path + "/" + CLIENT_NAME, json);
                        Clients();
                        CLIENT_NAME = "";
                        API_KEY = "";
```

```
                    SECRET_KEY = "";
                }
            }
        }
        catch (Exception c)
        {
            ErrorText.Add($"Button_Save {c.Message}");
        }
    }
    private void Clients()
    {
        try
        {
            string path = System.IO.Path.Combine(Environment.CurrentDirectory, "clients");
            if (!Directory.Exists(path)) Directory.CreateDirectory(path);
            List<string> filesDir = (from a in Directory.GetFiles(path) select
System.IO.Path.GetFileNameWithoutExtension(a)).ToList();
            if (filesDir.Count > 0)
            {
                ClientList file_list = new ClientList(filesDir);
                BOX_NAME.ItemsSource = file_list.BoxNameContent;
                BOX_NAME.SelectedItem = file_list.BoxNameContent[0];
            }
        }
        catch (Exception e)
        {
            ErrorText.Add($"Clients {e.Message}");
        }
    }
    private void Button_Login(object sender, RoutedEventArgs e)
    {
        try
        {
            if (API_KEY != "" && SECRET_KEY != "")
            {
```

```
            socket = new Socket(API_KEY, SECRET_KEY);
            Login_Click();
            CLIENT_NAME = "";
            API_KEY = "";
            SECRET_KEY = "";
        }
        else if (BOX_NAME.Text != "")
        {
            string path = System.IO.Path.Combine(Environment.CurrentDirectory, "clients");
            string json = File.ReadAllText(path + "\\" + BOX_NAME.Text);
            Client client = JsonConvert.DeserializeObject<Client>(json);
            socket = new Socket(client.ApiKey, client.SecretKey);
            Login_Click();
        }
    }
    catch (Exception c)
    {
        ErrorText.Add($"Button_Login {c.Message}");
    }
}
private void Login_Click()
{
    LOGIN_GRID.Visibility = Visibility.Hidden;
    EXIT_GRID.Visibility = Visibility.Visible;
    GetSumbolName();
}
private void Exit_Click(object sender, RoutedEventArgs e)
{
    EXIT_GRID.Visibility = Visibility.Hidden;
    LOGIN_GRID.Visibility = Visibility.Visible;
    socket = null;
    list_sumbols_name.Clear();
}
#endregion
```

```
        #region - Error -

        // ------------------------------------------------------ Start Error Text Block ----------
----------------------------

        private void ErrorWatcher()

        {

            try

            {

                FileSystemWatcher error_watcher = new FileSystemWatcher();

                error_watcher.Path = ErrorText.Directory();

                error_watcher.NotifyFilter = NotifyFilters.LastWrite | NotifyFilters.LastAccess |
NotifyFilters.FileName | NotifyFilters.DirectoryName;

                error_watcher.Changed += new FileSystemEventHandler(OnChanged);

                error_watcher.Filter = ErrorText.Patch();

                error_watcher.EnableRaisingEvents = true;

            }

            catch (Exception e)

            {

                ErrorText.Add($"ErrorWatcher {e.Message}");

            }

        }

        private void OnChanged(object source, FileSystemEventArgs e)

        {

            Dispatcher.Invoke(new Action(() => { ERROR_LOG.Text =
File.ReadAllText(ErrorText.FullPatch()); }));

        }

        private void Button_ClearErrors(object sender, RoutedEventArgs e)

        {

            File.WriteAllText(ErrorText.FullPatch(), "");

        }

        // ------------------------------------------------------ End Error Text Block ------------
----------------------------

        #endregion



    }

}
```

Додаток Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Бердник Михайла Генадійовича, д.т.н., доцент, професор каф. ПЗКС
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

На кваліфікаційну роботу

студента *Марченко Нікіти Вячеславовича*
(прізвище, ім'я, по батькові)

курсу II групи *122М-21- 1*
спеціальності *122 Комп'ютерні науки*
на тему *Розробка та дослідження ефективності впровадження криптовалютного бота для біржі Binance*

Актуальність теми Актуальність кваліфікаційної магістерської роботи визначена стрімким розвитком криптовалют та технологій, які вони використовують.

Мета досліджень *Метою кваліфікаційної роботи є створення програми бота для атоматичних торгів на біржі Binance з використанням алгоритму на основі причинно-наслідкових зв'язків із ключовими індикаторами*

Коротка характеристика розділів роботи *У першому розділі проаналізовано предметну область завдання, визначено актуальність направлення, сформульовано постановку проблеми, зазначено вимоги до реалізації боту. У другому розділі розглянуто існуючі рішення, обрано платформу розробки, описано розробку застосунка, названо основні алгоритми, індекатори та структуру, наведено довідкову інформацію та функціональні можливості. У третьому розділі продеманстованно роботу ПЗ, усі реалізовані*

налаштування та можливості.

Практичне значення роботи *Результатом роботи виступає бот який на основі причинно-наслідкових зв'язків із ключовими індикаторами відкриває ордера на купівлю або продаж криптовалюти у потрібний* момент.

Зауваження та недоліки

Висновки та оцінка Кваліфікаційна робота заслуговує оцінки «відмінно», а виконавець заслуговує на присвоєння відповідної кваліфікації.

Науковий керівник Бердник Михайло Генадійович, професор, каф. ПЗКС

(прізвище, ім'я, по батькові, посада, місце роботи)

«_18__»грудня_____2022 р. _____

(підпис)

Додаток В

РЕЦЕНЗІЯ
на кваліфікаційну роботу

студента *Марченко Нікіти Вячеславовича*
(прізвище, ім'я, по батькові)
курсу ІІ групи 122м-21-1
кафедри програмного забезпечення комп'ютерних систем
спеціальності *122 Комп'ютерні науки*
Тема роботи *Розробка та дослідження ефективності впровадження криптовалютного бота для біржі Binance*

Стисла характеристика розділів роботи У першому розділі визначено актуальність, сформульовано постановку задачі, зазначено вимоги до реалізації боту, технології та програмні засоби. У другому розділі розглянуто існуючі рішення, обрано платформу розробки, описано розробку застосунка. У третьому розділі продеманстровано роботу ПЗ, усі реалізовані налаштування та можливості. Також представлені результати торгівлі.

Пропозиції, внесені студентом, рівень їх наукового обґрунтування обґрунтовані і випливають із суті зробленої роботи.

Практичне значення роботи полягає у тому, що автоматична торгівля потребує набагато менше уваги з боку людини.

Якість оформлення роботи робота виконана у відповідності до вимог оформлення кваліфікаційних робіт і повністю відповідає поставленій задачі.
Недоліки в роботі
тематики роботи.
Загальний висновок отримані результати є закінченою науково-дослідною роботою і викликають науковий інтерес і демонструють здатність *Марченко Нікіти Вячеславовича* до самостійного аналізу і проведення наукової роботи та вміння розробки комп'ютерних програм. Кваліфікаційна робота заслуговує оцінки "відмінно", а Марченко Н. В. – присвоєння відповідної кваліфікації.
(підготовленість студента до самостійної роботи як спеціаліста)

Оцінка магістерської роботи "відмінно"

Рецензент Шедловський І.А., к.т.н., доцент, доцент каф. ІТКІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«_18__»_грудня_____2022р. _____
(підпис)

ДОДАТОК Г

СПИСОК ФАЙЛІВ НА ДИСКУ

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Марченко.doc	Пояснювальна записка роботи. Документ Word.
Диплом_Марченко.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Марченко.ppt	Презентація роботи