Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОЇ ТОРГІВЛІ НА РИНКУ КРИПТО-АКТИВІВ З ВИКОРИСТАННЯМ PYTHON ТА MYSQL

DEVELOPMENT OF SOFTWARE FOR AUTOMATED TRADING ON THE CRYPTOCURRENCY MARKET USING PYTHON AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Жалко Микола Володимирович

Керівник:
Бойко Лев Степанович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Жалку Миколі Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка програмного забезпечення для автоматизованої торгівлі на ринку крипто-активів з використанням Python та MySQL»*
Керівник роботи: *к.т.н., доцент Бойко Л. С.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01- 02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи*: MySQL, бібліотеки ccxt, Python, Telegram, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *формулювання вимог до нової системи, проєктування архітектури та функціональних можливостей програмного забезпечення, розробка Telegram-бота, розробка бази даних у MySQL.*
5. Перелік графічного матеріалу: *8 рисунків, з них – 4 діаграми.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Бойко Л. С.*		
Специфікація вимог до розробленої системи	*Бойко Л. С.*		
Розробка об'єкта проектування	*Бойко Л. С.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,58 %		
Академічна доброчесність	*Бойко Л. С.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	вик.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	вик.
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	вик.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	вик.
5	Практична реалізація об'єкта проектування та розробка базц даних	до 26.04.2025 р.	вик.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	вик.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	вик.

Здобувач вищої освіти _____ Микола ЖАЛКО

Керівник кваліфікаційної роботи _____ Лев БОЙКО

АНОТАЦІЯ

Жалко М. В. Розробка програмного забезпечення для автоматизованої торгівлі на ринку крипто-активів з використанням Python та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі здійснено аналіз крипто-валютного ринку та існуючих програмних засобів для автоматизованої торгівлі. У другому розділі визначено вимоги до системи та обґрунтовано вибір засобів та методів для розробки ПЗ. У третьому розділі розроблено базу даних та Telegram-бот для здійснення автоматизованої торгівлі, проведено тестування та підготовлено супровідну документацію. У висновках підбито підсумки виконаної роботи та окреслено напрямки подальшого розвитку.

Ключові слова: крипто-активи, cctx, MySQL, Python, API, Binance, автоматизація, торгівля.

ABSTRACT

Zalko M. Development of Software for Automated Trading on the Cryptocurrency Market Using Python and MySQL. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, and a list of references.

The first chapter provides an analysis of the cryptocurrency market and existing software tools for automated trading. The second chapter defines the system requirements and justifies the choice of tools and methods for software development. The third chapter presents the development of a database and a Telegram bot for executing automated trading, as well as testing and preparation of supporting documentation. The conclusion summarizes the work completed and outlines directions for further development.

Keywords: crypto-assets, cctx, MySQL, Python, API, Binance, automation, trading.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі відбувається стрімке зростання криптовалютного ринку, збільшення кількості користувачів цифрових активів і потреба в зручних, безпечних та ефективних засобах для автоматизованої торгівлі. Сучасні користувачі очікують від програмного забезпечення не лише високої функціональності й гнучкості, а й повноцінної інтеграції з мобільними платформами та надійного захисту персональних даних і API-ключів.

Метою роботи є створення програмного забезпечення для автоматизованої торгівлі криптовалютними активами з використанням мови програмування Python, бібліотеки ccxt для взаємодії з біржею Binance, Telegram Bot API для реалізації користувацького інтерфейсу та системи керування базами даних MySQL для зберігання інформації та забезпечення стабільної роботи системи.

Об’єктом роботи є процеси автоматизованої торгівлі на ринку криптовалютних активів.

Предметом кваліфікаційної роботи бакалавра є методи, алгоритми та інструменти програмної реалізації автоматизованої торгової системи, що використовує API криптовалютної біржі, засоби мови Python та базу даних MySQL.

Для реалізації поставленої мети були висунуті наступні завдання:

– провести аналіз поточного стану криптовалютного ринку та оцінити потенціал використання автоматизованих торгових систем;

– дослідити існуючі програмні рішення для реалізації торгових ботів та обґрунтувати потребу в розробці нового програмного продукту;

– спроєктувати архітектуру програмного забезпечення, що включає Telegram-інтерфейс, API-взаємодію з біржею Binance та базу даних;

– реалізувати функціонал для забезпечення взаємодії користувача з ботом, реєстрації API-ключів, моніторингу ринку та розміщення торгових ордерів;

– розробити структуру та реалізувати базу даних для зберігання інформації про транзакції, налаштування користувачів та історію взаємодій із системою;

– забезпечити захист конфіденційної інформації шляхом впровадження механізмів хешування та шифрування API-ключів;

– провести тестування програмного забезпечення в умовах, максимально наближених до реальних торгових сценаріїв;

– розробити рекомендації щодо встановлення, налаштування та експлуатації програмного продукту.

РОЗДІЛ 1
АНАЛІЗ СТАНУ АВТОМАТИЗОВАНОЇ ТОРГІВЛІ КРИПТО-АКТИВАМИ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Протягом останніх десяти років криптовалютний ринок зазнав суттєвої еволюції: із початкового експериментального інструменту він перетворився на важливий сегмент глобального інвестиційного простору [1]. Децентралізація, висока швидкість транзакцій, прозорість блокчейн-технологій та значна волатильність сприяли широкому поширенню цифрових активів як серед інституційних, так і роздрібних інвесторів. Паралельно з цим зросла потреба в потужних торгових інструментах, здатних автоматизувати аналіз ринку, прийняття рішень і виконання операцій у реальному часі [2, 3]. Хоча ручна торгівля й досі використовується початківцями, вона втрачає конкурентоспроможність у середовищі високоінтенсивних ринкових умов та швидких цінових коливань, поступаючись алгоритмічним і автоматизованим системам [4].

Проблема розробки систем автоматизованої торгівлі полягає в необхідності створення програмного забезпечення, яке здатне швидко реагувати на динаміку ринку, обробляти великі обсяги даних, ефективно взаємодіяти з біржовими API, забезпечувати надійний захист особистої інформації та бути зручним для користувача [5]. Наразі існує чимало рішень у цій сфері – від комерційних платформ, таких як 3Commas, Cryptohopper, HaasOnline, до безкоштовних або умовно безкоштовних open-source продуктів, зокрема Zenbot, Gekko чи скриптів на основі бібліотек ccxt та python-binance.

Однак, попри різноманіття доступних інструментів, багато з них мають суттєві обмеження. Зокрема, для їхнього використання часто потрібні глибокі технічні знання щодо налаштування середовища розробки, створення торгових стратегій і стабільної роботи з API. Крім того, значна частина таких рішень не

має інтуїтивно зрозумілого інтерфейсу, що ускладнює доступ до автоматизованої торгівлі для менш досвідчених користувачів.

Особливу увагу слід приділити питанням безпеки в контексті автоматизованої торгівлі. Зберігання API-ключів у відкритому вигляді, відсутність багаторівневої автентифікації та слабкий контроль за виконанням торгових операцій можуть спричинити серйозні фінансові втрати або призвести до компрометації облікових записів користувачів [6]. Актуальність цієї проблеми зростає на тлі зростання кількості кіберінцидентів, спрямованих на криптовалютні біржі та автоматизовані торгові платформи. У зв’язку з цим питання створення безпечного програмного забезпечення виходить на перший план і стає одним із головних завдань при проєктуванні та реалізації подібних систем.

Окрім технічних аспектів, варто підкреслити зростаючу популярність використання Telegram як платформи для інтеграції торгових систем. Завдяки широкому розповсюдженню, простоті у використанні та наявності зручного API, Telegram став ефективним інструментом для розробки чат-ботів, які забезпечують оперативну взаємодію з користувачем. Через такі боти можна надсилати сповіщення про зміну ринкових цін, здійснювати запити на торгівельні дії або підтвердження ордерів. Цей підхід дозволяє уникнути створення окремих вебінтерфейсів чи десктопних застосунків, акцентуючи увагу на мобільності, швидкому доступі та універсальності використання системи з будь-якої точки світу [7].

Ще одним важливим елементом автоматизованої торгової системи є база даних, яка забезпечує надійне зберігання ключової інформації – історії торгів, користувацьких налаштувань, API-ключів, логів роботи системи та результатів виконаних операцій. Найчастіше в таких рішеннях використовуються реляційні системи керування БД, зокрема MySQL, що відзначаються стабільністю, чіткою структурою таблиць і зручною мовою запитів SQL [8, 9]. Інтеграція бази даних із торговим скриптом відкриває можливість реалізації складних торгових

стратегій, обробки історичних даних для аналітики та створення гнучких звітів для подальшого аналізу ефективності системи.

Патентний та аналітичний аналіз показує, що, незважаючи на велику кількість вже реалізованих рішень, сегмент персоналізованих, захищених, адаптивних і водночас доступних систем автоматизованої торгівлі залишається недостатньо заповненим. Створення власного Telegram-бота для автоматизованої торгівлі з використанням бібліотеки ccxt на мові Python у поєднанні з базою даних MySQL дає змогу ефективно реалізовувати торговельні стратегії, забезпечуючи при цьому зручний інтерфейс та надійний захист персональних даних користувача. Такий підхід не лише відповідає сучасним вимогам ринку, а й закладає фундамент для подальшого вдосконалення системи в напрямах масштабування, аналітики та впровадження технологій машинного навчання, що відкриває перспективи для нових досліджень і практичного застосування.

Для вибору оптимальної платформи проведемо аналіз провідних криптовалютних бірж. За обсягом торгів на криптовалютних біржах перевага надається Binance (рис. 1.1).

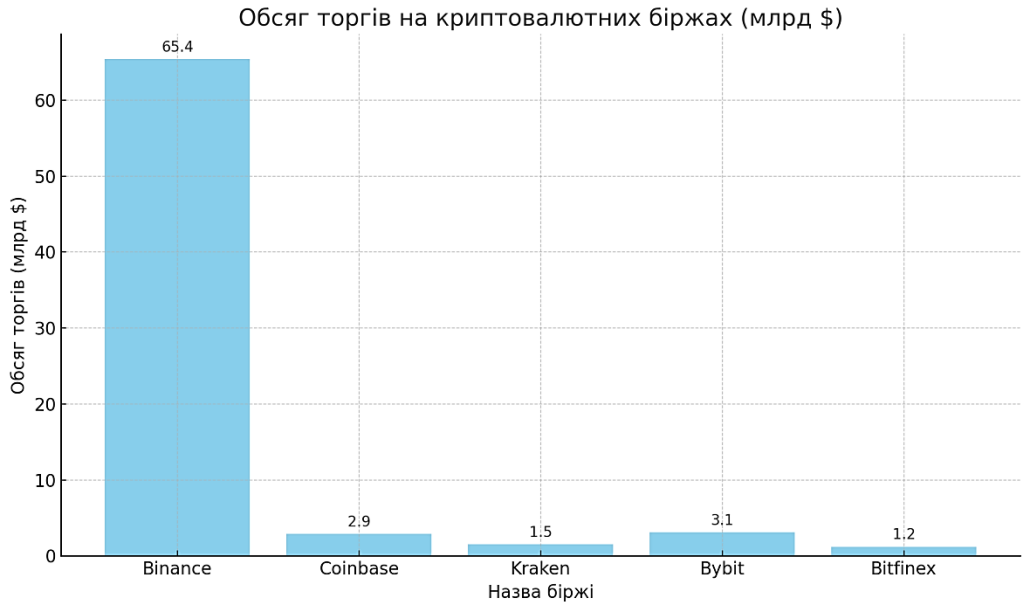

Рисунок 1.1 – Аналіз результатів бірж за обсягом торгів на криптовалютних біржах

З даної діаграми, що відображає порівняльну характеристику криптовалютних бірж за обсягом добових торгів (млрд $), бачимо, що Binance суттєво випереджає інші біржі за обсягом торгів. Bybit, Coinbase та Kraken мають приблизно схожі показники, але значно менші. Bitfinex має найменший добовий обсяг.

Розглянемо порівняльну характеристику криптобірж за наявністю тестової мережі та підтримки бібліотеки ccxt (рис. 1.2).

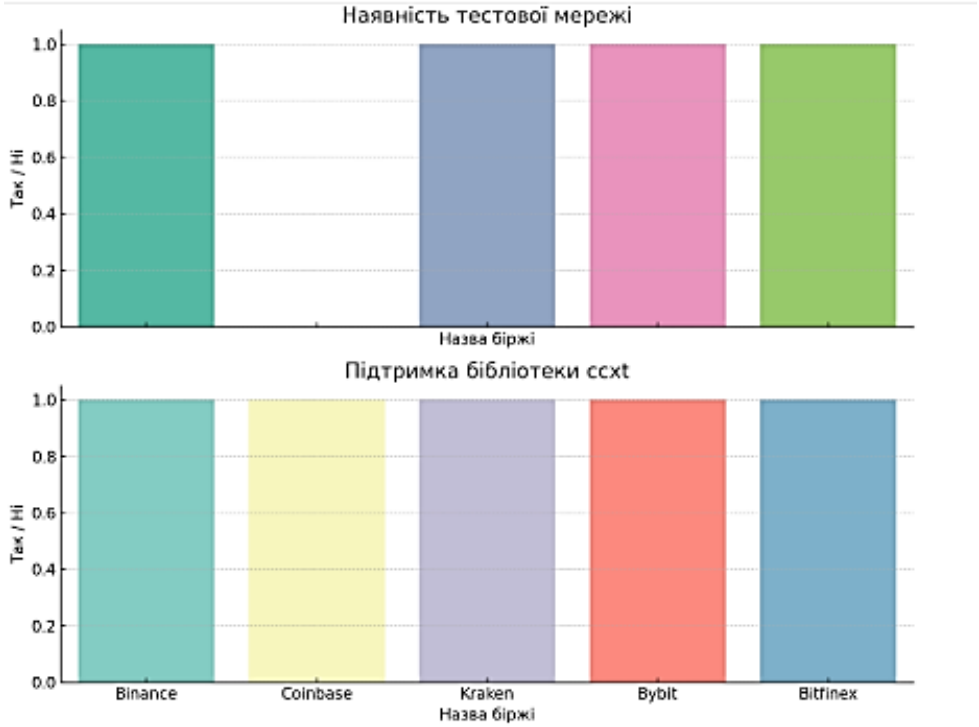

Рисунок 1.2 – Аналіз результатів бірж за наявністю тестової мережі та підтримки бібліотеки ccxt

У всіх бірж, крім Coinbase, є тестова мережа, що корисно для безпечного тестування алгоритмів автоматизованої торгівлі.

Усі біржі підтримують бібліотеку ccxt, що спрощує інтеграцію для автоматизації торгівлі через API.

У результаті ретельного аналізу провідних криптовалютних бірж можна зробити висновок, що Binance є однією з найоптимальніших платформ для реалізації автоматизованої торгівлі. Це пояснюється її значним добовим

торговим обігом, наявністю REST та WebSocket API, можливістю використання тестового середовища, а також широкою інтеграцією з бібліотекою ccxt, що значно полегшує створення програм на Python. Додатковою перевагою для розробників з усього світу є доступність документації кількома мовами.

З огляду на зазначені характеристики, для реалізації цілей кваліфікаційної роботи бакалавра було обрано саме Binance – біржу, яка надає весь необхідний інструментарій для створення надійного, масштабованого торгового бота та забезпечує відкритий доступ до API для ефективної інтеграції.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Через стрімкий розвиток цифрової економіки та зростання кількості фінансових операцій в онлайн-середовищі зростає актуальність удосконалення інструментів автоматизації торгівлі цифровими активами, зокрема криптовалютами. Крипторинок вирізняється високою волатильністю, постійною активністю 24/7 та швидкою зміною цінових тенденцій, що потребує використання програмних рішень, здатних миттєво реагувати на ринкові зміни та приймати обґрунтовані торгові рішення. У порівнянні з ручною торгівлею, яка вимагає безперервної уваги з боку трейдера, автоматизовані системи демонструють вищу ефективність: вони оперативно обробляють ринкові сигнали, знижують вплив емоцій на прийняття рішень та мінімізують людський фактор.

Формулювання завдання для кваліфікаційної роботи охоплює розв’язання комплексної проблеми, пов’язаної з розробкою повнофункціонального програмного забезпечення для автоматизованої торгівлі на ринку криптовалют. Реалізація передбачає використання бібліотек Python для інтеграції з API біржі Binance, а також організацію збереження торгових та аналітичних даних за допомогою системи керування БД MySQL.

Для реалізації поставленої мети слід виконати наступні завдання:

– провести аналіз поточного стану криптовалютного ринку та оцінити потенціал використання автоматизованих торгових систем;

– дослідити існуючі програмні рішення для реалізації торгових ботів та обґрунтувати потребу в розробці нового програмного продукту;

– спроєктувати архітектуру програмного забезпечення, що включає Telegram-інтерфейс, API-взаємодію з біржею Binance та базу даних;

– реалізувати функціонал для забезпечення взаємодії користувача з ботом, реєстрації API-ключів, моніторингу ринку та розміщення торгових ордерів;

– розробити структуру та реалізувати базу даних для зберігання інформації про транзакції, налаштування користувачів та історію взаємодій із системою;

– забезпечити захист конфіденційної інформації шляхом впровадження механізмів хешування та шифрування API-ключів;

– провести тестування програмного забезпечення в умовах, максимально наближених до реальних торгових сценаріїв;

– розробити рекомендації щодо встановлення, налаштування та експлуатації програмного продукту.

Висновки до розділу 1

У першому розділі роботи було проведено всебічний теоретичний аналіз предметної галузі, що охоплює як концептуальні засади, так і технічні особливості функціонування систем автоматизованої торгівлі криптовалютами. Значну увагу приділено аналізу поточного стану криптовалютного ринку, його характерних рис, потенційних ризиків та переваг, а також огляду технологічних рішень, які застосовуються при створенні торгових платформ. Виокремлено основні напрями розвитку алгоритмічної торгівлі, зокрема активне використання API, яке пропонують провідні біржі, зокрема Binance, що

відкриває нові можливості для створення автономних торгових систем за допомогою Python.

Окремим аспектом розглянуто наявні інструменти, які можуть бути використані розробником для досягнення цілей проєкту. Зокрема, проаналізовано переваги застосування бібліотеки ccxt, яка надає уніфікований доступ до багатьох криптовалютних бірж, включно з Binance, та значно полегшує процес створення програмного забезпечення для торгівлі цифровими активами. Також обґрунтовано вибір СКБД MySQL для зберігання торгової інформації, логів і конфігурацій, з огляду на її стабільність, високу продуктивність та зручну інтеграцію з Python-застосунками.

У межах аналізу також було проведено порівняння кількох криптобірж з точки зору їх відповідності вимогам до створення автоматизованої торгової системи. На основі цього порівняння визначено, що біржа Binance є найоптимальнішим вибором для реалізації продукту завдяки високій ліквідності, наявності тестового середовища, відкритій API-документації та повній сумісності з бібліотекою ccxt, що спрощує технічну інтеграцію.

Було сформульовано технічне завдання, яке передбачає розробку торгового програмного забезпечення з підтримкою Telegram-інтерфейсу. Такий підхід дозволяє гнучко налаштувати параметри торгівлі, а також зручність управління для кінцевих користувачів з різних пристроїв. Постановка завдання деталізована до рівня функціональних модулів, серед яких: ядро торгової логіки, модуль взаємодії з API, система зберігання та обробки даних, засоби комунікації, логування, а також елементи безпеки.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Етап аналізу та визначення вимог до програмного продукту відіграє основну роль у процесі створення ефективної, стабільної та повнофункціональної системи автоматизованої торгівлі на криптовалютному ринку. Саме на цьому етапі закладаються основи майбутньої архітектури програмного продукту, визначаються його функціональні можливості, нефункціональні властивості, а також принципи взаємодії з зовнішніми компонентами – такими як API криптобірж, системи керування базами даних та інтерфейси для користувачів. Зважаючи на швидкоплинність крипторинку, високу волатильність активів і вплив численних зовнішніх факторів, коректне формулювання вимог є вирішальним для забезпечення надійної роботи, високої продуктивності та адаптивності розроблюваної системи.

На основі теоретичного аналізу предметної області, криптовалютного ринку та відповідних інструментів автоматизації, проведеного в першому розділі, було визначено перелік функціональних вимог до майбутньої системи. Розроблюване програмне забезпечення повинно забезпечувати можливість обробки ринкових даних у реальному часі шляхом інтеграції з API криптовалютної біржі Binance. Для реалізації цієї взаємодії планується використання бібліотеки ccxt на мові програмування Python, яка дозволяє абстрагувати специфіку API та надає уніфікований інтерфейс до ключових біржових функцій: отримання ринкових котирувань, виставлення та скасування ордерів, перевірка балансу, а також доступ до історичних торгових даних.

Передбачається, що ядро торгової логіки буде побудоване на основі умовних конструкцій, які дозволятимуть системі реагувати на певні ринкові події. Зокрема, це можуть бути операції купівлі або продажу активів у випадку

досягнення встановлених порогових значень, або ж у відповідь на сигнали, сформовані технічними індикаторами.

Для взаємодії користувача з системою буде реалізовано інтерфейс у вигляді Telegram-бота. Через нього користувач зможе контролювати торговий процес: отримувати актуальну інформацію про ринкову ситуацію, змінювати параметри алгоритму, запускати або призупиняти торгівлю, а також переглядати аналітику здійснених операцій. Telegram-бот має бути інтегрований із основним модулем системи, забезпечуючи обмін командами та даними у внутрішньому форматі. Особлива увага приділяється безпечній авторизації користувача, що необхідно для запобігання несанкціонованому доступу до функціоналу системи.

Однією з ключових вимог до програмного забезпечення є реалізація системи збереження історичних даних. Для цього передбачено розробку бази даних на основі MySQL, яка забезпечуватиме збереження інформації про всі торгові операції, API-запити та відповіді, дії користувача, системні повідомлення, повідомлення Telegram-бота, а також логування помилок і подій [10]. Такий підхід гарантує прозорість роботи системи, створює можливості для аналізу результативності застосованих торгових стратегій і забезпечує передумови для реалізації механізмів резервного копіювання і відновлення даних у разі збоїв [11].

Розробка програмного забезпечення ґрунтується на попередньо визначених вимогах і включає створення структурної моделі взаємодії між основними складовими системи. Розглянемо UML-діаграму класів опису архітектури (рис. 2.1). Ключову роль виконує модуль торгової логіки, який у реальному часі приймає рішення, спираючись на біржові дані та параметри, встановлені користувачем. Цей компонент взаємодіє з API-підсистемою, що забезпечує обробку запитів до платформи Binance, з Telegram-модулем, який обробляє команди користувача, а також з модулем збереження інформації, відповідальним за логування та організацію даних у базі.

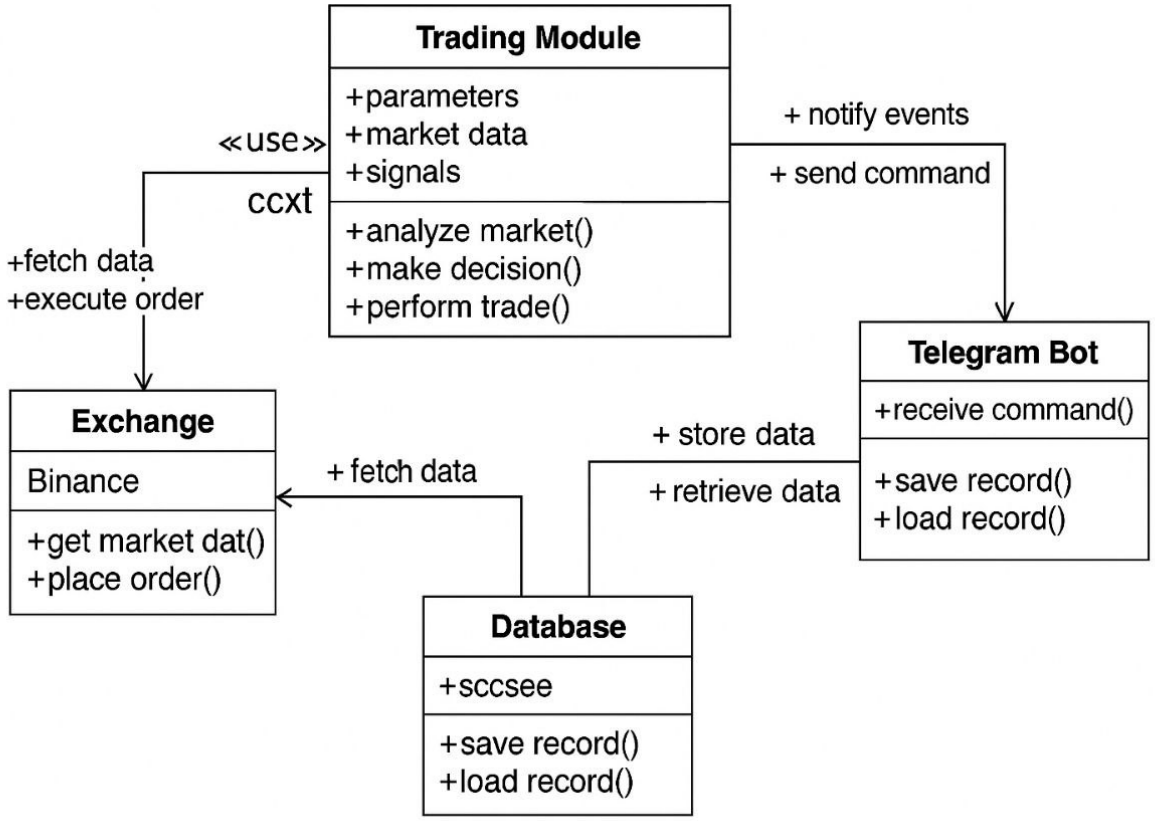

Рисунок 2.1 – UML-діаграма класів

UML-діаграма класів показує головні компоненти системи автоматизованої торгівлі криптовалютою, їхні атрибути, методи та зв’язки між ними.

Система побудована за модульним принципом, що дозволяє з легкістю змінювати окремі її елементи та забезпечує перспективу масштабування. Наприклад, за потреби можна інтегрувати нову біржу, реалізувати альтернативні торгові стратегії, додати інші канали сповіщень (як-от електронна пошта або SMS), змінити тип бази даних чи розширити аналітичну складову шляхом впровадження моделей машинного навчання для прогнозування змін на ринку.

На основі проведеного аналізу та етапу проєктування було сформовано чітке уявлення про програмний продукт, що підлягає реалізації в межах кваліфікаційної роботи бакалавра. Запланована система вирізнятиметься високим рівнем автоматизації, функціональністю, безпекою та орієнтованістю на зручність користувача. Вона забезпечуватиме ефективне виконання торгових

операцій на ринку криптовалют без необхідності втручання людини в рутинні завдання.

Виходячи з визначених вимог та розробленої структури програмного забезпечення, наступним етапом є деталізація взаємодії між окремими компонентами системи. Це необхідно для забезпечення злагодженої та ефективної роботи всіх функціональних модулів у режимі реального часу. Основним елементом архітектури виступає торговий модуль, який аналізує вхідні дані, отримані з біржі Binance за допомогою бібліотеки ccxt, і ухвалює рішення відповідно до параметрів, встановлених користувачем. Всі операції – від розміщення ордерів і перевірки балансу до отримання історії угод та скасування неактуальних заявок – здійснюються через відповідні API-запити.

Усі події, що відбуваються в системі, надсилаються до Telegram-бота, який забезпечує канал зворотного зв’язку з користувачем. Через нього надходять сповіщення про відкриття й закриття позицій, виникнення помилок під час виконання торгових операцій або успішне завершення певних дій. Крім того, бот надає можливість користувачеві змінювати параметри торгівлі, активувати або вимикати режим автоматичної торгівлі, а також виконувати ручні операції – наприклад, створювати лімітні ордери.

З технічного погляду ключовим елементом, що впливає на стабільність і безпеку роботи системи, є правильно організоване збереження даних. У рамках виконання кваліфікаційної роботи бакалавра цю функцію виконує база даних MySQL, яка забезпечує накопичення та структуроване зберігання історичної інформації. Зокрема, фіксуються час і умови виконання кожної торгової операції, значення технічних індикаторів на момент ухвалення рішень, результати роботи алгоритмів, а також зміст усіх повідомлень, надісланих через Telegram-бота. Окрім цього, база даних зберігає користувацькі налаштування, що дозволяє створювати персоналізовані торгові профілі й швидко відновлювати їх під час повторного запуску системи.

Під час проєктування архітектури системи окрему увагу приділено виявленню потенційних ризиків та впровадженню механізмів захисту від збоїв.

Зокрема, реалізовано систему повторного надсилання запитів у разі втрати з’єднання з сервером біржі, що забезпечує стійкість до мережевих несправностей. Критично важливі дані дублюються в окремих таблицях бази даних, що дає змогу автоматично відновити роботу системи у разі аварійного завершення. Для захисту облікових даних користувача API-ключі зберігаються у зашифрованому вигляді з використанням сучасних криптографічних алгоритмів і розміщуються в конфігураційному файлі з обмеженим доступом.

У процесі проєктування особлива увага приділяється забезпеченню гнучкості системи, що є необхідною умовою для подальшого масштабування. Це передбачає модульну побудову програмного коду з чітким розмежуванням функціональних обов’язків між окремими компонентами. Такий підхід дає змогу у майбутньому без суттєвого втручання в існуючу архітектуру додавати нові торгові стратегії, розширювати список підтримуваних торгових пар або інтегрувати альтернативні біржові платформи відповідно до нових вимог. Система повинна залишатися стійкою, масштабованою та здатною адаптуватися до змін ринкового середовища й потреб користувача.

Підсумовуючи, можна зазначити, що на етапі аналізу та проєктування було сформовано цілісне бачення майбутнього програмного продукту, яке охоплює як логічну архітектуру, так і схему взаємодії між ключовими модулями. Запропонована модель дозволяє не лише реалізувати цілі кваліфікаційної роботи бакалавра, але й створити життєздатне технологічне рішення, придатне до практичного застосування трейдерами та інвесторами на ринку криптовалют.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

У межах кваліфікаційної роботи бакалавра було обґрунтовано вибір певних інструментів та методів, з урахуванням їхньої актуальності, доступності, простоти інтеграції, а також відповідності вимогам до надійності та продуктивності майбутнього програмного забезпечення.

Основною мовою програмування обрано Python, яка вже зарекомендувала себе як універсальний засіб для створення рішень у галузях фінансових технологій, машинного навчання, аналізу даних і веб-розробки. Її перевагами є лаконічний синтаксис, розвинена екосистема бібліотек, активна спільнота та висока швидкість розробки.

Важливою перевагою Python у контексті реалізації торгового алгоритму є доступність бібліотеки ccxt, що забезпечує зручну взаємодію з великою кількістю криптовалютних бірж, зокрема Binance, через уніфікований API-інтерфейс [12]. Цей інструмент спрощує обробку запитів, дозволяє використовувати однаковий підхід для різних платформ, а також підтримує як синхронний, так і асинхронний режим, що забезпечує високу продуктивність і гнучкість програмного коду.

Для зберігання структурованої інформації, включаючи торгові операції, параметри угод, журнали подій та користувацькі налаштування, застосовується система керування базами даних MySQL [13]. Вибір саме цієї СКБД обумовлений її широким розповсюдженням, високою продуктивністю при обробці великих обсягів даних, підтримкою транзакцій і гнучкою мовою запитів SQL. MySQL легко інтегрується з Python-застосунками за допомогою таких бібліотек, як mysql-connector-python або SQLAlchemy, що спрощує розробку та підтримку системи. Організація даних у вигляді взаємопов’язаних таблиць забезпечує можливість виконання глибокого аналізу торгової історії, побудови аналітичних моделей, візуалізації прибутковості та оцінки ефективності застосованих торгових стратегій.

Інтерфейс взаємодії користувача з системою реалізовано у вигляді Telegram-бота, створеного з використанням бібліотеки python-telegram-bot. Цей інструмент забезпечує зручну обробку повідомлень, команд, інтерактивних кнопок, а також підтримує асинхронну взаємодію з зовнішніми API, що критично для своєчасної реакції системи. Telegram-бот виконує функцію не лише каналу комунікації, але й ключового елементу моніторингу та керування: через нього користувач у режимі реального часу отримує сповіщення про

відкриття та закриття позицій, переглядає поточний баланс, відстежує динаміку ринку, змінює параметри торгової стратегії та має змогу оперативно реагувати на нештатні події. Завдяки платформі Telegram, цей інтерфейс доступний на будь-якому пристрої без необхідності встановлення додаткового програмного забезпечення.

Алгоритмічне ядро торгової системи в рамках дипломного проєкту побудоване на основі простої порогової стратегії (threshold-based approach). Основна логіка полягає у здійсненні купівлі активу при досягненні ціною заданого рівня підтримки, а продажу – при наближенні до рівня опору. Такий механізм реалізує класичний принцип «купуй дешево – продавай дорого», що є фундаментом багатьох базових торгових стратегій.

Алгоритм у визначені інтервали часу обробляє ринкові цінові дані, зберігає останні значення у внутрішній пам’яті та виконує порівняння з заздалегідь заданими порогами. На основі цього аналізу приймається рішення про відкриття або закриття торгової позиції. Додатково реалізовано функціональність для обмеження збитків (stop-loss) і фіксації прибутку (take-profit), що дозволяє автоматизовано управляти ризиками під час роботи в умовах високої ринкової волатильності.

Проєкт розроблено з використанням модульної архітектури, що забезпечує ізольоване тестування основних компонентів системи: модуля взаємодії з API біржі, обробника повідомлень Telegram, системи зберігання даних, логіки прийняття рішень та підсистеми логування. Такий підхід спрощує підтримку та розвиток програмного забезпечення, полегшує інтеграцію нових функцій і зменшує ризик реалізації помилок під час внесення змін.

Для підвищення стабільності реалізовано обробку виключних ситуацій, перевірку з’єднання з торговою платформою, механізми повторного надсилання запитів у разі збоїв та повне логування дій системи. Усі ці елементи підвищують надійність і дозволяють оперативно виявляти та усувати потенційні проблеми.

Під час вибору інструментів особливу увагу приділяли не лише їх функціональним можливостям, а й сумісності між компонентами. Узгодженість між клієнтським інтерфейсом, торговим ядром, механізмом доступу до біржових API та базою даних забезпечує безперебійну роботу системи в режимі реального часу, враховуючи обмеження на частоту запитів і вимоги до безпеки персональної інформації.

Застосовані технології – Python [14], MySQL, Telegram API та бібліотека ccxt – добре інтегруються між собою, формуючи надійну та масштабовану платформу для реалізації автоматизованої торгової системи (рис. 2.2).

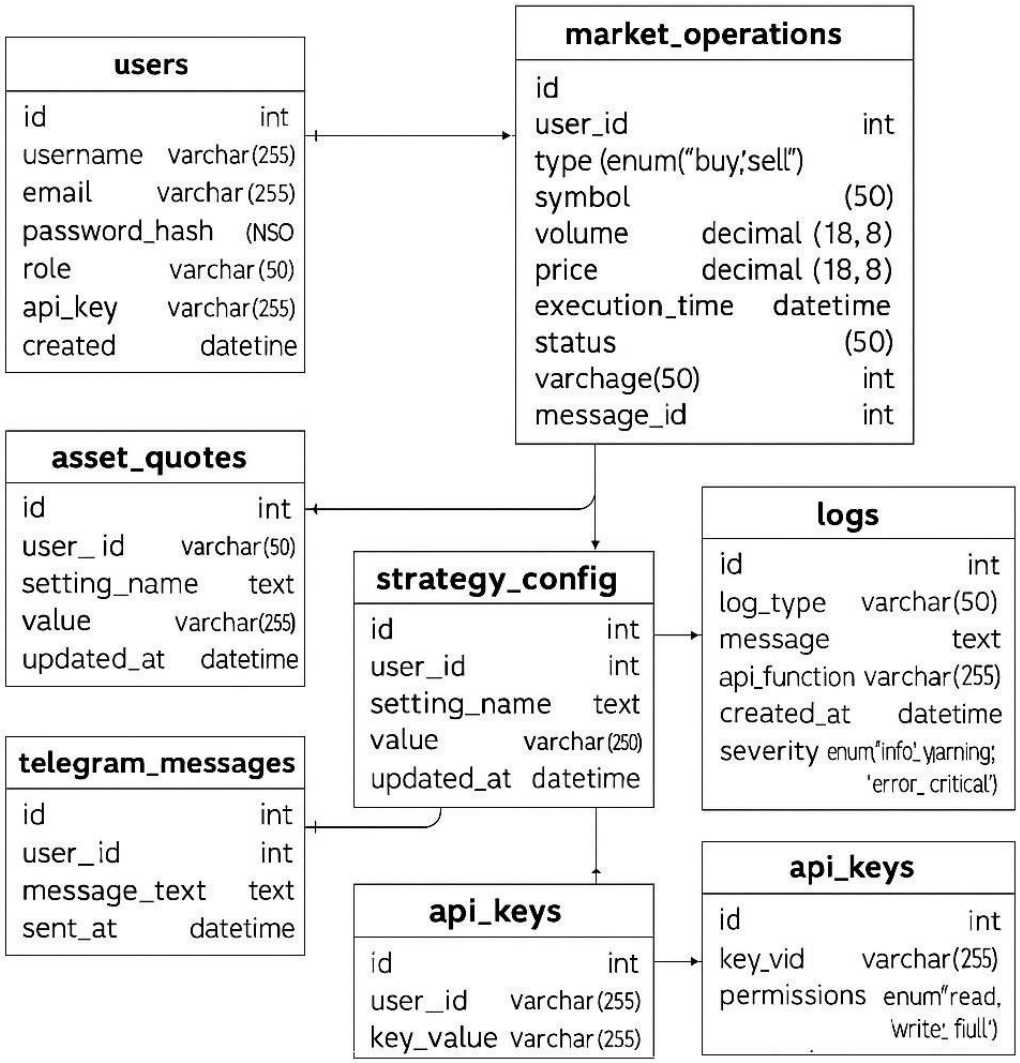

Рисунок 2.2 – Діаграма стуктури логічної моделі БД

Важливою характеристикою алгоритмічної частини торгової системи є її адаптивність і здатність до масштабування. Незважаючи на те, що на початковому етапі реалізовано базову стратегію на основі порогових значень, вона винесена в окремий модуль, що дозволяє без значних змін у загальній архітектурі замінювати або розширювати її іншими підходами. Зокрема, у майбутньому можлива інтеграція більш складних торгових стратегій – таких як методи на основі ковзних середніх (SMA/EMA), технічних індикаторів (RSI, MACD тощо) або алгоритмів, що використовують машинне навчання для прогнозування цінової динаміки.

Така модульна побудова забезпечує гнучкість у розвитку системи, відкриваючи можливості для її подальшого дослідження, оптимізації та вдосконалення вже за межами кваліфікаційної роботи бакалавра.

Окрему увагу в процесі розробки приділено ретельному тестуванню кожного з ключових компонентів системи та застосованих підходів. Зокрема, проводиться перевірка коректності взаємодії з API біржі Binance, достовірності та цілісності збереження даних у базі MySQL, стабільності роботи Telegram-бота під навантаженням, а також стійкості всієї системи до зовнішніх збоїв і нестабільності мережевих підключень.

Таке тестування є дуже важливим для забезпечення надійності та безперервної роботи системи в реальних умовах. Воно дозволяє впевнено стверджувати, що створене програмне забезпечення не лише виконує задані функції, але й здатне стабільно працювати протягом тривалого часу в умовах динамічного та непередбачуваного ринку.

У результаті системного підходу до вибору інструментів та методів реалізації вдалося побудувати технічну базу, яка поєднує в собі надійність, зручність експлуатації, можливість масштабування та гнучкість до подальших змін. Розроблене програмне забезпечення демонструє потенціал як для навчальних цілей, так і для адаптації до практичного використання в реальних умовах криптовалютного ринку.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра було проведено всебічний технічний аналіз поставленої задачі, визначено функціональні та нефункціональні вимоги до програмного забезпечення, а також виконано його детальне проєктування. На основі цього сформовано цілісне уявлення про структуру програмного продукту для автоматизованої торгівлі на криптовалютному ринку, визначено ключові модулі системи, принципи їх взаємодії та особливості інтеграції з зовнішніми сервісами.

У ході аналізу встановлено, що ефективна робота такої системи можлива лише за умови раціонального поєднання кількох технологічних компонентів. До основних інструментів, що забезпечують стабільність, гнучкість і функціональність системи, належать мова програмування Python, бібліотека для доступу до біржових API – ccxt, Telegram API для реалізації інтерфейсу користувача, а також система керування базами даних MySQL, яка відповідає за збереження і обробку структурованої інформації.

Розроблене архітектурне рішення комплексно охоплює всі етапи функціонування системи – від отримання ринкових даних через API Binance до їх обробки, збереження в структурованій базі даних і подальшої взаємодії з користувачем за допомогою Telegram-бота. Основу архітектури становить модульний підхід, що дозволяє забезпечити високу гнучкість у процесі проєктування, спрощує тестування окремих компонентів і створює передумови для подальшого розширення функціональності системи.

Зокрема, винесення торгового алгоритму в окремий модуль дозволяє змінювати або доповнювати стратегії без впливу на інші підсистеми, що значно підвищує адаптивність рішення. Структура бази даних оптимізована для зберігання великого обсягу історичних даних з можливістю їх ефективної обробки та аналітики, що є важливим для побудови звітності, аналізу ефективності та подальшого вдосконалення торгових підходів.

Окрему увагу в другому розділі приділено обґрунтуванню вибору технологічного стеку, що ліг в основу реалізації системи. Після порівняльного аналізу наявних рішень було прийнято виважені та аргументовані рішення на користь тих інструментів, які найбільш повно відповідають вимогам до стабільності, продуктивності та масштабованості програмного забезпечення.

Означено, що використання мови програмування Python обґрунтовано її високою гнучкістю, підтримкою процедурного та об’єктно-орієнтованого підходів, а також широким вибором бібліотек, зокрема для роботи з API, обробки даних і реалізації торгових стратегій. Python забезпечує ефективну інтеграцію між усіма компонентами системи та дозволяє швидко адаптувати код до нових умов.

Telegram було обрано як клієнтський інтерфейс завдяки його широкій популярності, простоті використання та підтримці різноманітних форматів взаємодії – повідомлень, команд, кнопок. Важливою перевагою є відсутність потреби у додатковому програмному забезпеченні на стороні користувача, що значно знижує вхідний поріг і робить систему доступною для широкого кола потенційних користувачів. Telegram-бот виступає зручним каналом комунікації з торговим ядром, забезпечуючи оперативну взаємодію у режимі реального часу.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОЇ ТОРГІВЛІ НА БІРЖІ BINANCE

3.1 Практична реалізація об’єкта проєктування

Реалізація об’єкта проєктування становить ключовий етап кваліфікаційної роботи бакалавра, у межах якої здійснюється розробка програмного забезпечення згідно з попередньо сформульованими вимогами, визначеною архітектурою та обраними технологічними підходами. У даному випадку реалізується повнофункціональна програмна система для автоматизованої торгівлі на криптовалютному ринку, що здійснює інтеграцію з біржею Binance через її API, обробляє торгові дані, виконує алгоритми купівлі та продажу активів, а також забезпечує взаємодію з користувачем за допомогою Telegram-бота.

Початковим етапом реалізації програмного забезпечення стало формування його базової інфраструктури, зокрема, організація структури проєкту та налаштування середовища розробки. Функціональні частини системи були логічно виокремлені у незалежні компоненти: модуль для інтеграції з Binance через API, блок реалізації торгової логіки, компонент для взаємодії з користувачем через Telegram, база даних MySQL для зберігання інформаційних записів, а також допоміжні утиліти, що відповідають за обробку помилок, ведення журналу подій та контроль стану системи. Такий підхід дозволив чітко визначити обов’язки кожного модуля та забезпечити високу адаптивність при тестуванні і подальших змінах у проєкті.

На початковому етапі розробки було здійснено інтеграцію з API біржі Binance шляхом використання бібліотеки ccxt, яка надала доступ до ключових торгових операцій: отримання переліку торгових інструментів, зчитування актуальних ринкових даних, створення торгових ордерів, перевірка балансу користувача та доступ до історії транзакцій. Використання цієї бібліотеки дало змогу уникнути прямої взаємодії з HTTP-запитами, забезпечивши уніфікований

інтерфейс для роботи з API, що є критично важливим для збереження стабільності функціонування системи при високому навантаженні на мережеву інфраструктуру. Крім того, було реалізовано обробку типових помилок, зокрема збоїв підключення, перевищення лімітів запитів та отримання некоректних відповідей від сервера.

На другому етапі було реалізовано Telegram-бот за допомогою бібліотеки python-telegram-bot, який виконує функцію основного інтерфейсу взаємодії між користувачем та торговою системою. Бот обробляє ключові команди, зокрема запуск і зупинку торгового процесу, відображення поточного балансу, інформування про відкриття або закриття позицій, а також зміну основних параметрів торгової стратегії. Для контролю доступу реалізовано базову систему авторизації, яка дозволяє працювати з ботом виключно авторизованому користувачеві. Це забезпечує мінімально необхідний рівень захисту під час роботи через відкритий комунікаційний канал.

Наступним кроком стала розробка бази даних MySQL зі структурою, що включає таблиці для збереження інформації про торгову історію, поточні налаштування користувача, журнали подій, відповіді від API, а також ключові ринкові показники, зафіксовані під час виконання торгових операцій. Було впроваджено механізм автоматичного оновлення даних після кожної успішної транзакції або зміни внутрішнього стану системи. Такий підхід забезпечує повноцінне логування діяльності торгового бота, дозволяє здійснювати аудит взаємодій і, в разі потреби, відновлювати попередні конфігурації або аналізувати результативність застосованої стратегії на основі історичних даних.

Часткова ER-діаграма бази даних, що ілюструє зв’язки між основними таблицями, наведена у лістингу 3.1.

Лістинг 3.1 – Демонстрація бази даних, що ілюструє зв’язки між основними таблицями

```
CREATE TABLE Settings (
    settings_id INT AUTO_INCREMENT PRIMARY KEY,
    trade_strategy VARCHAR(100),
```

```
    risk_level VARCHAR(50),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    auth_token VARCHAR(255) NOT NULL,
    settings_id INT,
    FOREIGN KEY (settings_id) REFERENCES Settings(settings_id)
);

CREATE TABLE Trade_History (
    trade_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    symbol VARCHAR(20),
    action ENUM('buy', 'sell'),
    amount DECIMAL(18, 8),
    price DECIMAL(18, 8),
    timestamp DATETIME,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE Market_Data (
    market_id INT AUTO_INCREMENT PRIMARY KEY,
    symbol VARCHAR(20),
    price DECIMAL(18, 8),
    volume DECIMAL(18, 8),
    timestamp DATETIME
);
```

кінець лістингу 3.1

Торгова логіка реалізована на основі порогового контролю цінових значень активу: система аналізує, чи досягнуто визначеного рівня, що є сигналом для виконання операції купівлі або продажу згідно з заданими умовами. Основні параметри – вхідна ціна, цільовий рівень виходу, стоп-лосс і тейк-профіт – зберігаються в базі даних та можуть змінюватися користувачем у реальному часі через інтерфейс Telegram-бота. Передбачено механізм періодичного звернення до API біржі для отримання актуальних ринкових даних, а також реалізовано систему обробки типових помилок, таких як втрата мережевого підключення, некоректні параметри, недостатній баланс або аномальні відповіді від сервера.

У процесі розробки значна увага була зосереджена на забезпеченні надійної та безперебійної роботи системи. Було реалізовано механізм логування, який реєструє всі дії програми у відповідних лог-файлах, що дає змогу повністю відстежити її виконання та оперативно виявляти помилки. У випадку виникнення критичних збоїв система відразу надсилає сповіщення користувачу через Telegram та призупиняє торгові операції з метою запобігання можливим фінансовим втратам.

Під час реалізації було забезпечено кросплатформену сумісність системи, що дозволяє її запуск у будь-якому середовищі з підтримкою Python та доступом до інтернет-мережі. Такий підхід надає можливість гнучкого розгортання як на локальних пристроях, так і на віддалених серверах. Завдяки інтеграції сторонніх бібліотек та використанню конфігураційних файлів у форматах .env і .json, система залишається зручною в адмініструванні та придатною до подальшого масштабування. Опишемо архітектуру програмного забезпечення (рис. 3.1).

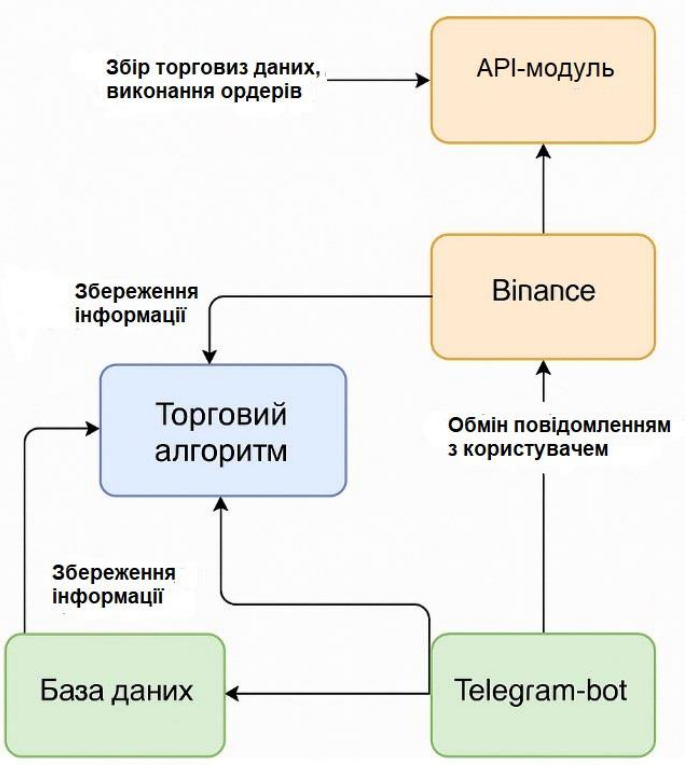

Рисунок 3.1 – Архітектура розроблюваного програмного забезпечення

Після наведеної схемної візуалізації, яка відображає загальну архітектуру програмного забезпечення для автоматизованої криптовалютної торгівлі, доцільно здійснити детальний аналіз функціональних взаємозв'язків між окремими модулями системи та пояснити їхнє прикладне значення в контексті реалізації проєкту.

У представленій схемі ключову роль відіграє модуль торгового алгоритму, що виступає основним елементом логічної структури системи. Цей компонент відповідає за обробку ринкових даних, отриманих через API-інтерфейс, та генерацію торгових рішень, таких як відкриття або закриття позицій, виставлення ордерів і запуск захисних механізмів у разі досягнення критичних показників. Усі дії алгоритму фіксуються в базі даних, що забезпечує ведення повного журналу операцій, дає змогу оцінювати ефективність застосованої стратегії та виконувати ретроспективний аналіз (backtesting).

API-модуль виконує роль інтерфейсу взаємодії між торговим алгоритмом і зовнішнім середовищем криптовалютної біржі Binance. Через цей компонент здійснюється отримання поточних ринкових котирувань, даних про обсяги торгів, статуси балансу користувача, а також передача запитів на відкриття, модифікацію або скасування торгових ордерів. Надійність та безперебійність роботи даного модуля є критично важливою умовою, оскільки він гарантує оперативне виконання торгових операцій у динамічному ринковому середовищі.

Telegram-бот є інтегрованим компонентом системи, що забезпечує зручний канал взаємодії з користувачем у мобільному середовищі. Він інформує про виконання торгових ордерів, фіксує та передає сповіщення про помилки чи нестандартні ситуації, а також надає можливість оперативного коригування торгових параметрів безпосередньо з мобільного пристрою. Завдяки реалізації обміну повідомленнями в режимі реального часу, бот підтримує інтерактивну взаємодію з користувачем, що дозволяє використовувати систему не лише в повністю автономному, а й у напівавтоматизованому режимі (рис. 3.2).

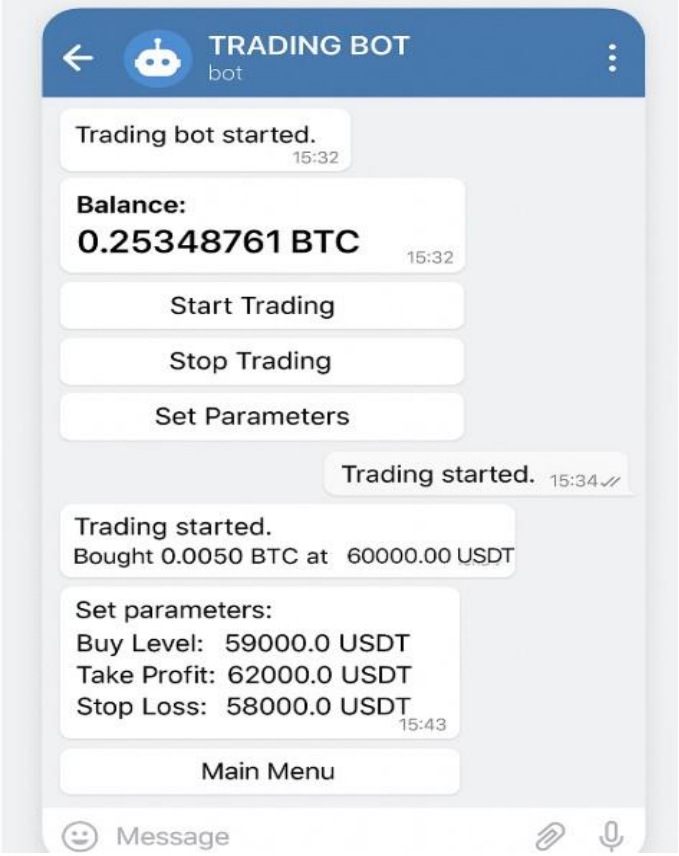

Рисунок 3.2 – Візуалізація розробленого продукту

База даних, що представлена на архітектурній схемі, виконує функцію централізованого сховища критично важливої інформації, зокрема історії торгів, журналів подій, користувацьких налаштувань, даних про баланс та активні торгові позиції. Такий підхід дає змогу не лише підтримувати актуальний стан усіх компонентів системи під час її роботи, але й гарантує можливість відновлення функціональності у разі непередбаченого завершення процесу або міграції на іншу серверну інфраструктуру.

Архітектурна структура системи побудована таким чином, щоб забезпечити безперервний і керований потік даних: від отримання ринкової інформації через API-модуль, її обробки торговим алгоритмом, фіксації всіх дій у базі даних до передачі результатів користувачу через Telegram-бот у реальному часі. Така логіка взаємодії компонентів сприяє високій продуктивності та надійності роботи системи, забезпечує прозорість процесів,

спрощує контроль над виконанням торгових операцій і створює передумови для подальшого масштабування.

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Завершальний етап впровадження інформаційно-комп’ютерної системи передбачає проведення комплексного тестування та налагодження, що дозволяє перевірити функціональність усіх компонентів, виявити потенційні помилки та оцінити відповідність програмного забезпечення встановленим технічним вимогам. Основна мета цього етапу полягає у забезпеченні стабільної роботи системи в умовах наближених до реальних, включаючи стандартні, граничні та стресові сценарії. Особлива увага приділяється перевірці стійкості до збоїв зовнішніх сервісів, втрати інтернет-з’єднання, помилок у введених користувачем даних, а також несправностей, пов’язаних з API-взаємодією.

У процесі тестування було застосовано поетапну стратегію, що охоплювала модульне, інтеграційне, системне та функціональне тестування. Кожен із ключових компонентів – зокрема торговий алгоритм, модуль API-взаємодії з Binance, Telegram-бот і база даних – проходив первинну перевірку в ізольованому середовищі. Такий підхід дозволив своєчасно виявити логічні помилки, проблеми сумісності форматів даних, недоліки в обробці виняткових ситуацій, а також оцінити продуктивність системи за умов високої частоти запитів.

Зокрема, для API-модуля було проведено навантажувальне тестування, яке визначило граничні показники частоти запитів до сервера Binance. За результатами аналізу було оптимізовано інтервали оновлення ринкових котирувань і викликів торгових функцій. Додатково було перевірено реакцію системи на типові збої з боку біржі: недоступність API, перевищення встановлених лімітів запитів, недостатність коштів на балансі користувача або спроби виконання ордерів із некоректними параметрами.

Telegram-бот було протестовано на відповідність основним командам: /start, /stop, /balance, /status, /config, які забезпечують базову функціональність взаємодії з користувачем. В рамках тестування перевірялися можливість запуску та зупинки торгового процесу, коректність відображення актуального стану системи, зміна параметрів алгоритму та отримання повідомлень про критичні події. Особливу увагу було приділено перевірці часу реакції бота, правильності форматування відповідей, а також реалізації захисту від несанкціонованого доступу шляхом авторизації за Telegram ID користувача.

Окремо були змодельовані ситуації втрати з’єднання з Telegram API або сервером Binance. У таких випадках система автоматично повідомляла користувача про зупинку торгової активності та виконувала відповідні захисні дії згідно з реалізованою логікою коду.

База даних MySQL була піддана повному циклу тестування з метою перевірки цілісності даних та коректності їх збереження. Було протестовано основні операції, включаючи запис історії виконаних ордерів, збереження конфігураційних параметрів, логування помилок і фіксацію повідомлень Telegram. Сценарії тестування охоплювали як нормальні умови експлуатації, так і критичні ситуації, зокрема перевантаження системи та аварійне завершення роботи.

У результаті випробувань підтверджено, що база даних стабільно зберігає всі події без втрат, а при повторному запуску програма успішно відновлює актуальний стан системи автоматично, без потреби у втручанні з боку користувача.

У ході системного тестування всі програмні модулі були інтегровані та запущені в єдиному середовищі з метою перевірки узгодженості їхньої взаємодії та цілісності функціонування системи. Зокрема, торговий модуль ініціював запити до API Binance, отримував поточні ринкові дані та передавав їх до логічного ядра, яке, керуючись заданими конфігураціями, ухвалювало рішення щодо створення ордерів. Telegram-бот паралельно інформував

користувача про кожну торгову дію, а база даних реєструвала всі події, забезпечуючи їх подальший аналіз і відтворення.

Всі процеси відбувалися в режимі реального часу, демонструючи мінімальну затримку та високу синхронізацію між компонентами системи. Після виконання кожної торгової операції Telegram-бот надсилав користувачу деталізований звіт з параметрами дії, що забезпечувало прозорість роботи системи та дозволяло оперативно коригувати торгові налаштування при потребі.

Окрім технічного тестування, було проведено верифікацію функціональності системи щодо відповідності заявленим вимогам. Перевірці підлягали повнота реалізації запланованих функцій, відповідність логіки торгівлі заданому алгоритму, зручність використання Telegram-інтерфейсу, а також потенціал масштабування системи – зокрема можливість додавання нових торгових пар, впровадження альтернативних стратегій та інтеграції з іншими торговими платформами.

Результати перевірки підтвердили стабільну роботу системи впродовж тривалого періоду, її здатність до відновлення після критичних збоїв, правильну обробку виняткових ситуацій і повну відповідність функціональним вимогам. Процес налагодження здійснювався поетапно шляхом запуску окремих модулів із використанням реальних API-ключів на тестовому акаунті Binance, що дозволило уникнути фінансових ризиків у період розробки.

Усі виявлені помилки фіксувалися, піддавалися аналізу, усувалися на рівні відповідного модуля та повторно перевірялися в умовах, наближених до реальної експлуатації. Такий підхід забезпечив досягнення високого рівня надійності та готовності системи до тривалого використання в реальних торгових середовищах.

Підсумки проведеного тестування та процесу налагодження підтверджують високу надійність і повну функціональну відповідність створеного програмного забезпечення технічним вимогам. Система продемонструвала стабільну роботу в автономному режимі, що свідчить про її

готовність до практичного використання. Результати тестування також засвідчили якісну реалізацію програмної логіки, ефективність алгоритмів торгівлі, зручність користувацької взаємодії через Telegram-інтерфейс, а також наявність потенціалу для подальшого масштабування – зокрема адаптації до більш складних торгових стратегій та розширення функціональності. Нижче наведемо інформацію з послідовністю етапів тестування (рис. 3.3).

Рисунок 3.3 – Візуалізація послідовності етапів тестування

Після наведеної візуалізації, що ілюструє послідовні етапи тестування та налагодження інформаційно-комп’ютерної системи, доцільно акцентувати увагу на значущості кожного з етапів у контексті досягнення основної мети – розробки стабільного, надійного та продуктивного програмного забезпечення.

Модульне тестування, яке є початковою фазою цього процесу, дає змогу ізольовано перевірити працездатність окремих функціональних компонентів, незалежно від загальної архітектури. Такий підхід дозволяє своєчасно виявляти логічні помилки, некоректну обробку вхідних даних або неправильне реагування на виключення, що запобігає поширенню помилок у межах усієї системи.

Наступним етапом є інтеграційне тестування, під час якого перевіряється коректність взаємодії між окремими модулями системи. Для даного програмного комплексу цей етап має особливу важливість через наявність складної архітектури, що включає взаємозв’язки між торговим алгоритмічним ядром, API Binance, Telegram-ботом і модулем бази даних. У межах тестування аналізуються шляхи передачі даних, відповідність форматів параметрів, час відгуку кожного модуля, а також поведінка системи в разі виникнення конфліктних ситуацій між компонентами. Це дозволяє підтвердити узгодженість дій усіх частин системи, уникнути дублювання функціональності та забезпечити уніфікований обмін інформацією між модулями.

Системне тестування, яке виступає третім етапом перевірки, охоплює аналіз функціонування програмного забезпечення в цілісному вигляді. У рамках цього етапу моделюються повноцінні сценарії взаємозв’язку користувача із системою, враховуючи ініціалізацію Telegram-бота, отримання ринкових сигналів через API, виконання торгових операцій (відкриття та закриття позицій), фіксацію даних у базі даних та надсилання інформаційних повідомлень. Такий підхід дозволяє виявити не лише логічні помилки, а й проблеми, що можуть виникнути внаслідок навантаження на систему, нестабільного мережевого з’єднання, обмежень API або дефіциту обчислювальних ресурсів.

Підсумкове функціональне тестування має на меті перевірку відповідності програмного забезпечення вимогам, визначеним у технічному завданні. У процесі тестування оцінюється повнота реалізації функціональності, зручність взаємодії з користувачем, коректність реакцій на введення, а також здатність системи адаптуватися до змін конфігурації. Результати перевірки підтвердили, що програмний продукт забезпечує запуск торгових операцій, моніторинг ключових параметрів у режимі реального часу, надсилання сповіщень і журналювання всіх дій у базі даних для подальшого аналізу.

Проведення повного циклу тестування та налагодження підтвердило високий рівень реалізації інформаційно-комп’ютерної системи, її відповідність актуальним стандартам автоматизованої торгівлі, а також придатність до впровадження в реальні умови експлуатації. Підсумковий етап практичного впровадження засвідчив, що створене програмне рішення є не лише функціонально повноцінним, але й технічно стабільним, захищеним і придатним для подальшого розширення функціоналу та модернізації.

3.3 Розробка бази даних

Проєктування бази даних становить один із визначальних етапів створення інформаційно-комп’ютерної системи для автоматизованої торгівлі криптовалютними активами. Саме база даних виконує функції централізованого зберігання, структурованого представлення та швидкого доступу до критично важливої інформації, необхідної для роботи торгових алгоритмів, взаємодії з API криптобіржі Binance, а також організації зв’язку з користувачем через Telegram-бот. У рамках реалізації кваліфікаційної роботи бакалавра було обґрунтовано вибір системи управління базами даних MySQL, яка характеризується стабільністю, сумісністю з мовою Python та широкими можливостями розгортання як у локальному середовищі, так і на хостингових платформах.

Процес проєктування бази даних розпочався зі створення її логічної моделі. На цьому етапі було визначено ключові об’єкти, інформація про які повинна зберігатися: операції на ринку, котирування активів, параметри торгової стратегії, повідомлення Telegram-бота, журнали з помилками та винятками, а також дані користувачів і API-ключі. Для кожної із зазначених сутностей було створено окрему таблицю з дотриманням принципів реляційної моделі, що забезпечує ефективне виконання запитів і зменшення надлишковості даних. У структурі таблиць реалізовано первинні ключі для унікальної

ідентифікації записів, індекси для оптимізації запитів і зовнішні ключі для встановлення логічних зв’язків між таблицями.

Зокрема, у таблиці торгових операцій зберігаються дані щодо типу ордера (купівля або продаж), дати й часу виконання, обсягу та ціни угоди, ідентифікатора торгової пари, стану виконання, а також посилання на відповідні записи у логах або повідомленнях, надісланих користувачу. Таблиця параметрів конфігурації містить назви налаштувань, їхні актуальні значення та дату останньої зміни, що дає змогу зберігати індивідуальні стратегії користувача та оперативно їх оновлювати за допомогою Telegram-бота. У таблиці журналів передбачено реєстрацію всіх викликів API-функцій, зафіксованих помилок, винятків і критичних подій, що дозволяє ефективно контролювати стан роботи системи та здійснювати діагностику у разі збоїв.

Для реалізації взаємодії з базою даних у програмному забезпеченні була використана технологія об’єктно-реляційного відображення (ORM) із застосуванням бібліотеки SQLAlchemy. Такий підхід дозволяє працювати з даними на рівні Python-об’єктів, що підвищує безпечність доступу та забезпечує централізоване управління логікою обробки інформації [15, 16]. Усі основні дії з базою – створення, оновлення, пошук, видалення записів – реалізовані у вигляді окремих функцій, що також включають спеціалізовані агреговані запити для отримання статистичних даних, фільтрації за часовими параметрами та аналізу ефективності торгових операцій. Додатково було впроваджено підтримку транзакцій, що гарантує атомарність виконання запитів і збереження цілісності даних навіть у разі виникнення помилок у роботі основного програмного модуля.

У проєктуванні структури бази даних було передбачено механізми обмеження доступу до конфіденційної інформації, зокрема до API-ключів та параметрів користувацької конфігурації. Для підвищення рівня безпеки реалізовано зберігання облікових даних у зашифрованому вигляді, а також впроваджено авторизований доступ із розмежуванням прав – окремі програмні модулі мають доступ лише для читання до певних таблиць. Такий підхід

дозволить знизити ймовірність несанкціонованого доступу або втрати даних у разі збоїв Telegram-компонента чи стороннього втручання.

Значну увагу було приділено організації резервного копіювання та забезпеченню можливості відновлення даних. Було розроблено механізм автоматичного експорту критичних таблиць у форматі .sql із заданою періодичністю, що дає змогу швидко перенести систему на інший сервер або відновити її працездатність після збоїв без втрати накопичених даних. Крім того, реалізовано логіку первинної ініціалізації бази даних: у разі відсутності необхідних таблиць під час старту застосунку система автоматично формує їхню структуру відповідно до описаних ORM-моделей.

У процесі експлуатації розроблена база даних показала стабільну продуктивність, гарантуючи високу швидкість обробки запитів і оперативний доступ до критично важливої інформації (рис. 3.4). Завдяки використанню оптимізованих індексів та ефективній структурі таблиць вдається знизити навантаження на систему навіть при інтенсивній роботі торгового алгоритму.

Була проведена детальна нормалізація даних для усунення надмірностей і підвищення цілісності інформації. Крім того, реалізовано систему логування всіх ключових подій у спеціалізованій таблиці logs, що дозволяє відслідковувати помилки та контролювати виконання API-запитів.

Інтеграція з Telegram-ботом забезпечила можливість оперативного інформування користувача про важливі події або помилки у роботі системи. Автоматичне формування повідомлень та звітів базується на інформації з таблиць market_operations та strategy_config, що забезпечує високу гнучкість моніторингу.

Таким чином, розроблена база даних не лише відповідає сучасним вимогам до збереження й обробки інформації, а й гарантує масштабованість і розширюваність системи в майбутньому.

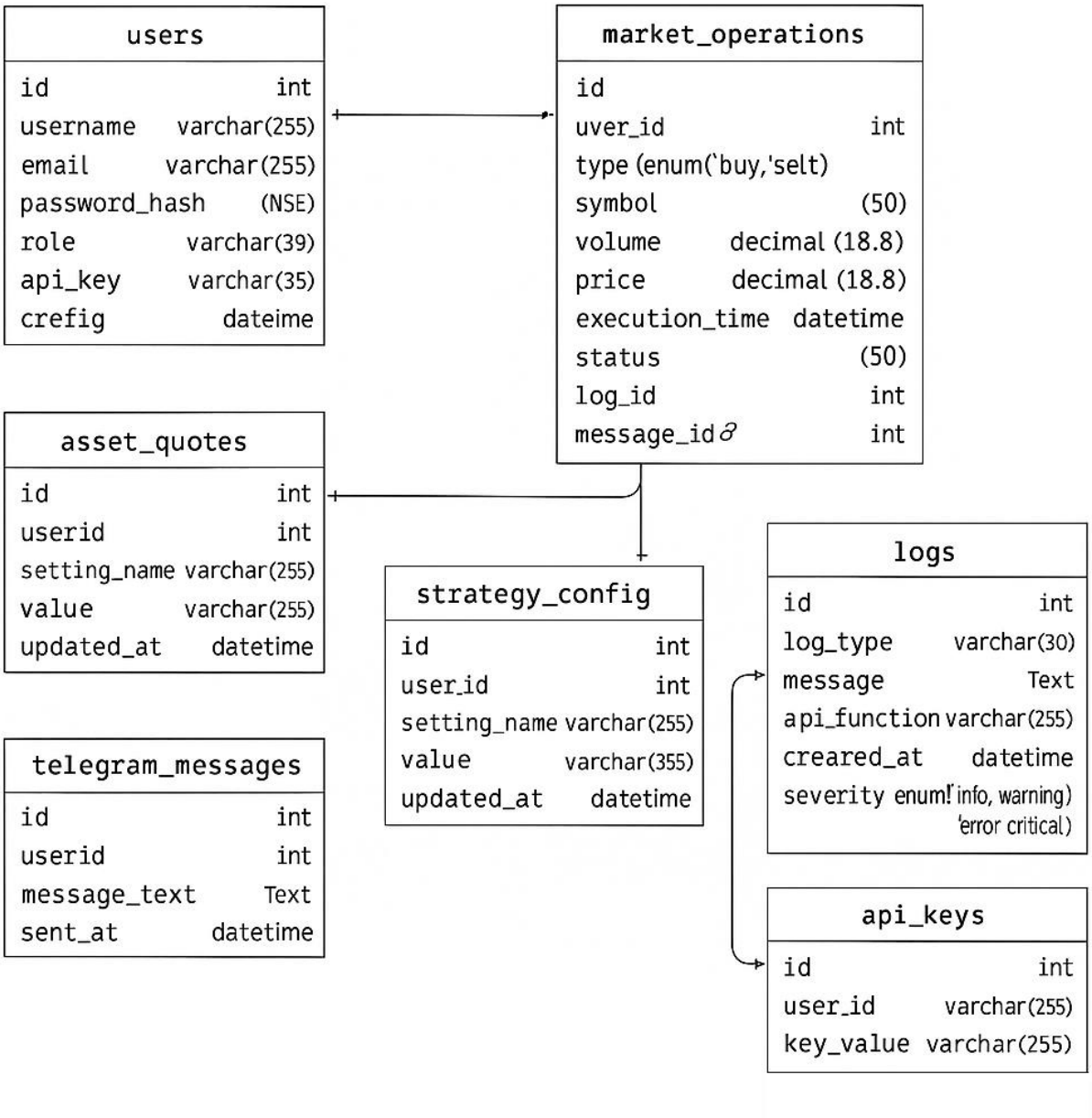

Рисунок 3.4 – ERD-діаграма БД

3.4 Захист інформаційно-комп’ютерної системи

Забезпечення безпеки інформаційно-комп’ютерної системи, яка автоматизує торгівельні операції на криптовалютному ринку, є критично важливим завданням. Система обробляє чутливу інформацію, включаючи облікові дані користувачів на платформі Binance, API-ключі та фінансові транзакції, що мають високу значущість. Потенційні загрози безпеці можуть спричинити несанкціонований доступ до торговельного функціоналу, втрату

фінансових ресурсів або викривлення даних, що здатне завдати серйозної шкоди. У зв’язку з цим при проєктуванні та реалізації системи було впроваджено комплекс заходів, орієнтованих на дотримання принципів конфіденційності, цілісності та доступності.

У першу чергу було впроваджено політику обмеженого доступу до системного середовища. Єдиним інтерфейсом для взаємодії користувача із системою виступає Telegram-бот, який обробляє команди виключно від верифікованого користувача, Telegram ID якого попередньо збережено у базі даних. У разі надходження запиту від неавторизованого джерела система автоматично його ігнорує, не здійснюючи передачу жодної інформації у відповідь. Такий підхід дозволяє ефективно запобігати несанкціонованому доступу до механізмів керування торгівельною діяльністю.

Захист API-ключів реалізовано з підвищеним рівнем безпеки, оскільки вони забезпечують безпосередній доступ до біржового облікового запису користувача. У базі даних ключі зберігаються лише в зашифрованому вигляді – перед збереженням застосовуються криптографічні алгоритми симетричного (AES) або асиметричного (RSA) типу. Розшифрування виконується лише під час ініціалізації системи, після чого ключі завантажуються до оперативної пам’яті. Подальше зберігання у файловій системі або логах не здійснюється, що істотно знижує ймовірність їх компрометації навіть у випадку несанкціонованого доступу до бази даних.

Для гарантування безпеки внутрішніх процесів системи впроваджено багаторівневий захист, що включає обробку виключних ситуацій, перевірку типів вхідних даних, валідацію користувацьких параметрів, а також аналіз відповідей від API сервера Binance. Кожна функціональна складова містить механізми обробки помилок і відновлення, що дає змогу запобігати критичним збоям та небажаним сценаріям, таким як повторна відправка ідентичного ордера або блокування облікового запису внаслідок надмірної кількості некоректних запитів.

Система оснащена багаторівневим механізмом журналювання подій, який охоплює як типові операції (наприклад, відкриття та закриття ордерів), так і нетипові ситуації, зокрема спроби несанкціонованого доступу, помилки мережевого з’єднання, перевищення API-лімітів, недостатній баланс та інші аномальні запити. Такий підхід до логування забезпечує ефективне виявлення функціональних збоїв і дозволяє оперативно реагувати на загрози безпеці або технічні відхилення. Усі журнали подій захищені від модифікацій та зберігаються у вигляді зашифрованих архівів з обмеженим доступом, що підвищує загальний рівень стійкості системи до внутрішніх і зовнішніх втручань.

Для забезпечення безпечного обміну даними між ботом, серверною частиною та базою даних у системі реалізовано використання захищених мережевих протоколів. При розгортанні на віддаленому сервері обов’язковою умовою є застосування HTTPS для всіх веб-запитів, шифрування з’єднання через SSH при доступі до бази даних, а також захист облікових даних і конфігураційних файлів. Усі критично важливі параметри – такі як API-ключі, паролі та Telegram-токени – зберігаються у файлі .env, який ізольований від загальнодоступних середовищ, не включається до публічних репозиторіїв і недоступний для неавторизованих користувачів.

У разі виявлення критичного збою або втрати з’єднання з сервером біржі система автоматично переходить у захищений режим роботи. Це передбачає призупинення всіх торгових операцій, фіксацію поточного стану в базі даних, генерацію повідомлення для користувача про інцидент, а також перехід у режим очікування до моменту ручного перезапуску, ініційованого адміністратором. Такий механізм мінімізує ризик подальших фінансових втрат і виключає виконання некоректних дій у середовищі з порушеною стабільністю.

У контексті забезпечення захищеної роботи інформаційно-комп’ютерної системи важливим компонентом виступає система моніторингу в реальному часі. Реалізація механізму активного сповіщення забезпечує оперативне інформування користувача про критичні події, включно з втратами з’єднання з

API Binance, змінами параметрів торгової стратегії, нетиповою динамікою торгової активності або неочікуваним завершенням сесії. Такий підхід дозволить вчасно виявляти відхилення у роботі системи та оперативно вживати заходів для зниження ризику втрати контролю або фінансових збитків.

Одним із ключових елементів безпеки системи є контроль її цілісності, який реалізується через вбудовані механізми діагностики стану критичних модулів. Зокрема, під час ініціалізації програми виконується перевірка структури бази даних, наявності необхідних таблиць, відповідності типів полів, а також актуальності резервних копій. У разі виявлення пошкоджених даних або відсутності критичних конфігурацій система автоматично інформує користувача про інцидент і призупиняє виконання торговельної логіки до усунення виявлених проблем.

З метою підвищення безпеки в систему було інтегровано механізм автоматичного контролю частоти запитів до API. Це дозволяє дотримуватись обмежень, встановлених платформою Binance, і запобігати тимчасовому блокуванню облікового запису через перевищення лімітів. Такий механізм також мінімізує ризик зловживання функціоналом застосунку для реалізації атак або навмисного перевантаження інфраструктури. У разі досягнення критичного порогу запитів система автоматично активує режим зниженого навантаження та інформує користувача про зміну режиму роботи.

Захист на рівні файлової системи реалізовано з особливим акцентом на обмеження доступу до конфігураційних файлів, що містять критичну інформацію – такі як токени, шляхи до БД та системні змінні. Доступ до цих файлів обмежено на рівні прав доступу, що дозволяє їх використання виключно головному виконуваному процесу або адміністративно авторизованим користувачам. Зовнішні модулі, що взаємодіють із системою, здійснюють обов’язкову перевірку цифрових сигнатур вхідних даних та блокують виконання невалідованого коду.

З огляду на безперервний режим функціонування системи, її безпека значною мірою залежить від актуального стану програмного середовища. У

цьому контексті реалізовано автоматизований контроль версій залежностей, зокрема таких як ccxt, python-telegram-bot і mysql-connector, що забезпечує своєчасне оновлення уразливих компонентів. Такий підхід гарантує відповідність сучасним вимогам інформаційної безпеки та дозволяє ефективно протидіяти загрозам, пов’язаним із використанням застарілих бібліотек із відомими вразливостями.

3.5 Розробка документації для програмного продукту

Завершальним, але критично важливим етапом розробки інформаційно-комп’ютерної системи є формування супровідної документації. Вона виконує функцію ключового джерела інформації щодо архітектури, принципів роботи та правил експлуатації програмного забезпечення для всіх зацікавлених сторін – від користувачів до технічного персоналу. Документація виступає не лише засобом користування системою, а й інструментом для її ефективної підтримки, технічного обслуговування, розширення функціоналу та подальшої масштабованості. У рамках дипломного проєкту було сформовано повний комплект документаційних матеріалів, які охоплюють як технічну, так і організаційну складову експлуатації програмного продукту.

Центральним елементом супровідної документації є технічний опис системи, в якому детально представлено архітектурну модель програмного забезпечення. У цьому розділі здійснено модульний розбір функціональних компонентів, наведено обґрунтування вибору застосованих технологій, алгоритмів та інструментів реалізації. Особлива увага приділяється логіці взаємодії основних складових системи – торгового ядра, модуля API, Telegram-бота та підсистеми бази даних.

Документація також включає детальний опис структури бази даних: наведено перелік таблиць, їхні атрибути, типи даних, реляційні зв’язки та принципи зберігання інформації. Такий рівень деталізації дає змогу технічним спеціалістам – як розробникам, так і адміністраторам – швидко орієнтуватися у

внутрішній архітектурі системи та безпечно вносити зміни до конфігурації без ризику порушення цілісності логіки чи втрати даних.

Окремим розділом документації є інструкція користувача, представлена у форматі поетапного керівництва з експлуатації системи. У цьому документі детально описано процес запуску програмного забезпечення, початкове налаштування середовища, вимоги до заповнення конфігураційних файлів, а також алгоритм взаємодії з Telegram-ботом – від ініціалізації торгової сесії до внесення змін у параметри торгової стратегії.

Інструкція містить опис усіх доступних команд, доповнений прикладами запитів і типових відповідей системи. Окремо наведено перелік потенційних помилок, що можуть виникнути під час роботи, із вказівками щодо їх діагностики та методів усунення. Такий підхід дозволяє користувачеві ефективно взаємодіяти з системою навіть без глибоких технічних знань.

Ключовим елементом документації є розділ, присвячений засобам забезпечення безпеки. У ньому висвітлено методи захисту конфіденційних даних, зокрема принципи зберігання та використання API-ключів, реалізацію механізмів автентифікації Telegram-користувача, а також політики обмеження доступу до бази даних.

Окрему увагу приділено системі логування – з фіксацією критичних подій і можливістю оперативного сповіщення відповідального персоналу. Крім того, описано механізми резервного копіювання даних і процедури відновлення функціонування системи в разі аварійних ситуацій. Зміст цього розділу є особливо значущим для адміністратора, відповідального за стабільність і безперервність роботи системи у продукційному середовищі.

Для забезпечення можливості масштабування та подальшого розвитку системи підготовлено програмну документацію, яка охоплює структуру директорій, опис функцій, класів, змінних, а також деталізовану логіку обробки даних. Кодова база супроводжується коментарями до ключових функціональних блоків, що полегшує розуміння внутрішньої логіки роботи системи іншими розробниками.

Коментарі містять вказівки щодо залежностей, які слід враховувати при зміні або розширенні функціоналу, а також відображають існуючі обмеження, пов’язані з модифікацією алгоритмів. Такий підхід значно знижує ймовірність помилок у процесі інтеграції нових торгових стратегій, доопрацювання функціоналу або адаптації системи під інші торгові платформи.

Окремо підготовлено файл README, який слугує стислим довідником для первинного ознайомлення з програмним продуктом. У ньому міститься опис призначення системи, специфікації до середовища виконання (включаючи рекомендовану версію Python, необхідні бібліотеки та драйвери), а також покрокова інструкція з інсталяції, ініціалізації бази даних та запуску основного виконуваного модуля.

Наявність README-файлу значно спрощує процес розгортання системи на новому сервері або передачу проєкту іншим розробникам, забезпечуючи швидкий вхід у роботу без необхідності глибокого аналізу всієї кодової бази.

Створення повного пакета документації забезпечило всебічний супровід програмного забезпечення, зробивши його зрозумілим та доступним для кінцевих користувачів, системних адміністраторів і розробників. Документаційні матеріали не лише виконують роль технічного опису, а й формують основу для подальшого розвитку, масштабування та безпечного впровадження системи в експлуатацію.

Наявність структурованої документації гарантує стабільну роботу системи, підвищує зручність її використання та сприяє надійності функціонування у виробничому середовищі.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи бакалавра детально описано практичну реалізацію інформаційно-комп’ютерної системи для автоматизованої торгівлі криптовалютами. Особлива увага приділена архітектурі програмного забезпечення, розробці ключових модулів, алгоритмам взаємодії з

криптобіржею Binance, побудові бази даних, а також процесам тестування та налагодження системи. Результатом цих робіт стало повнофункціональне програмне рішення, здатне автономно виконувати торгові операції згідно із заданою логікою, забезпечувати збереження даних у базі, надавати керування через Telegram-інтерфейс і працювати у реальному часі без втручання користувача.

Одним із основних досягнень етапу стало впровадження архітектурної моделі з чітким розподілом відповідальностей між компонентами системи. Такий підхід забезпечив створення гнучкої, масштабованої та безпечної структури, де кожен модуль – API-взаємодія, торговий алгоритм, база даних і Telegram-бот – функціонує незалежно, але узгоджено із загальною системою. Це сприяє легкому внесенню змін, додаванню нових торгових стратегій та параметрів без необхідності зупинки роботи, а також подальшому розширенню функціоналу.

Реалізовано повний цикл обробки даних: отримання ринкових котирувань через бібліотеку ccxt, обробка торговим алгоритмом, формування ордерів та їх фіксація у базі даних. Telegram-бот надає зручний текстовий інтерфейс, що дозволяє користувачу оперативно взаємодіяти із системою, змінювати параметри стратегії, контролювати процес торгівлі, отримувати повідомлення про статус угод і помилки. Такий інтерфейс робить систему зручною для щоденного використання без необхідності запуску додаткового графічного середовища.

Особливу увагу було приділено питанням безпеки: захисту API-ключів, авторизації користувачів, логуванню операцій і резервному копіюванню даних. Це гарантує надійність роботи системи в умовах реального криптовалютного ринку, де будь-які збої або несанкціонований доступ можуть спричинити фінансові втрати. Система продемонструвала стабільність під час тестових сесій, підтверджуючи ефективність обраної архітектури та реалізованих алгоритмів.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було здійснено аналіз стану криптовалютного ринку, зокрема, таких систем, як Bybit, Coinbase та Kraken, що дозволило оцінити значний потенціал і доцільність впровадження автоматизованих торгових систем. Це дослідження допомогло глибше зрозуміти актуальні тенденції та виклики, з якими стикаються трейдери, а також виявити переваги та обмеження існуючих торгових ботів.

Було проведено детальний огляд існуючих програмних рішень для автоматизованої торгівлі, таких, як 3Commas, Cryptohopper, HaasOnline, безкоштовних або умовно безкоштовних open-source продуктів, зокрема Zenbot, Gekko чи скриптів на основі бібліотек ccxt та python-binance, що дало змогу аргументовано обґрунтувати необхідність створення нового програмного продукту, який би враховував сучасні вимоги до гнучкості, безпеки та зручності використання.

Наступним кроком стала розробка продуманої архітектури системи, що інтегрує Telegram-інтерфейс для зручної взаємодії користувача, надійну API-взаємодію з біржею Binance для отримання актуальних даних і здійснення торгових операцій, а також ефективну базу даних для зберігання ключової інформації.

Реалізовано функціонал, який забезпечує просту і водночас безпечну роботу з торговим ботом: користувачі можуть реєструвати API-ключі, моніторити ринкові умови та здійснювати розміщення торгових ордерів без зайвих труднощів. Це значно спрощує процес управління автоматизованою торгівлею навіть для тих, хто лише починає працювати з криптовалютами.

Окрему увагу було приділено розробці надійної структури бази даних, що дозволяє ефективно зберігати інформацію про всі транзакції, налаштування користувачів та історію їх взаємодій із системою, забезпечуючи цілісність та доступність даних.

Для захисту конфіденційної інформації впроваджено сучасні механізми хешування та шифрування API-ключів, що гарантує безпеку і запобігає несанкціонованому доступу.

Ретельне тестування програмного продукту в умовах, максимально наближених до реальних торгових сценаріїв, підтвердило стабільність, надійність і точність роботи системи.

Наприкінці роботи розроблено детальні рекомендації щодо встановлення, налаштування та експлуатації програмного забезпечення, що забезпечить комфортне впровадження системи в різних середовищах та полегшить її використання кінцевими користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Батракова Т. І. Турубарова Я. О. Криптовалюта в світі: стан, регулювання та перспективи. Економічні студії. 3 (21). URL: https://journals.indexcopernicus.com/api/file/viewByFileId/623706.pdf (дата звернення: 11.03.2025 р.).

2. Криптовалюта. URL: https://forbes.ua/tags/kriptovalyuta (дата звернення: 11.03.2025 р.).

3. Олійник А. А. Денкова І. М. Криптовалюти як новий інструмент на міжнародних фінансових ринках: можливості та виклики для глобальної економіки. Економіка і суспільство. Випуск # 56. 2023. С.1-6.

4. Федорова Ю. В. Криптовалюти та їх місце у фінансовій системі. Економіка і суспільство. Випуск # 15. URL: https://economyandsociety.in.ua/journals/15_ukr/116.pdf (дата звернення: 12.03.2025 р.).

5. Найкращі тенденції інтеграції та управління API у 2023 році. URL: https://stfalcon.com/uk/blog/post/Best-API-Integration-and-Management-Trends (дата звернення: 10.04.2025 р.).

6. OpenProject API. URL: https://www.openproject.org/ (дата звернення: 10.04.2025 р.).

7. Слабінога М. О. Бойчук Т. В. Фещак Д. О. Автоматизація процесу прийому та опрацювання заявок засобами telegram-бота та програмного інтерфейсу системи OpenProject/ Вісник Херсонського національного технічного університету, № 3(90). 2024. С. 293-298.

8. Реляційні бази даних: усе, що необхідно про них знати. URL: https://foxminded.ua/reliatsiini-bazy-danykh/ (дата звернення: 20.04.2025 р.).

9. Реляційні бази даних: структура і застосування. URL: https://www.run-it.com.ua/reliatsijni-bazy-danykh-struktura-i-zastosuvannia/ (дата звернення: 20.04.2025 р.).

10. Балик Н. Р., Мандзюк В. І. Бази даних MySQL: навчальний посібник. Тернопіль: Навчальна книга. URL: https://bohdan-books.com/upload/iblock/ebe/ebe73ff67804bc6dc1f6f464a3ca4469.pdf?srsltid=AfmBOornuTBUrnCbDeu4WVWj8oURAUTT1heXgpwxdCIXmMRDrNxgdT5k (дата звернення: 20.04.2025 р.).

11. База даних MySQL. URL: https://promoter.net.ua/articles/baza-danix-mysql.html (дата звернення: 02.05.2025 р.).

12. Підручник з Python. URL: https://docs.python.org/uk/3.13/tutorial/index.html (дата звернення: 02.05.2025 р.).

13. Про схеми баз даних. URL: https://foxminded.ua/skhemy-bazy-danyh/ (дата звернення: 02.05.2025 р.).

14. Костюченко А. О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. 180 с.

15. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/ (дата звернення: 21.05.2025 р.).

16. Васильєв О. М. Програмування в Python. Теорія і практика: навч. посіб. Київ: Видавництво Ліра-К, 2023. 462 с.