МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА

Навчально-науковий інститут

«Інститут інформаційних технологій в економіці»

Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»

галузь знань 12 «Інформаційні технології»

спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему:

«Розробка веб-застосунку криптовалютної біржі»

здобувача Сороки Дмитра Сергійовича _____

Науковий керівник:
к.е.н., доцент
_____Ситник Н.В.

Проєкт допущений до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент.
_____Тішков Б.О.

Київ 2024

Міністерство освіти і науки України
Київський національний економічний університет імені Вадима Гетьмана
Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

ПОГОДЖЕНО:	**ЗАТВЕРДЖУЮ:**
Керівник проєктної групи(гарант)	**завідувач кафедри**
освітньо-професійної програми	**_____Тішков Б.О.**
_____Іванченко Г.Ф.	**"____" _____2024 р.**
«____» _____**2024** р.	

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти : **Сороки Дмитра Сергійовича** ***очної (денної) форми навчання***
на підготовку кваліфікаційного бакалаврського проєкту
на тему: « Розробка web-застосунку криптовалютної біржі»

Тему затверджено наказом ректора Університету від «11» березня 2024 р. № 529-ст.
Кваліфікаційний бакалаврський проєкт виконується на матеріалах *наукових публікацій та матеріалів з мережі інтернет*

План кваліфікаційного бакалаврського проєкту
Розділ І *Характеристика та аналіз предметної області*

Розділ II *Розробка вимог і моделювання інформаційної системи.*

Розділ III *Проєктування та реалізація компонентів системи.*

Об'єкт дослідження *інформаційні процеси, що характеризують діяльність криптовалютної біржі.*

Предмет дослідження *сукупність теоретичних, методичних і прикладних аспектів з моделювання, проєктування та WEB-технології для підвищення ефективності функціонування криптовалютної біржі.*

Мета кваліфікаційного бакалаврського проєкту *систематизація і узагальнення сучасного досвіду розробки нових рішень та розробка web-застосунку криптовалютної біржі.*

Конкретні завдання, які студент повинен виконати для досягнення поставленої мети:

У розділі I *Описати предметну галузь та інформаційні процес які характеризують роботу криптовалютної біржі. Навести характеристику об'єкта дослідження. Подати результати аналізу літературних джерел та існуючих web-застосунків криптовалютних бірж.*

У розділі II *Визначити бізнес-вимоги, функціональні та нефункціональні вимоги до проєктованої web-системи та подати їх у формі діаграм і проєктної специфікації, розроблених з використанням CASE-засобів та мови SysML. Подати постановку та алгоритм розв'язання досліджуваної задачі. Розробити модель системи в специфікації мови UML.*

У розділі III *Виконати проєктування компонентів інформаційної системи, зокрема інформаційного, технічного та програмного забезпечення. Виконати реалізацію web-застосунку криптовалютної біржі, привести відповідні результати, вказавши можливість та доцільність застосування їх на практиці.*

Завдання підготував

науковий керівник  **Ситник Ніна Василівна**

"14 березня" 2024 р.

Завдання одержав

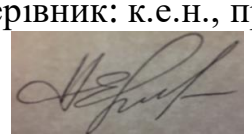
здобувач _____ **Сорока Дмитро Сергійович**

"14 березня" 2024 р.

В і д г у к
на кваліфікаційний бакалаврський проєкт
здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»
галузі знань 12 «Інформаційні технології»
спеціальності 122 «Комп'ютерні науки»
Сороки Дмитра Сергійовича
на тему: «Розробка web-застосунку криптовалютної біржі»

1. **Актуальність теми:** Віртуалізація та цифрові технології сьогодення отримали поширення у фінансовій сфері, зокрема широко застосовуються в роботі криптовалютних бірж. Саме тому, розробка веб-застосунки криптовалютної біржі, присвячего обміну криптовалют з пошуком мінімального курсу є актуальною та цікавою для трейдерів.
2. **Позитивні риси кваліфікаційного бакалаврського проєкту**: В рамках завдання на кваліфікаційний бакалаврський проєкт проаналізовано бізнес-процеси та бізнес-вимоги до функціонування криптовалютних бірж та враховано їх при проєктуванні web-застосунку. Запропонована веб-система зорієнтована на використання новітніх технологій в організації роботи криптовалютних бірж. Зокрема, використано інноваційний алгоритм виявлення та маршрутизації 1inch API, який забезпечує обмін активами та API CoinGecko, який надає актуальні криптовалютні ціни, забезпечуючи тим самим ефективне проведення транзакцій за найпривабливішими курсами.
3. **Наявність самостійних розробок автора**: Представлені у кваліфікаційному бакалаврському проєкті розробки виконані самостійно з застосуванням сучасного інструментарію. Захищений обмін криптовалют забезпечує мережа Binance Smart Chain та електронний криптовалютний гаманець MetaMask, який інтегрований в систему. Нереляційна документо-орієнтована база даних MongoDB запропонована для зберігання даних транзакцій..
4. **Цінність теоретичних висновків та практичних рекомендацій:**. Пропозиції, які надані в проєкті, є безперечно цікавими та практично зорієнтованими, вони відповідають вимогам сучасних фінансових ринків. Застосування інноваційних технологій дозволить досягти безпечного та швидкого виконання обмінних криптовалютних транзакцій за найкращими цінами та з низькою комісією.
5. **Загальна оцінка кваліфікаційного бакалаврського проєкту та його допущення до захисту перед ЕК:** У цілому, представлений кваліфікаційний бакалаврський проєкт відповідає встановленим вимогам методичних вказівок щодо структури, обсягу та змісту та рекомендується до захисту з **високою оцінкою**, а **Сороці Дмитру Сергійовичу** може бути присвоєний освітньо-кваліфікаційний рівень бакалавр галузі знань 12 «Інформаційні технології» за спеціальністю 122 «Комп'ютерні науки».

Науковий керівник: к.е.н., професор кафедри ІСЕ

Ситник Н.В. 

"12" червня 2024 р.

Рецензія
на кваліфікаційний бакалаврський проєкт
Сороки Дмитра Сергійовича

тема **«Розробка веб-застосунку криптовалютної біржі»**

Актуальність теми кваліфікаційного бакалаврського проєкту і доцільність його розроблення: Тема кваліфікаційного бакалаврського проєкту є досить актуальною та своєчасною. У контексті швидкого розвитку фінансових технологій та зростаючої популярності криптовалют, розробка веб-застосунку криптовалютної біржі є доцільною та перспективною. Така платформа дозволяє користувачам безпечно та ефективно здійснювати операції з криптовалютами, що сприяє розвитку цифрової економіки.

Якість проведеного дослідження: Дослідження характеризується високою якістю. Автор детально проаналізував існуючі рішення на ринку криптовалютних бірж, визначив їх основні переваги та недоліки, а також запропонував власний підхід до вирішення виявлених проблем. У роботі було застосовано сучасні методи аналізу та розробки програмного забезпечення, що забезпечило високу функціональність і надійність розробленої системи.

Позитивні риси кваліфікаційного бакалаврського проєкту: Серед позитивних рис кваліфікаційного бакалаврського проєкту слід відзначити розробку простого і зрозумілого інтерфейсу, що дозволяє інтегрувати гаманці та здійснювати операції з криптовалютами. Проєкт надає можливість вибору криптовалютних пар, введення обмінних сум та автоматичний розрахунок обмінних курсів, що підвищує ефективність обміну. Використання API для пошуку найнижчих цін на інших біржах зменшує витрати під час обміну криптовалют. Додавання вкладки з інформацією про криптовалюти забезпечує користувачів актуальними даними для прийняття обґрунтованих рішень.

Зауваження: Основні зауваження до проєкту стосуються відсутності інформації щодо підтримки мов інтерфейсу та обмеженості кількості підтримуваних криптовалют в мережі BSC, що може обмежити аудиторію користувачів.

Практична значимість висновків і рекомендацій: Розроблена система здатна значно спростити процеси обміну криптовалют, підвищити їх ефективність та зменшити витрати, що є важливим внеском у розвиток фінансових технологій. Запропоновані методи та алгоритми можуть бути використані для подальшого вдосконалення подібних платформ, а також для розробки нових фінансових інструментів.

Загальний висновок і оцінка роботи. Незважаючи на вказані недоліки, кваліфікаційний дипломний проєкт на тему «Розробка web-застосунку криптовалютної біржі» у цілому відповідає встановленим вимогам та рекомендується до захисту з позитивною оцінкою, а здобувачеві Сорокі Дмитру Сергійовичу може бути присвоєний освітньо-кваліфікаційний рівень «бакалавр» галузі знань 12. Інформаційні технології за спеціальністю 122. Комп'ютерні науки.

Місце роботи та посада рецензента: **Погорєловська І.Д.,** к.е.н., доцент,
доцент кафедри комп'ютерних та інформаційних технологій і систем факультету фінансів та цифрових технологій
Державного податкового університету

«___» _____ 2024

АНОТАЦІЯ

кваліфікаційного бакалаврського проєкту студента 4 курсу Навчально-наукового інституту «Інститут інформаційних технологій в економіці» **Сороки Дмитра Сергійовича**, виконаної на тему: «Розробка веб-застосунку криптовалютної біржі» Київ: кафедра інформаційних систем в економіці, 2024 р.

Кваліфікаційний бакалаврський проєкт присвячений актуальній проблемі створення автоматизованої системи для обміну криптовалют, що дозволяє здійснювати обмін за найнижчими цінами на ринку, надає актуальну інформацію про криптовалюти, підтримує підключення гаманця MetaMask та забезпечує збереження історії транзакцій у базі даних MongoDB.

Проєкт складається з трьох розділів, логічно пов'язаних між собою.

У першому розділі наведено характеристику предметної галузі й об’єкта дослідження, здійснено аналіз існуючих рішень, визначено їх недоліки та обґрунтовано необхідність розробки нового підходу.

Другий розділ присвячений проєктуванню системи, зокрема обґрунтуванню методу проєктування, розробці архітектури системи та постановці завдань. Тут детально розглянуто вибір технологій, розроблено алгоритм розв'язання задач та виконано інтеграцію з API сервісів 1inch та CoinGecko, а також забезпечено підтримку роботи з гаманцем MetaMask і мережею Binance Smart Chain.

Третій розділ містить конструктивні аспекти реалізації проєкту. Розглянуто інформаційне, технічне, програмне та організаційне забезпечення системи. Описано структуру інформаційного забезпечення, форми вхідних та вихідних документів, обґрунтовано комплекс технічних засобів та програмного забезпечення. Наведено структури інформаційних масивів для забезпечення функціональності системи.

Висновки містять рекомендації щодо доцільності розробки та впровадження автоматизованої системи обміну криптовалют, а також пропозиції щодо подальшого вдосконалення системи.

РЕФЕРАТ

Кваліфікаційний бакалаврський проєкт містить 56 сторінок, 8 таблиць, 18 рисунків, список літератури з 13 найменувань, 4 додатків.

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ КРИПТОВАЛЮТНОЇ БІРЖІ

Перелік ключових слів: обмін криптовалют, веб-застосунок, API 1inch, API CoinGecko, MetaMask, MongoDB, Binance Smart Chain.

Предметом дослідження є автоматизована система обміну криптовалют, яка забезпечує інтеграцію з різними сервісами та платформами для проведення транзакцій.

Об’єктом дослідження виступають інформаційні процеси, які характеризують обмін криптовалют.

Мета випускного бакалаврського проєкту полягає в розробці та впровадженні системи обміну криптовалют, яка забезпечувала б обмін криптовалют за найнижчими цінами на ринку, відображення актуальної інформації про криптовалюти, підключення гаманця MetaMask та збереження історії транзакцій у базі даних MongoDB.

Завданнями кваліфікаційного бакалаврського проєкту є:

- аналіз існуючих рішень та визначення їх недоліків;
- розробка архітектури системи;
- вибір та інтеграція необхідних технологій та сервісів;
- реалізація системи та її тестування;
- оцінка ефективності та надійності роботи системи.

Апаратні та програмні засоби, що використовувались при проєктуванні: мова програмування JavaScript, фреймворк Node.js, бази даних MongoDB, API сервісів 1inch та CoinGecko, мережа Binance Smart Chain, криптовалютний гаманець MetaMask.

Результати, досягнуті в процесі роботи, включають розробку веб-застосунку, який забезпечує автоматизований обмін криптовалют з використанням найнижчих цін на ринку, відображення актуальної інформації про криптовалюти, підключення гаманця MetaMask для безпечного

проведення транзакцій, та збереження історії транзакцій у базі даних MongoDB. Новизна проєкту полягає у використанні інтегрованих API та смарт-контрактів для забезпечення високої ефективності та надійності системи. Ступінь впровадження включає тестування системи у реальних умовах, що підтвердило її працездатність та зручність використання.

Одержані результати можуть бути використані для подальшого розвитку криптовалютних бірж та сервісів обміну криптовалют, що забезпечить підвищення ефективності та безпеки транзакцій. Результати проєкту можуть бути впроваджені у комерційні рішення, що працюють у сфері фінансових технологій, зокрема для забезпечення швидких та безпечних обмінів криптовалют у мережі Binance Smart Chain.

Рік виконання випускного бакалаврського проєкту: 2024.

Рік захисту випускного бакалаврського проєкту: 2024.

ЗМІСТ

ВСТУП

Для виконання кваліфікаційного бакалаврського проєкту було обрано тему «Розробка веб-застосунку криптовалютної біржі».

Обрана тема дослідження є вкрай актуальною у контексті сучасних тенденцій розвитку фінансових технологій та цифрової економіки. Розробка автоматизованої системи для обміну криптовалют представляє значний науковий і практичний інтерес, оскільки сучасний ринок криптовалют демонструє стрімке зростання і вимагає нових підходів до забезпечення безпеки, ефективності та зручності проведення транзакцій. У світлі цього, дослідження проблематики обміну криптовалют, а також розробка програмних рішень для їх реалізації, має значну соціальну та економічну значущість.

Актуальність даного дослідження визначається кількома факторами. По-перше, ринок криптовалют постійно змінюється, з'являються нові валюти, змінюються курси, що створює потребу в ефективних засобах для проведення обмінних операцій. По-друге, існує необхідність у забезпеченні високого рівня безпеки транзакцій, що включає захист від шахрайства та зловживань. По-третє, важливим є питання зручності користування системами обміну для широкого кола користувачів, що включає інтуїтивно зрозумілий інтерфейс і швидкість обробки запитів.

Розробка системи, яка б забезпечувала обмін криптовалют за найнижчими цінами на ринку за допомогою API 1inch [7], відображення актуальної інформації про криптовалюти за допомогою API CoinGecko [6], підключення гаманця MetaMask та збереження історії транзакцій у базі даних MongoDB [8], є комплексним завданням, яке охоплює кілька важливих аспектів. Це включає не тільки технічну реалізацію, але й питання організації інформаційного забезпечення, вибору технологій, забезпечення надійності та ефективності роботи системи.

Огляд літератури показує, що на сьогоднішній день існує значна кількість наукових та практичних досліджень, присвячених різним аспектам функціонування ринку криптовалют. Дослідження охоплюють питання

безпеки [1, 2], алгоритмів обміну [3], використання смарт-контрактів [4], а також технічні аспекти реалізації криптовалютних платформ. Проте, незважаючи на це, тема створення інтегрованої автоматизованої системи обміну криптовалют, яка б об'єднувала найкращі практики та новітні технології, залишається недостатньо висвітленою, що і обумовлює її актуальність.

Об'єктом дослідження є процеси обміну криптовалют, а предметом дослідження є автоматизована система обміну криптовалют, яка забезпечує інтеграцію з різними сервісами та платформами для проведення транзакцій. Метою дослідження є розробка та впровадження такої системи, яка б відповідала вимогам сучасного ринку криптовалют, забезпечувала високу швидкість і надійність транзакцій, а також була зручною для користувачів.

Для досягнення поставленої мети були визначені наступні завдання: аналіз існуючих рішень та визначення їх недоліків, розробка архітектури системи, вибір та інтеграція необхідних технологій та сервісів, реалізація системи та її тестування, оцінка ефективності та надійності роботи системи. Для вирішення цих завдань були використані різноманітні методи дослідження, включаючи математичний аналіз, структурно-логічний підхід та вербально-описовий метод.

Теоретична значущість даної роботи полягає у розробці нових підходів до організації обміну криптовалют, що можуть бути використані у подальших наукових дослідженнях. Практична значущість полягає у створенні системи, яка може бути використана як основа для комерційних рішень у сфері фінансових технологій. Структура роботи відображає логічну послідовність етапів розробки, починаючи від аналізу проблеми до реалізації та тестування системи.

РОЗДІЛ 1 ХАРАКТЕРИСТИКА ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика предметної галузі та об’єкта дослідження

Предметна галузь даного дослідження охоплює розробку інформаційної системи криптовалютної біржі, зокрема децентралізованої біржі (DEX). Криптовалютна біржа - це платформа, яка дозволяє користувачам обмінювати одну криптовалюту на іншу або на фіатні гроші.

Централізовані та децентралізовані валютні біржі

Централізовані біржі (CEX) є найпоширенішим типом криптовалютних бірж. Вони працюють аналогічно традиційним біржам, виконуючи роль посередника між покупцем і продавцем. Центральна організація, яка керує біржею, надає користувачам можливість зберігати свої активи на рахунках біржі, обробляє транзакції та встановлює правила торгівлі. Приклади централізованих бірж включають Binance, Coinbase та Kraken. Основними перевагами CEX є висока ліквідність, швидкість транзакцій та зручність користування. Однак вони також мають значні недоліки, такі як ризик хакерських атак та втрати коштів, а також обмежений контроль користувачів над своїми активами.

Децентралізовані біржі (DEX), на відміну від централізованих, не мають центрального контролюючого органу. Вони працюють на основі блокчейну та смарт-контрактів, що забезпечує більш високий рівень безпеки та приватності. DEX дозволяють користувачам зберігати свої активи на власних гаманцях, що зменшує ризик втрати коштів через хакерські атаки. Приклади децентралізованих бірж включають Uniswap, SushiSwap та PancakeSwap. Основними перевагами DEX є підвищена безпека, анонімність та децентралізований контроль над активами. Однак вони часто мають нижчу ліквідність і вищі транзакційні витрати порівняно з CEX.

Види гаманців

Для користування криптовалютними біржами необхідно мати криптовалютний гаманець, який забезпечує зберігання та управління

криптовалютами. Гаманці поділяються на два основних види: гарячі (онлайн) та холодні (офлайн).

Гарячі гаманці є підключеними до Інтернету та надають зручний доступ до криптовалют. Вони можуть бути у вигляді мобільних додатків, веб-сайтів або програм для настільних комп'ютерів. Основною перевагою гарячих гаманців є їхня зручність і швидкість доступу до активів. Проте, вони більш вразливі до хакерських атак і можуть бути небезпечними для зберігання великих сум.

Холодні гаманці є автономними пристроями або програмами, що зберігають криптовалюту офлайн, забезпечуючи високий рівень безпеки. Вони включають апаратні гаманці, паперові гаманці та спеціальні пристрої для зберігання криптовалют. Основною перевагою холодних гаманців є їхня безпека, оскільки вони не підключені до Інтернету та менш вразливі до хакерських атак. Однак, вони можуть бути менш зручними для щоденного використання через складність доступу до активів.

Роль і місце криптовалютної біржі в економіці

Криптовалютні біржі відіграють важливу роль у сучасній економіці, забезпечуючи ліквідність ринку криптовалют та сприяючи їхньому розвитку. Вони дозволяють інвесторам та трейдерам обмінювати криптовалюти, залучати нові інвестиції та підтримувати стабільність ринку. Крім того, криптовалютні біржі сприяють поширенню нових технологій та інновацій у фінансовому секторі, включаючи блокчейн та смарт-контракти.

Особливості предметної галузі

Особливості предметної галузі криптовалютних бірж мають значний вплив на організацію інформаційної системи та технологій розв'язання задач. Основні з них включають:

– Безпека: Оскільки криптовалюти є цифровими активами, вони піддаються ризику хакерських атак та шахрайства. Важливо забезпечити високий рівень захисту даних та транзакцій.

– Ліквідність: Ліквідність є ключовим фактором успіху біржі. Інформаційна система повинна забезпечувати швидкий і ефективний пошук найкращих цін на ринку для здійснення обмінних операцій.

– Прозорість та анонімність: Користувачі криптовалютних бірж часто очікують високого рівня прозорості операцій та анонімності. Важливо забезпечити баланс між цими вимогами та дотриманням регуляторних норм.

– Інтеграція з іншими біржами: Для забезпечення високої ліквідності та конкурентних цін важливо інтегрувати біржу з іншими платформами за допомогою API.

Ці особливості визначають основні вимоги до розробки інформаційної системи криптовалютної біржі, яка повинна бути безпечною, ефективною та зручною для користувачів.

Об’єкт дослідження

Об'єктом дослідження є процес організації та функціонування децентралізованої криптовалютної біржі (DEX). Зокрема, дослідження фокусується на інтеграції криптовалютних гаманців, виборі торгових пар, визначенні курсів обміну та здійсненні транзакцій з мінімальними витратами, а також забезпеченні інформаційної підтримки користувачів про доступні криптовалюти.

Складовими об’єкту дослідження виступають:

– Інтерфейс користувача: Простота і зручність підключення гаманців, вибір криптовалютних пар, введення суми обміну та здійснення транзакцій.

– Інтеграція з зовнішніми біржами: Механізм пошуку найкращих цін на інших біржах для здійснення обмінів при низькій ліквідності власної платформи.

– Безпека транзакцій: Забезпечення безпеки даних користувачів і захист від хакерських атак.

– Інформаційна підтримка: Надання користувачам актуальної інформації про криптовалюти, курси обміну та аналітичні дані.

Дослідження має на меті розробку та вдосконалення цих компонентів для забезпечення ефективної роботи децентралізованої криптовалютної біржі, що відповідає сучасним вимогам ринку та потребам користувачів.

1.2 Аналіз літературних джерел та практичного досвіду використання ІС і технологій в предметній галузі

Для порівняльної характеристики обрані чотири популярні криптовалютні біржі: Binance, Coinbase, Uniswap та SushiSwap. Кожна з цих бірж має свої особливості та функціонал, спрямовані на задоволення потреб трейдерів та інвесторів у криптовалютному просторі.

Загальна характеристика Binance [9]

Binance є однією з найбільших та найпопулярніших криптовалютних бірж у світі. Розробником та власником системи є компанія Binance. Платформа має наступні ключові характеристики та можливості:

– Реєстрація та безпека: Легкий процес реєстрації з можливістю верифікації для підвищення рівня безпеки. Двоетапна аутентифікація (2FA) та інші заходи безпеки для захисту облікового запису.

– Торгівельна платформа: Широкий вибір торгових пар для криптовалют. Різноманітні типи ордерів, включаючи ринкові та лімітні.

– Комісії: Низькі торгові комісії, особливо для користувачів, які використовують власний токен BNB (Binance Coin).

– Ліквідність: Висока ліквідність завдяки великій кількості торгових пар та активності користувачів.

– API та Розробницькі інструменти: Надає потужний API для автоматизованої торгівлі та інтеграції сторонніх додатків. Розширений функціонал для торговців та розробників.

– Страхування активів: Фонд Страхування Фонду Безпеки Binance (SAFU) забезпечує страхування користувачів у разі втрат через несподівані події.

– Зняття коштів: Зручні методи зняття коштів, включаючи криптовалютні та банківські перекази.

– Освіта та Аналітика: Платформа надає користувачам інструменти для освіти та аналітики ринку.

– Стійке оновлення: Активне впровадження нових функцій та покращень для задоволення потреб ринку.

Загальна характеристика Coinbase [10]

Coinbase - одна з найпопулярніших і найбільш відомих криптовалютних бірж та платіжних платформ, яка забезпечує простий спосіб купівлі, продажу та управління різними криптовалютами. Ось ключові характеристики та можливості платформи:

– Платіжні можливості: Дозволяє користувачам купувати та продавати популярні криптовалюти, такі як Bitcoin, Ethereum, Litecoin та інші. Підтримує зручні методи оплати, такі як кредитні картки, дебетові картки та банківські перекази.

– Безпека: Високий рівень безпеки, включаючи двофакторну аутентифікацію для захисту облікових записів користувачів. Засоби збереження криптовалют в холодних гаманцях для мінімізації ризиків хакерських атак.

– Простий інтерфейс: Інтуїтивно зрозумілий та дружелюбний інтерфейс для новачків та досвідчених користувачів.

– Мобільний додаток: Мобільний додаток для доступу до облікового запису та управління криптовалютами з будь-якого місця.

– Карткові продукти: Coinbase Card - дебетова картка, яка дозволяє витрачати криптовалюту в місцях, де приймають картки Visa. Інтеграція з Apple Pay та Google Pay.

– Широкий вибір криптовалют: Підтримка ряду криптовалют, включаючи ті, які наразі не є основними.

– Освітні ресурси: Надає освітні матеріали та ресурси для новачків у світі криптовалют та блокчейн технологій.

– Зручність та швидкість: Швидкість виконання операцій та доступ до криптовалют на ринку в реальному часі.

– Бонусні програми: Програми лояльності та бонусні акції для користувачів.

– Інтеграція з іншими сервісами: Забезпечує можливість інтеграції з різними фінансовими та бухгалтерськими програмами.

– Опції збереження: Варіанти збереження криптовалют, включаючи гаманці для користувачів, які бажають утримувати свої активи поза біржею.

– Легальність та регулювання: Співпраця з регуляторами та дотримання вимог щодо забезпечення безпеки та легальності операцій.

Загальна характеристика Uniswap [11]

Uniswap є однією з найвідоміших децентралізованих криптовалютних бірж (DEX), яка працює на базі блокчейну Ethereum. Вона надає можливість обміну токенів без участі центрального посередника, використовуючи смарт-контракти. Основні характеристики та можливості платформи Uniswap включають:

– Децентралізація та безпека: Uniswap працює без центрального контролюючого органу, що зменшує ризик хакерських атак. Користувачі зберігають свої криптовалюти на власних гаманцях, забезпечуючи повний контроль над активами.

– Простота використання: Інтерфейс платформи інтуїтивно зрозумілий і дозволяє швидко здійснювати обмін токенів.

– Автоматизований маркет-мейкер (AMM): Uniswap використовує AMM, де користувачі забезпечують ліквідність у пулах, що дозволяє автоматично визначати ціни на токени на основі співвідношення між ними.

– Ліквідність: Користувачі можуть надавати ліквідність до пулів та отримувати винагороду у вигляді комісій за транзакції.

– API та інтеграція: Uniswap надає API для інтеграції з іншими додатками та автоматизації торгівлі.

– Освітні ресурси: Платформа надає користувачам навчальні матеріали про використання децентралізованих фінансів (DeFi) та смарт-контрактів.

– Прозорість: Усі транзакції на Uniswap є публічними і можуть бути перевірені на блокчейні Ethereum.

– Гнучкість у виборі токенів: Користувачі можуть обмінювати широкий спектр токенів ERC-20 без потреби в листингу на централізованій біржі.

– Інноваційність: Uniswap активно розвиває свою платформу, впроваджуючи нові функції та технології, такі як Uniswap V3, який забезпечує більш ефективне управління ліквідністю.

Загальна характеристика SushiSwap [12]

SushiSwap є популярною децентралізованою криптовалютною біржею, яка виникла як форк Uniswap і з часом розвинула власний унікальний набір функцій. Основні характеристики та можливості платформи SushiSwap включають:

– Децентралізація та безпека: SushiSwap працює на базі смарт-контрактів і децентралізованого управління, що дозволяє користувачам зберігати контроль над своїми активами.

– Торгівельна платформа: SushiSwap надає зручний інтерфейс для обміну токенів та надання ліквідності до пулів.

– Автоматизований маркет-мейкер (AMM): Платформа використовує AMM, де користувачі надають ліквідність у пулах, а ціни на токени визначаються автоматично на основі співвідношення між активами в пулах.

– Ліквідність та винагороди: Користувачі можуть надавати ліквідність до пулів та отримувати винагороду у вигляді токенів SUSHI, які можна стейкати для отримання додаткових прибутків.

– Мульти-ланцюговість: SushiSwap підтримує кілька блокчейнів, включаючи Ethereum, Binance Smart Chain, Polygon, і Avalanche, що забезпечує більш широкий доступ до ліквідності та токенів.

– API та інтеграція: SushiSwap надає API для інтеграції з іншими додатками та автоматизації торгівлі.

– Гнучкість у виборі токенів: SushiSwap підтримує широкий спектр токенів ERC-20 та інших стандартів, що дозволяє користувачам легко обмінювати різні активи.

– Прозорість та децентралізоване управління: SushiSwap використовує модель децентралізованого управління, де користувачі можуть голосувати за пропозиції та зміни на платформі, що забезпечує прозорість і врахування думок спільноти.

Порівняння систем наведено в таблиці 1.1.

Таблиця 1.1

ПОРІВНЯННЯ СИСТЕМ

№	*Можливості*	*Binance*	*Coinbase*	*Uniswap*	*SushiSwap*
1	Тип біржі	Централізована (CEX)	Централізована (CEX)	Децентралізована (DEX)	Децентралізована (DEX)
2	Реєстрація та безпека	Так	Так	Не потрібна	Не потрібна
3	Торгова платформа	Широкий вибір торгових пар	Простий інтерфейс	Різноманітні торгові пари	Різноманітні торгові пари
4	Комісії	Низькі	Високі	Міняються залежно від пулу ліквідності	Міняються залежно від пулу ліквідності
5	Ліквідність	Висока	Висока	Міняється залежно від пулу ліквідності	Міняється залежно від пулу ліквідності
6	API та розробницькі інструменти	Потужний API	Немає	Доступні	Доступні
7	Страхування активів	SAFU (Secure Asset Fund for Users)	Немає	Немає	Немає
8	Зняття коштів	Зручні методи	Зручні методи	Непрямі	Непрямі
9	Освіта та аналітика	Інструменти та ресурси	Освітні матеріали	Немає	Немає
10	Стійке оновлення	Активні оновлення	Регулярні оновлення	Постійні оновлення протоколу	Постійні оновлення протоколу
11	Платіжні можливості	Фіатні валюти	Фіатні валюти	Криптовалюти	Криптовалюти
12	Мобільний додаток	Так	Так	Немає	Немає
13	Карткові продукти	Binance Card	Coinbase Card	Немає	Немає
14	Широкий вибір криптовалют	Так	Так	Більше 10 000	Більше 10 000
15	Освітні ресурси	Так	Так	Так	Немає
16	Зручність та швидкість	Висока	Висока	Міняється залежно від пулу ліквідності	Міняється залежно від пулу ліквідності
17	Бонусні програми	Так	Так	Немає	Немає
18	Інтеграція з іншими сервісами	Так	Так	Обмежена	Обмежена
19	Опції збереження	Гаманці Binance	Гаманці Coinbase	Гаманці Metamask	Гаманці Metamask
20	Легальність та регулювання	Відповідність нормативам	Відповідність нормативам	Децентралізована	Децентралізована

Висновки за результатами аналізу програмних систем

Після здійснення аналізу програмних систем можна зробити наступні висновки:

1. Binance вирізняється широким функціоналом та високою ліквідністю. Надає потужний API для автоматизованої торгівлі, а також визначається відмінною безпекою та страхуванням активів через SAFU.

2. Coinbase вражає своєю орієнтацією на простоту використання, що робить її популярною серед новачків. Має великий вибір криптовалют та різноманітні методи оплати, а також надає засоби збереження та високий рівень безпеки.

3. Uniswap забезпечує децентралізований обмін токенів з використанням автоматизованих маркет-мейкерів (AMM), що дозволяє користувачам надавати ліквідність і отримувати винагороду. Вона має високу гнучкість у виборі токенів і прозорість завдяки роботі на блокчейні Ethereum.

4. SushiSwap є децентралізованою платформою з додатковими фінансовими інструментами, такими як лендінг та фармінг, що дозволяє користувачам отримувати прибуток від своїх активів. Підтримує кілька блокчейнів, що забезпечує широкий доступ до ліквідності.

Недоліки існуючих систем та напрями їх удосконалення

1. Binance

– Недоліки: Велика кількість функцій може бути складною для новачків; обмежені опції для децентралізованих фінансів (DeFi).

– Напрями удосконалення: Зменшення складнощів інтерфейсу для новачків та розвиток DeFi-інтеграції для розширення можливостей.

2. Coinbase

– Недоліки: Високі комісії, особливо для невеликих транзакцій; обмежена кількість доступних криптовалют.

– Напрями удосконалення: Зменшення комісій для зростання конкурентоспроможності та розширення списку підтримуваних криптовалют.

3. Uniswap

– Недоліки: Залежність ліквідності від пулів; високі комісії на базі Ethereum через проблеми масштабованості блокчейну.

– Напрями удосконалення: Оптимізація механізмів надання ліквідності та зниження комісій шляхом переходу на ефективніші блокчейн-рішення або шарові рішення другого рівня (Layer 2).

4. SushiSwap

– Недоліки: Непряма процедура зняття коштів; залежність від змін у ліквідності пулів.

– Напрями удосконалення: Підвищення зручності зняття коштів через інтеграцію з більшою кількістю блокчейнів та покращення механізмів управління ліквідністю для стабільності користувацького досвіду.

РОЗДІЛ 2 РОЗРОБКА ВИМОГ І МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

2.1 Аналіз і специфікація вимог до інформаційної підсистеми

Бізнес-вимоги до системи

Бізнес-вимоги визначають цілі та завдання, які система повинна допомогти досягти бізнесу. Вони описують бажані результати з точки зору бізнесу, а не те, як система повинна їх досягти.

Стейкхолдерами криптовалютної біржі є:

Користувачі криптовалютної біржі: Основні користувачі веб-застосунку, які будуть використовувати його для обміну криптовалют.

Інвестори та акціонери: Особи або організації, які вклали фінансові ресурси в розвиток біржі та очікують прибутку від її успішної роботи.

Цілі стейкхолдерів (бізнес-цілі):

Користувачі криптовалютної біржі: Забезпечення швидкого та безпечного доступу до ринку криптовалют та забезпечення оптимальних умов для успішної торгівлі.

Інвестори та акціонери: Забезпечення високого рівня прибутковості їх інвестицій та збільшення вартості акцій.

Бізнес-вимоги до ІС:

Користувачі криптовалютної біржі:

1. Забезпечення швидкого доступу до платформи за допомогою ефективної архітектури та оптимізованої швидкості завантаження:
   - Платформа повинна бути високопродуктивною і забезпечувати мінімальні затримки при завантаженні сторінок та виконанні торгових операцій.
   - Використання сучасних технологій для забезпечення швидкої обробки запитів та масштабованості системи.
2. Розробка інтуїтивно зрозумілого інтерфейсу з зручними функціями для торгівлі та управління аккаунтом:

– Інтерфейс користувача повинен бути зрозумілим та легким у використанні, забезпечуючи доступ до всіх необхідних функцій торгівлі, перегляду балансу, історії транзакцій та інших важливих даних.

3. Інтеграція з популярними криптовалютними гаманцями для підключення користувачів до платформи:

– Платформа повинна підтримувати інтеграцію з популярними криптовалютними гаманцями (наприклад, MetaMask, Trust Wallet), що дозволить користувачам легко підключати свої гаманці для здійснення операцій.

– Зручний та безпечний процес підключення гаманця, без необхідності реєстрації на платформі.

4. Забезпечення прозорості та достовірності інформації про ринок криптовалют:

– Надання користувачам актуальної інформації про курси криптовалют, обсяги торгів, тренди та інші важливі аналітичні дані.

– Використання надійних джерел даних та регулярне оновлення інформації для забезпечення достовірності.

Інвестори та акціонери:

1. Максимізація прибутку та рентабельності:

– Розробка ефективної фінансової стратегії для збільшення прибутків та забезпечення стабільного росту компанії.

– Використання аналітичних інструментів для моніторингу фінансових показників та виявлення можливостей для підвищення рентабельності.

2. Збільшення ринкової частки та залучення нових користувачів:

– Впровадження маркетингових кампаній та акцій для залучення нових користувачів та утримання існуючих.

– Розширення географічної присутності на ринках різних країн.

Нефункціональні вимоги до системи

Нефункціональні вимоги описують, як система має працювати. Їх можна поділити на такі категорії:

– Вимоги до продуктивності: Систему має бути здатною обробляти великий обсяг транзакцій з низькою затримкою.

– Вимоги до надійності: Систему має бути стійкою до збоїв та мати високу доступність.

– Вимоги до безпеки: Систему має бути захищеною від несанкціонованого доступу та кібератак.

– Вимоги до масштабованості: Систему має бути можливо масштабувати для обслуговування зростаючої кількості користувачів та транзакцій.

– Вимоги до зручності використання: Система має бути інтуїтивно зрозумілою та простою у використанні.

Функціональні вимоги до системи

Функціональні вимоги до системи представляють собою опис того, що система повинна робити, які функції вона повинна виконувати для задоволення потреб користувачів та досягнення поставлених цілей. Вони є ключовою складовою частиною аналізу вимог і визначають обсяг функціональності, яку має мати система.

З урахуванням особливостей криптовалютного ринку та потреб користувачів, наведемо наступний перелік функціональних вимог до системи:

Управління крипто-гаманцем: Система повинна надавати можливість користувачам підключати свій крипто-гаманець для здійснення операцій з криптовалютами.

Проведення обмінних операцій: Система повинна дозволяти користувачам проводити обмін криптовалют, здійснюючи пошук найвигідніших пропозицій на інших біржах через API.

Калькуляція обмінних курсів: Система повинна надавати користувачам можливість ввести кількість криптовалюти, яку вони хочуть обміняти, і

автоматично розрахувати кількість іншої криптовалюти, яку вони отримають за цей обмін.

Виконання обмінних операцій: Система повинна забезпечувати процес підтвердження обмінної операції користувачем і автоматично здійснювати обмін криптовалют.

Відображення історії транзакцій: Система повинна надавати користувачам доступ до історії їхніх транзакцій, включаючи деталі кожної операції, такі як дати, суми та біржі, через які були проведені обміни.

Інформація про криптовалюти: Система повинна надавати користувачам доступ до детальної інформації про різні криптовалюти, включаючи їхні основні характеристики та історію цін.

На рисунку 2.1 зображено діаграму функціональних вимог.

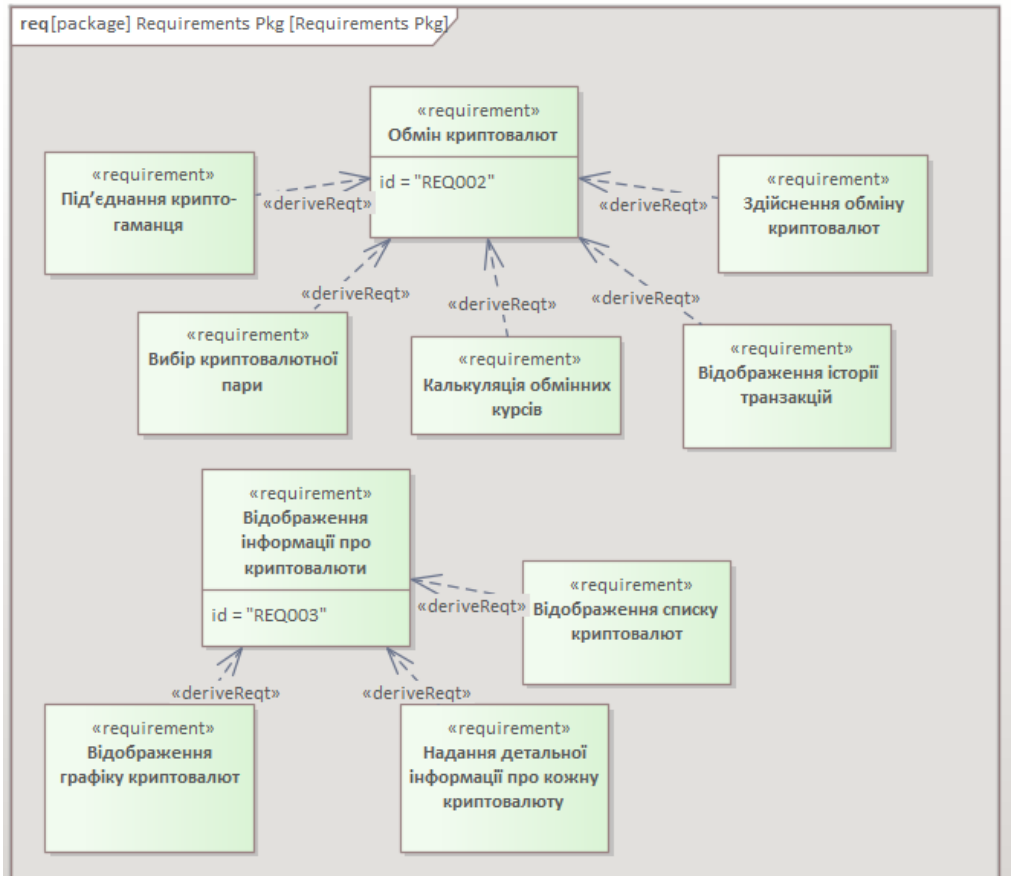


Рисунок 2.1 – Діаграма функціональних вимог

Специфікація вимог є документом, який описує детально всі вимоги до системи. Вона використовується як основа для розробки, тестування та впровадження системи.

На рисунку 2.2 зображено специфікацію функціональних вимог, підготовлену за допомогою менеджера специфікацій (Specification Manager).

Item	Stereoty...	Status	Difficulty	Priority
Вибір криптовалютної пари	requirem...	Proposed	Medium	Medium
Відображення графіку криптовалют	requirem...	Proposed	Medium	Medium
Відображення інформації про криптовалюти	requirem...	Proposed	Medium	Medium
Відображення історії транзакцій	requirem...	Proposed	Medium	Medium
Відображення списку криптовалют	requirem...	Proposed	Medium	Medium
Здійснення обміну криптовалют	requirem...	Proposed	Medium	Medium
Калькуляція обмінних курсів	requirem...	Proposed	Medium	Medium
Надання детальної інформації про кожну криптовалюту	requirem...	Proposed	Medium	Medium
Обмін криптовалют	requirem...	Proposed	Medium	Medium
Під’єднання крипто-гаманця	requirem...	Proposed	Medium	Medium

Рисунок 2.2 – Специфікація функціональних вимог

2.2 Постановка задачі та алгоритм розв’язання задачі

2.2.1 Постановка задачі

Характеристика задачі

Задача "Обмін криптовалют" полягає у створенні автоматизованої системи для проведення операцій з обміну криптовалют через інші біржі та надання користувачам актуальної інформації про ринок криптовалют.

Використання ЕОМ для розв’язання цієї задачі обумовлено необхідністю забезпечення швидкого, безпечного та ефективного доступу до криптовалютного ринку, автоматизації обмінних операцій та оперативного отримання даних про криптовалюти.

При розв'язанні цієї задачі передбачено керування наступними об'єктами: криптовалютні гаманці, криптовалютні платформи, список доступних криптовалют для обміну, API з інформацією про криптовалюти, історія транзакцій.

Вихідна інформація включає підтвердження успішних обмінних операцій, поточні ціни на криптовалюти, історію транзакцій користувачів, аналітичні дані ринку та сповіщення про статус обмінних операцій. Вона використовується для надання користувачам актуальної інформації про ринок криптовалют та забезпечення ефективного управління їхніми криптовалютними активами.

Задача розв’язується в реальному часі. Інформація про ринки криптовалют оновлюється безперервно, а дані про виконані обмінні операції надаються користувачам одразу після їх завершення.

Автоматизоване розв’язання задачі припиняється у разі відсутності з’єднання з API бірж або джерел даних про криптовалюти, технічних проблем з платформою чи у разі припинення підтримки API з боку бірж.

Вихідна інформація

Вихідна інформація є критичним компонентом системи, оскільки вона забезпечує користувачів необхідними даними для прийняття рішень і

виконання дій. Призначення вихідної інформації полягає у своєчасному інформуванні користувачів про результати їх запитів та про стан ринку.

Перелік і опис вихідних повідомлень наведено у табл. 2.1.

Таблиця 2.1

ПЕРЕЛІК І ОПИС ВИХІДНИХ ПОВІДОМЛЕНЬ

Назва вихідного повідомлення	*Ідентифікатор*	*Форма подання і вимоги до неї*	*Періодичність видання*	*Термін видання і допустимий час затримки*	*Користувач інформації*
Повідомлення про успішний обмін	isSuccess	Екранна форма	При успішному виконанні обміну	Одразу після завершення обміну	Система
Кількість другої криптовалюти	tokenTwoAmount	Число	При введені кількості першої криптовалюти	Негайно	Система
Ранг за капіталізацією ринку криптовалюти	market_cap_rank	Число	При оновленні сторінки	Негайно	API
Зображення криптовалюти	image	URL зображення	При оновленні сторінки	Негайно	API
Поточна ціна криптовалюти	current_price	Число	При оновленні сторінки	Негайно	API
Відсоток зміни ціни за останні 24 години	price_change_percentage_24h	Число	При оновленні сторінки	Негайно	API
Капіталізація ринку криптовалют	market_cap	Число	При оновленні сторінки	Негайно	API
Графік зміни цін	chart	Масив	При оновленні сторінки	Негайно	API
Найвища ціна за 24 години	high_24h	Число	При оновленні сторінки	Негайно	API
Найнижча ціна за 24 години	low_24h	Число	При оновленні сторінки	Негайно	API
ID транзакції	transaction_id	String	При виконанні операції обміну	Негайно	Система
Номер транзакції	number	Число	При виконанні операції обміну	Негайно	Система
Номер гаманця	wallet_id	Рядок	При виконанні операції обміну	Негайно	Користувач
З криптовалюти	from_ticker	Рядок	При виконанні операції обміну	Негайно	Користувач
В криптовалюту	to_ticker	Рядок	При виконанні операції обміну	Негайно	Користувач
Віддано	input	Число	При виконанні операції обміну	Негайно	Користувач

Отримано	output	Число	При виконанні операції обміну	Негайно	Система
Статус	status	Рядок	При виконанні операції обміну	Негайно	Система

Вихідна інформація є необхідною для повноцінного взаємодії користувачів із системою, надаючи їм достатньо засобів для контролю та інформування про їхні дії на біржі. Вона також має структурований характер, що допомагає користувачам з легкістю орієнтуватися та здійснювати свої операції.

Вхідна інформація

Вхідними повідомленнями для використання веб-застосунку криптовалютної біржі є:

Запит на обмін криптовалют (fetchDexSwap) використовується для формування замовлення на обмін криптовалют через API зовнішнього сервісу. Користувачі оформляють такі запити через екранну форму на веб-платформі. Це дозволяє здійснювати обмін криптовалют автоматично, забезпечуючи ефективність і зручність процесу.

Запит на під’єднання гаманця (address) спрямований на додавання криптовалютного гаманця користувача до системи. Користувачі ініціюють цей запит, натискаючи на кнопку «Гаманець» у веб-інтерфейсі та заповнюючи відповідну екранну форму. Це дозволяє інтегрувати особисті гаманці користувачів з платформою, задля управління їхніми активами.

Баланс гаманця (balance) представляє собою запит на отримання інформації про поточний баланс гаманця користувача. Цей запит надходить при натисканні на кнопку «Гаманець» у веб-інтерфейсі і дозволяє користувачам оперативно отримувати актуальну інформацію про стан їхніх активів.

Вибір першої криптовалюти (tokenOne) дає можливість користувачам обрати першу криптовалюту для обміну. Користувачі здійснюють цей вибір через екранну форму на веб-платформі. Це забезпечує гнучкість у виборі криптовалютної пари для проведення обмінних операцій.

Вибір другої криптовалюти (tokenTwo) дає можливість користувачам обрати другу криптовалюту для обміну. Користувачі здійснюють цей вибір через екранну форму на веб-платформі, що дозволяє точно налаштувати параметри обміну відповідно до їхніх потреб.

Введення суми обміну першої криптовалюти (tokenOneAmount) призначене для введення користувачами суми криптовалюти для обміну. Користувачі вводять цю суму через екранну форму на веб-платформі, що дозволяє точно визначити обсяг транзакції і забезпечити її коректне виконання.

Введення суми обміну другої криптовалюти (tokenTwoAmount) автоматично розраховується системою на основі введеної суми першої криптовалюти. Це забезпечує користувачів інформацією про очікуваний результат обміну і дозволяє здійснювати обмін з урахуванням поточних ринкових курсів.

Вартість першої криптовалюти в доларах (priceOne) представляє собою запит на отримання поточної вартості першої криптовалюти. Цей запит надходить від API при введенні кількості першої криптовалюти і дозволяє користувачам отримувати актуальну ринкову інформацію.

Вартість другої криптовалюти в доларах (priceTwo) аналогічно представляє собою запит на отримання поточної вартості другої криптовалюти. Цей запит також надходить від API при введенні кількості першої криптовалюти, що дозволяє користувачам отримувати точні дані для прийняття рішень.

Перелік і опис вхідних повідомлень наведено у табл. 2.2.

Таблиця 2.2

ПЕРЕЛІК І ОПИС ВХІДНИХ ПОВІДОМЛЕНЬ

Назва вхідного повідомлення	*Ідентифікатор*	*Форма подання*	*Термін і частота надходження*	*Джерело*
Запит на обмін криптовалют	fetchDexSwap	JSON-запит	При оформленні нового замовлення на обмін криптовалют	Користувач
Адреса гаманця	address	Рядок	При натисканні на кнопку «Гаманець»	Система
Баланс гаманця	balance	Число	При натисканні на кнопку «Гаманець»	Система

Перша криптовалюта	tokenOne	Рядок	При виборі криптовалютної пари для обміну	Користувач
Друга криптовалюта	tokenTwo	Рядок	При виборі криптовалютної пари для обміну	Користувач
Кількість першої криптовалюти	tokenOneAmount	Число	При введені кількості першої криптовалюти	Користувач
Кількість другої криптовалюти	tokenTwoAmount	Число	При введені кількості першої криптовалюти	Система
Вартість першої криптовалюти в доларах	priceOne	Число	При введені кількості першої криптовалюти	API
Вартість другої криптовалюти в доларах	priceTwo	Число	При введені кількості першої криптовалюти	API

Інформаційна модель задачі зображена на (рис.2.3).

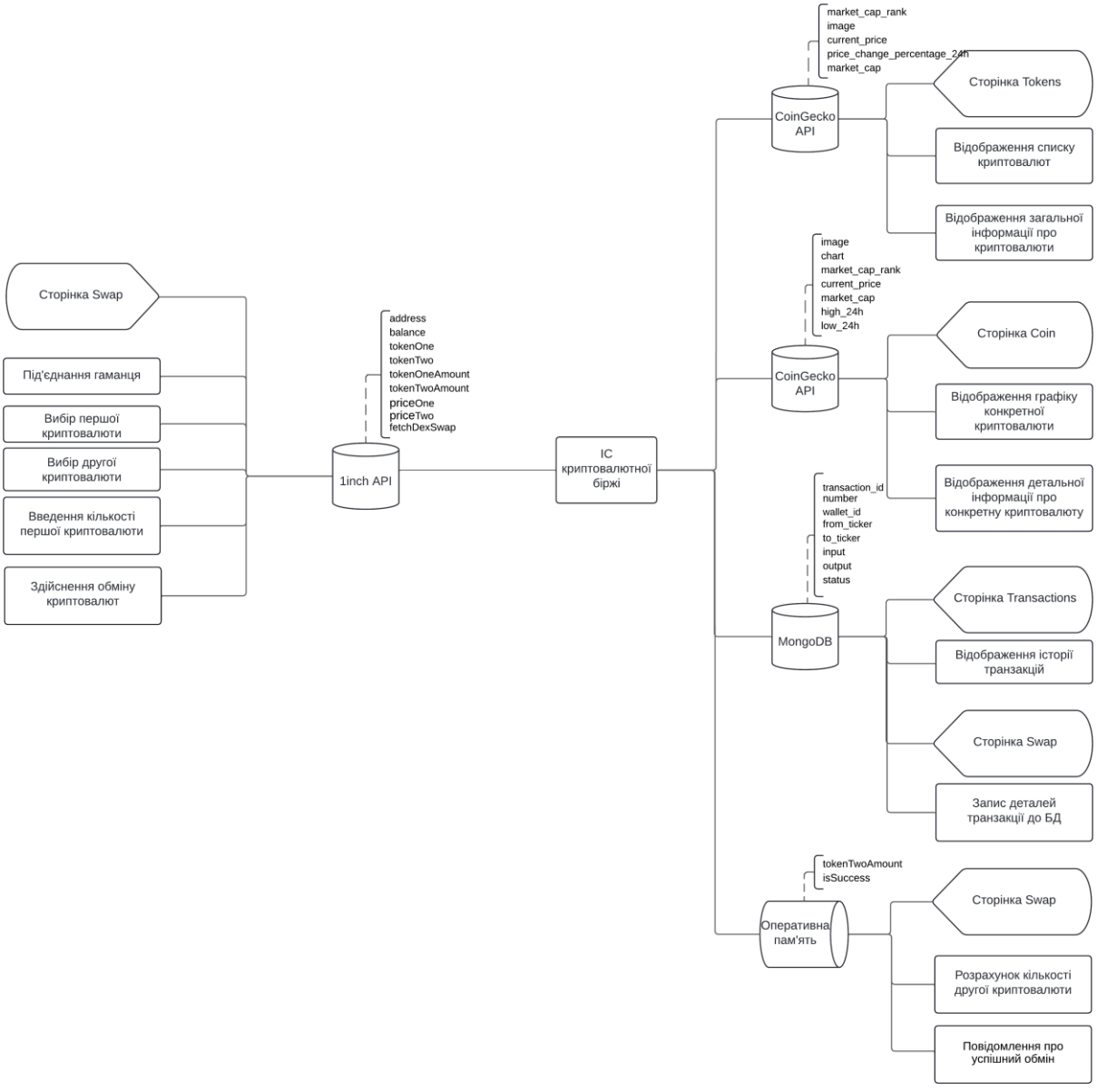


Рисунок 2.3 – Інформаційна модель задачі

2.2.2. Алгоритм розв’язання задачі.

Використовувана інформація

Призначення інформації, яку використовують в процесі розв'язання задачі "Обмін криптовалют", має на меті надати дані, які необхідні для обчислення різних параметрів та виконання операцій з обміну криптовалют. Основними цілями є визначення кількості та вартості криптовалют під час обміну.

Ці дані використовуються для обчислення кількості другої криптовалюти.

Перелік масивів використовуваної інформації наведено таблиці 2. 3.

Таблиця2.3

ПЕРЕЛІК МАСИВІВ ВИКОРИСТОВУВАНОЇ ІНФОРМАЦІЇ

Масив	Ідентифікатор	Максимальна кількість записів
1	2	3
Кількість першої криптовалюти	tokenOneAmount	10000
Вартість першої криптовалюти в доларах	priceOne	10000
Вартість другої криптовалюти в доларах	priceTwo	10000

Результати розв'язання

Результати розв'язання цієї задачі мають на меті надати користувачеві необхідну інформацію про виконання операції обміну криптовалют.

Перелік масивів результатної інформації наведено таблиці 2. 4.

Таблиця 2.4

ПЕРЕЛІК МАСИВІВ РЕЗУЛЬТАТНОЇ ІНФОРМАЦІЇ

Масив	Ідентифікатор	Максимальна кількість записів
1	2	3
Кількість другої криптовалюти	tokenTwoAmount	10000

Математичний опис

1. Визначення кількості криптовалюти 2 за певну кількість криптовалюти 1:

$$Q_2 = (Q_1 / P_{1-USD}) \times P_{2-USD}, \text{ де:}$$

– Q_1 - кількість першої криптовалюти;

– P_{1-USD} - ціна першої криптовалюти в доларах;

– P_{2-USD} - ціна другої криптовалюти в доларах.

Перша криптовалюта (Q_1) – це криптовалюта, яку користувач хоче обміняти. Її вартість виражається в доларах (P_{1-USD}). Друга криптовалюта (Q_2) – це криптовалюта, яку користувач хоче отримати в результаті обміну. Її вартість також виражається в доларах (P_{2-USD}). Процес обміну включає розрахунок кількості другої криптовалюти, яку можна отримати за певну кількість першої криптовалюти, використовуючи їх вартість у доларах.

Алгоритм розв'язання задачі на ЕОМ

Алгоритм розв'язання спрямований на визначення кількості другої криптовалюти за певну кількість першої криптовалюти. Під час розрахунків

використовуються вхідні дані, такі як кількість першої криптовалюти, її вартість в доларах, вартість другої криптовалюти в доларах.

Першим етапом алгоритму є введення користувачем кількості першої криптовалюти (Q_1). Далі алгоритм отримує вартість першої криптовалюти в доларах (P_{1-USD}) та вартість другої криптовалюти в доларах (P_{2-USD}) через API.

Після введення даних алгоритм обчислює кількість другої криптовалюти (Q_2) за допомогою вказаної формули. Кінцевим етапом є виведення результатів, які включають кількість другої криптовалюти.

Алгоритм розв’язання задачі наведено на рисунку 2.4.

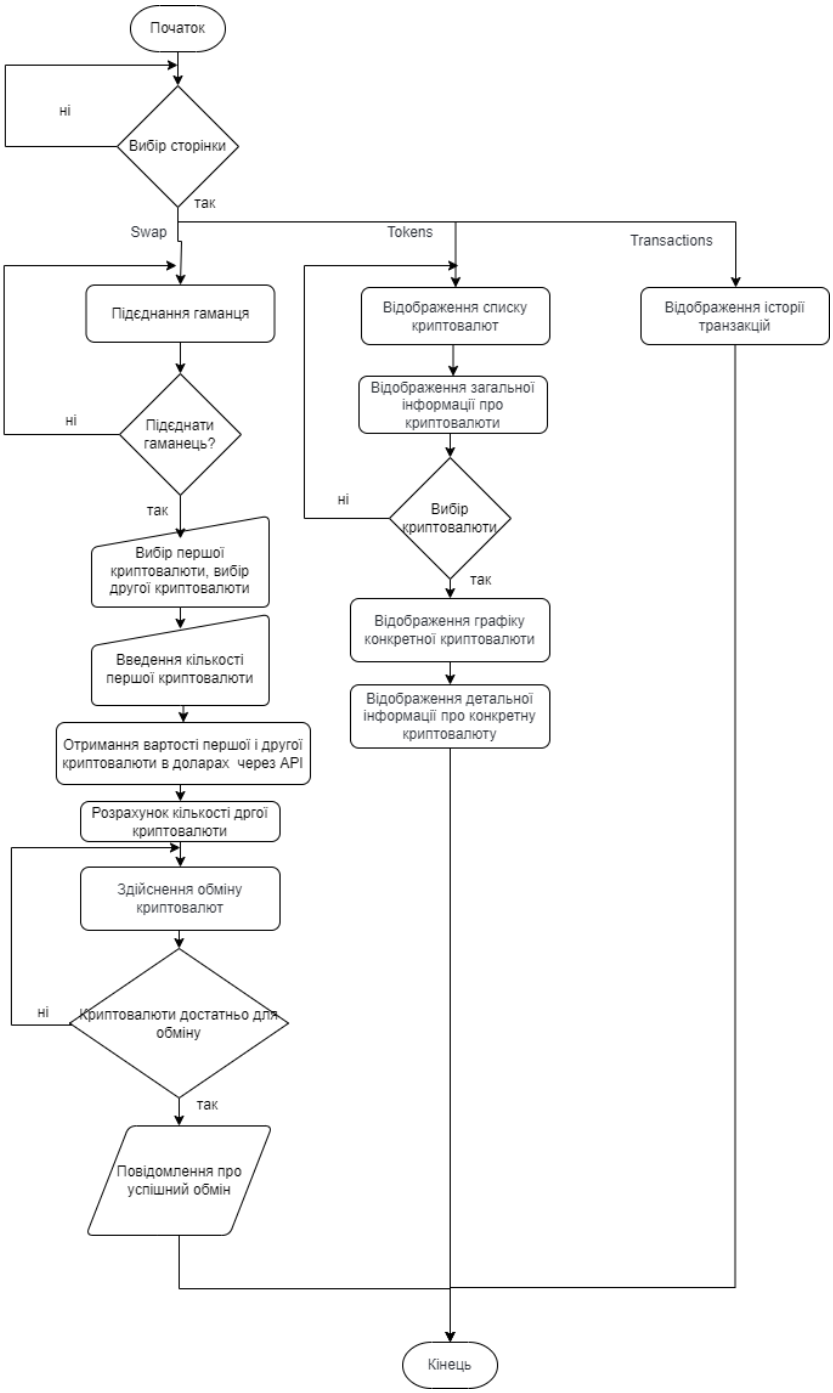


Рисунок 2.4 – Алгоритм розв’язання задачі

2.3 Моделювання інформаційної підсистеми

Для визначення функціоналу та взаємодії складових системи використовується SysML (Systems Modeling Language). Нижче наведені скріншоти діаграм, що описують функціонування інформаційної системи.

Діаграма прецедентів (Use Case Diagram) зображена на рис. 2.5.

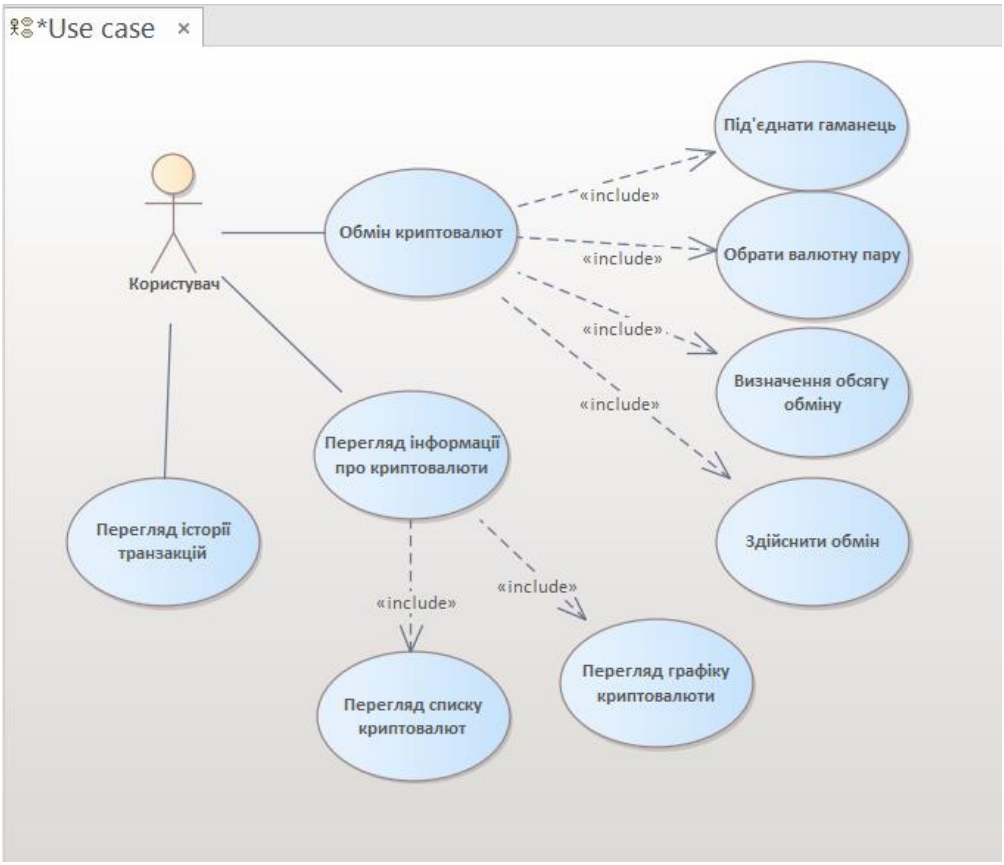


Рисунок 2.5 – Головна сторінка

На діаграмі прецедентів (Use Case Diagram) визначено основні взаємодії користувачів та системи, такі як обмін криптовалют, перегляд інформації про криптовалюти, перегляд історії транзакцій.

На рисунку 2.6 зображено матрицю відповідності вимог і прецедентів.

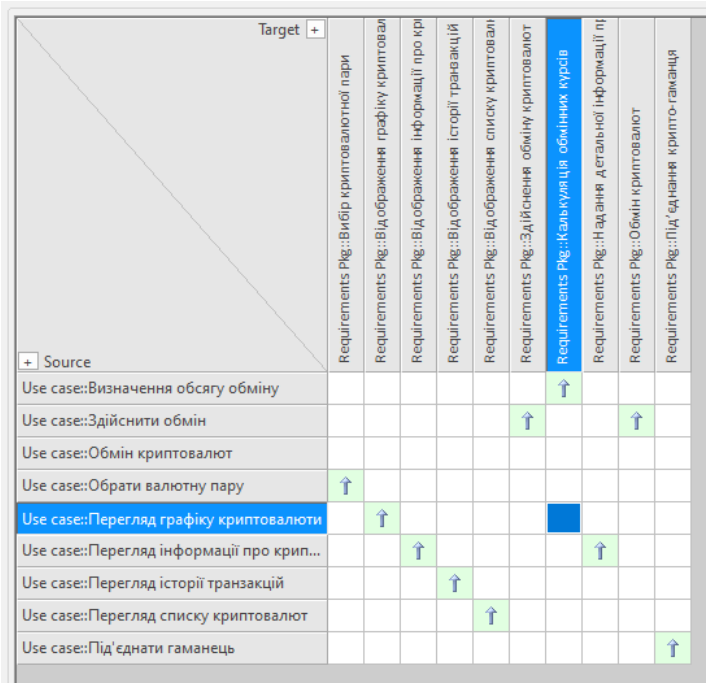


Рисунок 2.6 – Матриця відповідності вимог і прецедентів.

На цих діаграмах послідовності (Sequence Diagram) зображено послідовність дій для кожного прецеденту (Use Case) (рис. 2.7 – 2.9):

1) Обмін криптовалют. Користувач переходить на сторінку обміну, браузер ініціалізує компонент. Компонент отримує дані про ціни на газ, баланс гаманця та ціни на токени. Користувач взаємодіє з параметрами толерантності проскальзування, компонент оновлює їх. Користувач вводить кількість токенів, компонент оновлює кількість та розраховує еквівалентну кількість токенів. Користувач перемикає токени, компонент змінює їх та отримує ціни для переміщених токенів. Користувач відкриває модальне вікно вибору токенів, компонент відкриває його. Користувач вибирає токен, компонент оновлює вибраний токен та отримує ціни для обраного токена. Користувач взаємодіє з кнопкою обміну, компонент виконує обмін токенів та додає транзакцію. Сервер підтверджує додавання транзакції, компонент показує повідомлення про успішний обмін. Браузер показує повідомлення про успішний обмін користувачу.

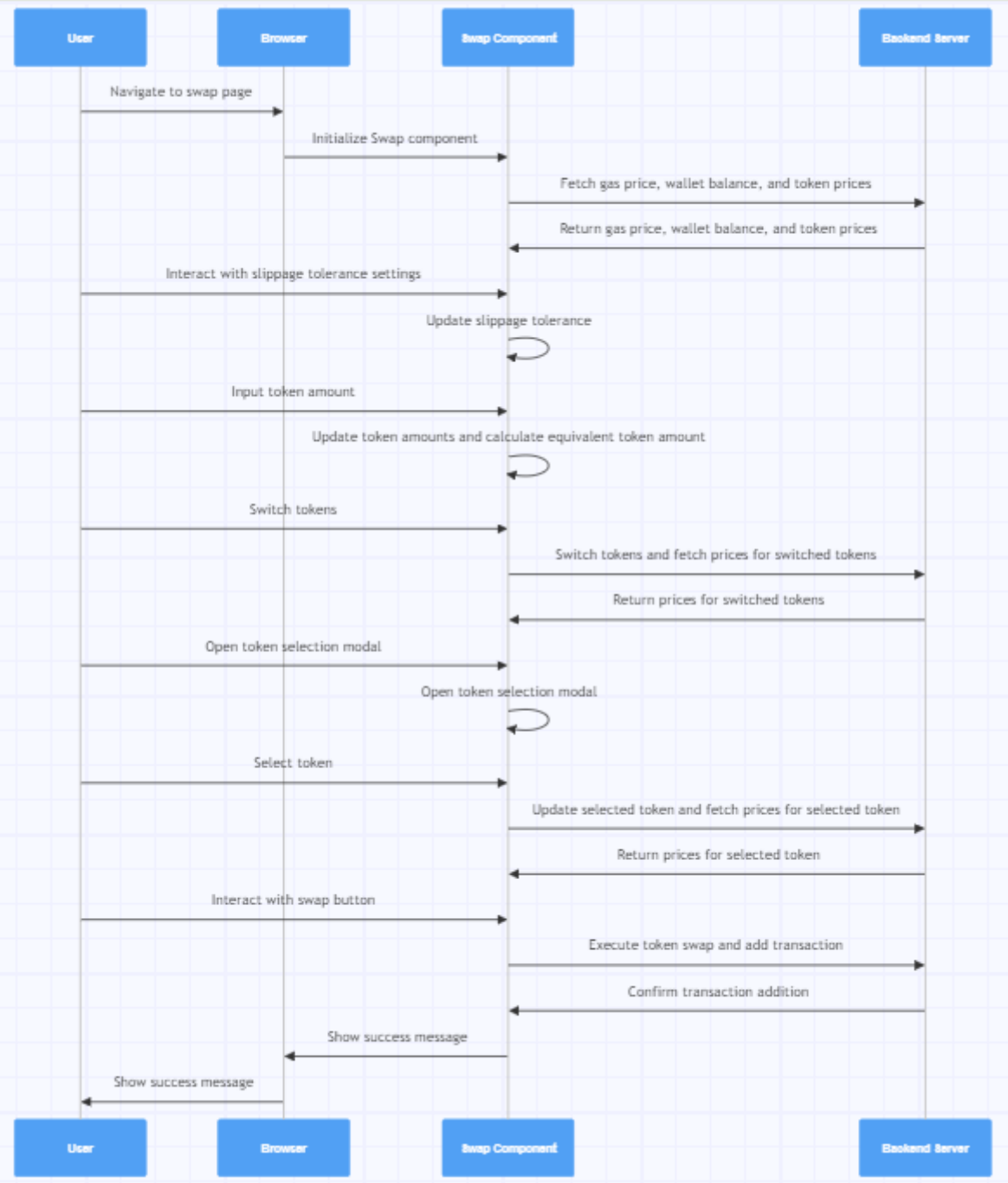


Рисунок 2.7 – Діаграма послідовності (Sequence Diagram) здійснення обміну криптовалют

2) Перегляд історії транзакцій. Користувач переходить на сторінку історії транзакцій, браузер ініціалізує компонент. Компонент відправляє запит на сервер для отримання транзакцій. Сервер запитує базу даних, яка повертає записи транзакцій. Сервер повертає дані про транзакції компоненту. Компонент оновлює стан з отриманими даними та відображає транзакції. Користувач взаємодіє з елементами керування пагінацією, компонент оновлює

поточну сторінку та відображає нові транзакції. Компонент показує оновлені дані транзакцій браузеру, який відображає їх користувачу.

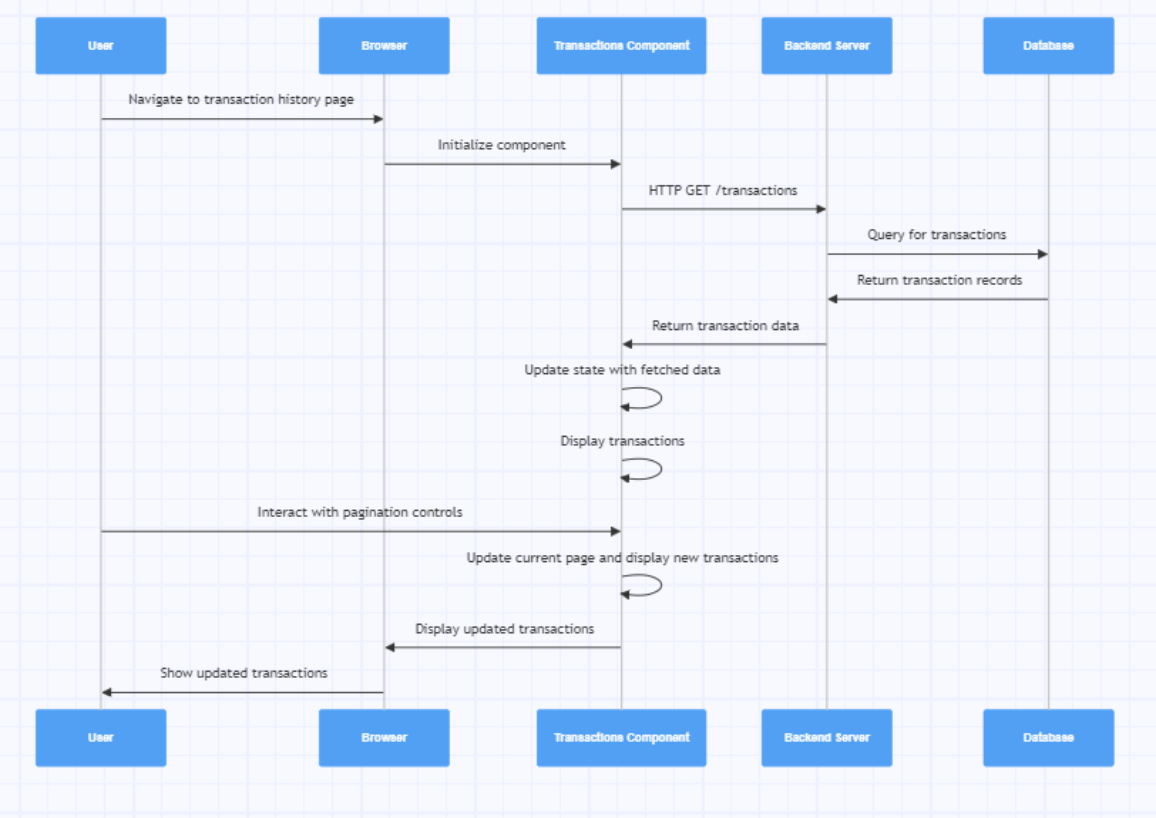


Рисунок 2.8 – Діаграма послідовності (Sequence Diagram) перегляду історії транзакцій

3) Перегляд аналітичних даних по криптовалютах. Користувач переходить на сторінку токенів, браузер ініціалізує компонент Tokens. Компонент Tokens отримує дані про всі токени, виконує фільтрацію, оновлює та відображає список. Користувач вибирає валюту, компонент оновлює та відображає токени в обраній валюті. Користувач взаємодіє з полем пошуку, компонент фільтрує та відображає відфільтровані токени. Користувач взаємодіє з контролами пагінації, компонент оновлює та відображає нову сторінку токенів. Браузер показує користувачеві оновлені токени.

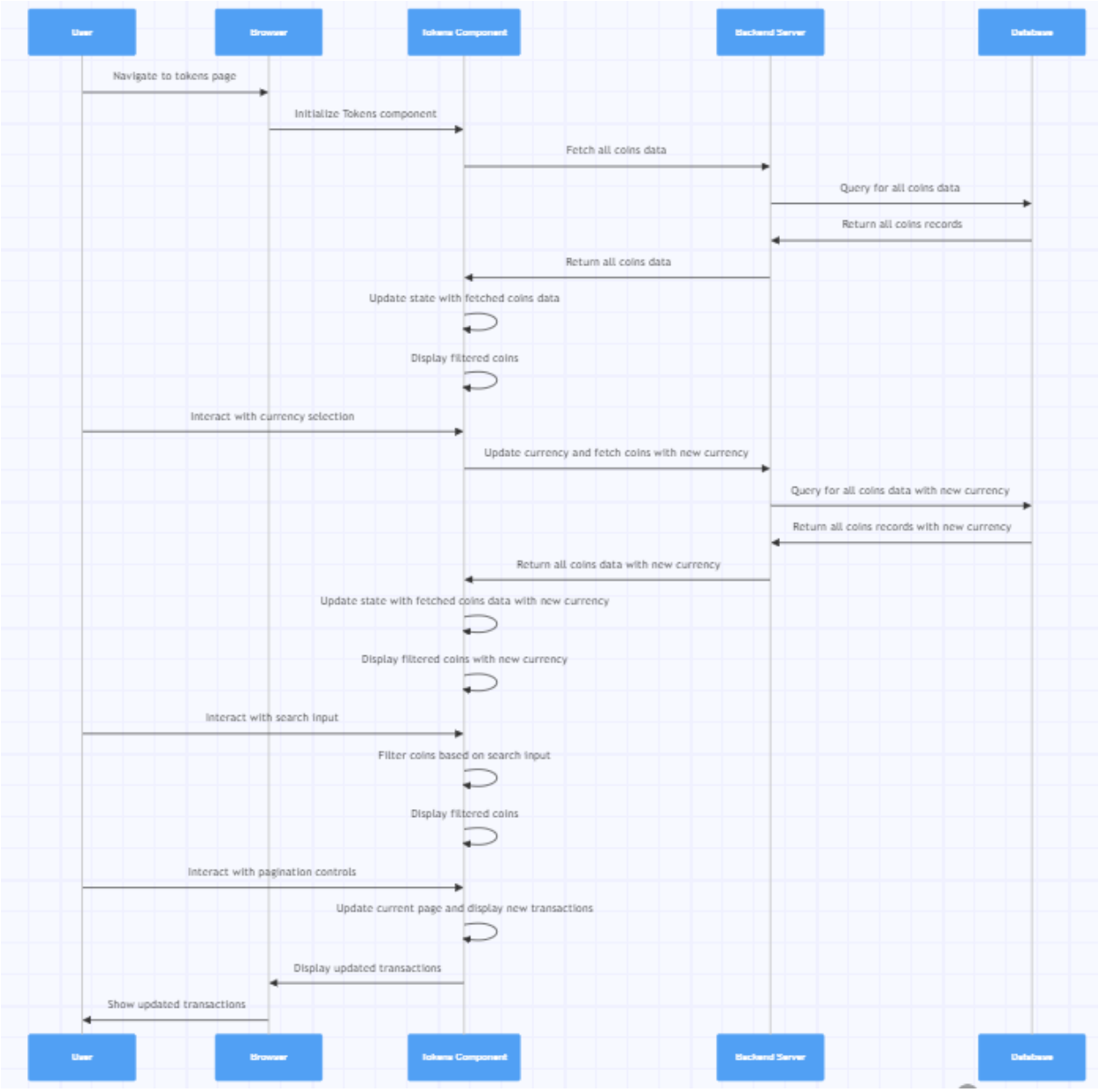


Рисунок 2.9 – Діаграма послідовності (Sequence Diagram) перегляду аналітичних даних по криптовалютах

Архітектура системи

Ця система розроблена з використанням клієнт-серверної архітектури, де фронтенд застосунок комунікує з бекенд сервером через HTTP запити. Бекенд сервер, у свою чергу, виконує логіку системи, взаємодіє з базою даних для зберігання та отримання інформації і взаємодіє з зовнішніми сервісами для отримання додаткових даних, наприклад, обмінних курсів. База даних забезпечує постійне зберігання даних системи.

У моїй системі для реєстрації транзакцій використовується Binance Smart Chain (BSC).

Binance Smart Chain (BSC) - це блокчейн-платформа, створена Binance, яка забезпечує швидкі та низьковартісні транзакції для децентралізованих додатків (DApps) і смарт-контрактів. BSC працює паралельно з Binance Chain, що дозволяє користувачам мати найкраще з обох світів: високу продуктивність Binance Chain і функціональність смарт-контрактів у BSC.

У цій системі блокчейн використовується для реєстрації транзакцій, забезпечуючи прозорість та безпеку операцій. Кожна транзакція має унікальний ідентифікатор та зберігається у блоках, що дозволяє користувачам перевіряти та переглядати історію операцій через ресурс BscScan [13].

Компоненти системи:

1) Фронтенд застосунок: Реалізований у вигляді веб-додатку, що надає користувачам зручний інтерфейс для взаємодії з криптовалютною біржею.

2) Бекенд сервер: Відповідає за обробку запитів від фронтенду, взаємодію з базою даних та зовнішніми сервісами, а також логіку обробки операцій і забезпечення безпеки.

3) База даних: Зберігає дані про транзакції, гаманці та інші важливі дані для функціонування системи.

4) CoinGecko API: Використовується для отримання поточних курсів криптовалют, інформації про ринки та іншої статистики, необхідної для відображення користувачам.

5) 1inch API: Використовується для здійснення обміну криптовалют та взаємодії з блокчейном для виконання транзакцій.

6) Блокчейн: Виконує функцію запису та підтвердження транзакцій, забезпечуючи безпечність і прозорість операцій.

Схема архітектури системи зображена на рисунку 2.10.

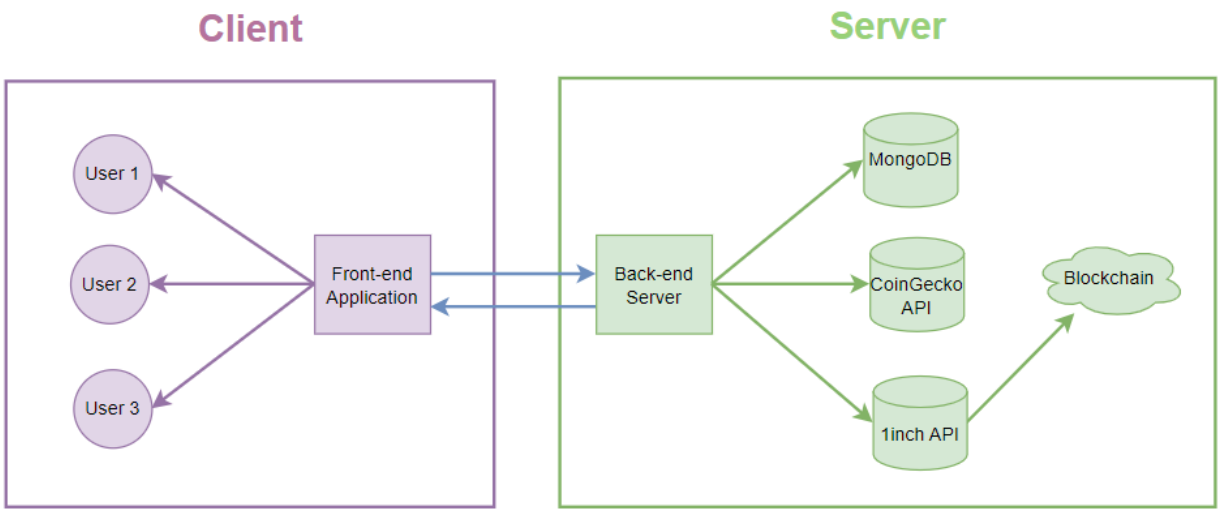


Рисунок 2.10 – Схема архітектури системи криптовалютної біржі

РОЗДІЛ 3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Інформаційне забезпечення

Загальна характеристика інформаційного забезпечення (ІЗ)

Інформаційне забезпечення криптовалютної біржі являє собою сукупність даних, методів їхнього зберігання та обробки, а також системи контролю, необхідних для забезпечення функціонування та безпеки системи. Основними компонентами інформаційного забезпечення даної системи є дані про криптовалюти, історія транзакцій користувачів та інформація про стан гаманців.

Структура інформаційного забезпечення включає в себе декілька ключових елементів. По-перше, це API сервісів, таких як 1inch, що забезпечує обмін криптовалют за найнижчими цінами на ринку, та CoinGecko, який надає актуальну інформацію про криптовалюти. Дані з цих API є динамічними і оновлюються в режимі реального часу, що дозволяє користувачам отримувати найактуальнішу інформацію для прийняття рішень.

Методи контролю інформації включають в себе аутентифікацію та авторизацію користувачів через підключення гаманця MetaMask, що забезпечує високий рівень безпеки і конфіденційності даних.

Вимоги до надійності і достовірності інформації включають забезпечення цілісності даних, мінімізацію можливості втрати інформації та зниження ризику помилок під час обміну криптовалют.

Загальна схема інформаційного забезпечення включає API для отримання даних про курси криптовалют, підключення гаманців для проведення транзакцій, базу даних MongoDB для збереження історії транзакцій. Конкретними елементами, які використовуються при функціонуванні інформаційної системи, є API 1inch і CoinGecko, гаманці MetaMask, база даних MongoDB.

Загальна структура інформаційного забезпечення показана на рис.3.1.

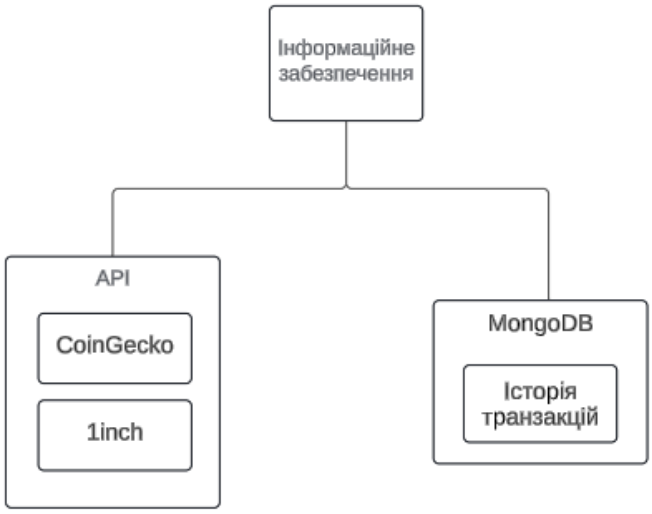


Рисунок 3.1 – Інформаційне забезпечення

Організація збору і передання первинної інформації

Збір і передавання первинної інформації в рамках розробки криптовалютної біржі є критичним етапом для забезпечення її ефективного функціонування. Основними джерелами інформації виступають зовнішні API-сервіси, такі як 1inch та CoinGecko. API 1inch, зокрема 1inch Swap API v6 та Pathfinder, являють собою передовий алгоритм виявлення та маршрутизації, який пропонує обмін активами за найкращими ставками на ринку. Pathfinder знаходить найефективніші шляхи для обміну токенів, здатний розподіляти між різними протоколами та навіть різною глибиною ринку в межах одного протоколу за найкоротший час. CoinGecko [6] надає дані про ринкову інформацію криптовалют. Внутрішніми джерелами інформації є системи, що забезпечують підключення гаманців MetaMask для здійснення транзакцій, а також база даних для збереження історії транзакцій. Додатковим джерелом інформації є користувачі, які вводять запити на обмін криптовалют через веб-інтерфейс.

Періодичність збору даних з API 1inch і CoinGecko є високочастотною, з оновленням інформації в режимі реального часу, що дозволяє користувачам отримувати найактуальніші дані. Спосіб надання цієї інформації здійснюється через автоматизовані запити до API, що забезпечує безперервний потік даних

у систему. Дані передаються у форматі JSON, що є стандартом для веб-сервісів і дозволяє легко інтегрувати їх у базу даних MongoDB.

При підключенні гаманця MetaMask, користувачі використовують розширення браузера, яке дозволяє їм взаємодіяти з криптовалютними операціями. Вони авторизуються в розширенні, використовуючи свої дані доступу, і підтверджують транзакції, вказуючи необхідні параметри. Проте важливо відзначити, що це розширення не проводить аутентифікацію користувачів в межах внутрішньої системи криптовалютної біржі. Замість цього воно лише надає можливість користувачам виконувати операції з криптовалютами у веб-середовищі, які потім обробляються системою біржі. Таким чином, розширення MetaMask діє як інтерфейс між користувачем і криптовалютною системою, спрощуючи процес виконання та підтвердження транзакцій безпосередньо в браузері.

Користувачі криптовалютної біржі є також важливим джерелом інформації. Вони вводять запити на обмін криптовалют через веб-інтерфейс. Ці запити передаються до системи у реальному часі та обробляються для виконання транзакцій. Вхідні повідомлення від користувачів надходять безпосередньо до серверної частини системи, де відбувається їх обробка та передача необхідної інформації до відповідних модулів, зокрема до API 1inch для виконання обміну та до бази даних для збереження інформації про транзакцію.

Таким чином, організація збору і передавання первинної інформації у криптовалютній біржі забезпечується через автоматизовані зовнішні та внутрішні джерела даних, включаючи введення даних користувачами, що гарантує своєчасне подання та обробку інформації для подальшого використання в системі.

Структура інформаційних масивів

Під час проєктування форм первинних документів, машинограм та відеокадрів для криптовалютної біржі, важливим етапом є також розробка структури інформаційних масивів. У цьому контексті розглянемо структуру

інформаційних масивів, які використовуються для збереження даних про транзакції на біржі.

В таблиц 3.1 подано інформаційний масив колекції Transaction.

Таблиця 3.1

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний / вторинний ключ	Умова назначення	Обов'язкове поле	Індексне поле	
ID	transaction_id	-	ObjectID	PK	unique	так	ІНД	-
Номер транзакції	number	-	double	-	unique	так	ІНД	-
Номер гаманця	wallet_id	-	string	FK	-	так	ІДД	-
З криптовалюти	from_ticker	-	string	FK	-	так	ІДД	-
В криптовалюту	to_ticker	-	string	FK	-	так	ІДД	-
Віддано	input	Q1	double	-	-	так	-	-
Отримано	output	Q_2	double	-	-	так	-	-
Статус	status	-	string	-	-	так	-	-

В таблиці 3.2 подано інформаційний масив колекції Wallet.

Таблиця 3.2

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний / вторинний ключ	Умова назначення	Обов'язкове поле	Індексне поле	
ID	wallet_id	-	ObjectID	РК	unique	так	ІНД	-
Адреса гаманця	address	-	string	-	-	так	-	-

В таблиці 3.3 подано інформаційний масив колекції Cryptocurrency.

Таблиця 3.3

Найменування	Ідентифікатор у програмі	Умовне позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний / вторинний ключ	Умова назначення	Обов'язкове поле	Індексне поле	
Тікер криптовалюти	ticker	-	string	РК	unique	так	ІНД	-
Назва криптовалюти	name	-	string	-	-	так	-	-
Зображення криптовалюти	img	-	URL	-	-	так	-	-
Адреса криптовалюти	address	-	string	-	-	так	-	-
Кількість десяткових знаків	decimals	-	double	-	-	так	-	-

Вибір СКБД

Для реалізації проєкту з розробки криптовалютної біржі було обрано MongoDB [8] як систему керування базами даних (СКБД). Обгрунтування цього вибору базується на кількох ключових аспектах. MongoDB є документно-орієнтованою СКБД, яка дозволяє зберігати дані у вигляді JSON-подібних документів, що добре відповідає потребам проєкту з урахуванням структури даних, таких як інформація про транзакції та профілі користувачів.

MongoDB відрізняється високою продуктивністю та масштабованістю, що важливо для проєктування криптовалютної біржі з великим обсягом даних та високою швидкістю обробки транзакцій. Крім того, її гнучка схема дозволяє легко вносити зміни до схеми даних в майбутньому, що є важливим аспектом для системи, яка може зазнавати постійних змін та розширень у зв'язку з розвитком криптовалютного ринку.

Інфологічна модель бази даних

Інфологічна модель – це ER-модель, яка відображає сутності предметної області та звязки між ними.

Сутності та атрибути:

Гаманець (Wallet):

- wallet_id (PK): унікальний ідентифікатор гаманця.
- address: адреса гаманця.

Криптовалюта (Cryptocurrency):

- ticker (PK): унікальний тікер криптовалюти.
- name: назва криптовалюти.
- img: зображення криптовалюти.
- address: адреса криптовалюти.
- decimals: кількість десяткових знаків.

Транзакція (Transaction):

- transaction_id (PK): унікальний ідентифікатор транзакції.
- number: номер транзакції.
- wallet_id (FK): зовнішній ключ, що посилається на гаманець.

– from_ticker (FK): зовнішній ключ, що посилається на криптовалюту, з якої проводиться транзакція.

– to_ticker (FK): зовнішній ключ, що посилається на криптовалюту, в яку проводиться транзакція.

– input: вхідні дані транзакції.

– output: вихідні дані транзакції.

– status: статус транзакції.

Зв'язки:

– Кожен гаманець може мати багато транзакцій, але кожна транзакція здійснюється з одним конкретним гаманцем (один до багатьох).

– Кожна транзакція пов'язана з криптовалютою, з якої проводиться операція, та криптовалютою, в яку проводиться операція, і кожна криптовалюта може брати участь у багатьох транзакціях (багато до багатьох).

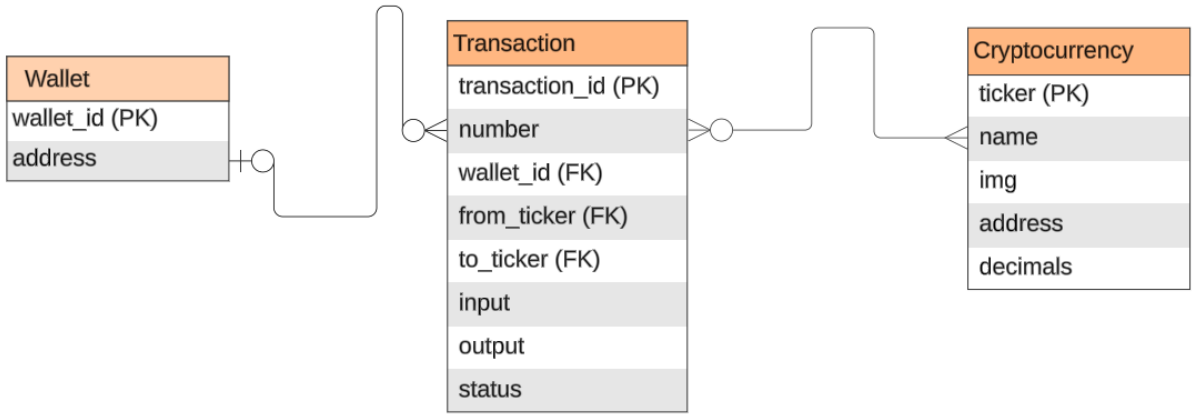


Рисунок 3.2 – Інфологічна модель бази даних

3.2 Технічне забезпечення

Загальні положення та схема автоматизації.

Система автоматизації для криптовалютної біржі передбачає існування віддаленого серверу, на якому знаходиться ядро системи, яке забезпечує виконання всіх функцій та операцій, пов'язаних з обміном криптовалют. Також на цьому сервері знаходяться необхідні файли та інші компоненти програмного забезпечення, що забезпечують правильну роботу системи.

Віддалений сервер MongoDB призначений для зберігання бази даних, в якій зберігається інформація про історію транзакцій.

Клієнти отримують доступ до системи через веб-додаток, який працює в мережі Інтернет. Цей веб-додаток надає користувачам можливість взаємодіяти з біржею, виконувати торгівельні операції та переглядати інформацію про ринок криптовалют. Всі операції виконуються в мережі Binance Smart Chain (BSC), що забезпечує швидкість та надійність обміну криптовалют.

Структура комплексу технічних засобів.

Під час визначення технічних вимог до сервера для ефективної роботи криптовалютної біржі, важливо враховувати кілька ключових аспектів. По-перше, процесор повинен мати достатню продуктивність для обробки великого обсягу транзакцій та запитів користувачів. Тому рекомендується використовувати процесори класу Intel Xeon або еквівалентні серверні моделі, які забезпечують високу швидкодію та ефективність обчислень.

Другим важливим аспектом є об'єм оперативної пам'яті, яка відповідає за швидкодію обробки даних та виконання програм. Мінімальний обсяг оперативної пам'яті повинен становити 16 ГБ, але рекомендується використання моделей з обсягом 32 ГБ або більше для забезпечення оптимальної продуктивності.

У зв'язку з необхідністю ефективного зберігання та доступу до даних, рекомендується використання жорстких дисків типу SSD з мінімальним обсягом 500 ГБ або еквівалентними моделями для забезпечення швидкого доступу до інформації та зменшення часу відгуку системи на запити користувачів. Оскільки база даних буде зберігатися віддалено на сервері MongoDB, основна функція локального SSD полягатиме в прискоренні операцій з кешування та обробки тимчасових даних.

Щодо мережевих інтерфейсів, важливо використовувати гігабітний Ethernet або вище, щоб забезпечити швидку передачу даних та низьку затримку мережі. Це особливо важливо для забезпечення швидкої обробки транзакцій та підтримки великої кількості одночасних з'єднань.

Нарешті, система охолодження та живлення повинна бути ефективною та надійною, з можливістю резервування, щоб уникнути виникнення перебоїв у роботі сервера через перегрів чи відмову живлення. Додатково, необхідно забезпечити захист даних через резервне копіювання та захист від втрати даних у разі аварійних ситуацій.

3.3 Програмне забезпечення

Програмне забезпечення складається з двох частин: клієнтської (клієнт) та серверної (сервер). Розглянемо їх окремо:

Клієнтське програмне забезпечення: Клієнтське програмне забезпечення написане мовою програмування JavaScript з використанням бібліотек React та Ant Design для створення інтерактивного та зручного інтерфейсу користувача. Крім того, використовуються додаткові бібліотеки для забезпечення роботи з мережею, такі як Axios для здійснення запитів на сервер, та wagmi для взаємодії з блокчейном. При використанні клієнтського програмного забезпечення користувачі мають можливість здійснювати операції обміну криптовалют, переглядати інформацію про криптовалюти, а також переглядати історію транзакцій.

Файл Swap.jsx містить компонент React, який відповідає за відображення обміну криптовалют в інтерфейсі користувача (див. Додаток А). Основні функції цього компонента включають:

1) Відображення форми обміну: Компонент містить поля для введення суми першої криптовалюти та відображення еквівалентної суми другої криптовалюти. Користувач може вибрати криптовалюту для обміну, а також встановити рівень толерантності зносу (slippage).

2) Взаємодія зі зовнішніми сервісами: Компонент взаємодіє з зовнішніми сервісами для отримання інформації про токени, їх ціни, баланс гаманця, ціну за газ (gas fee) тощо. Він використовує бібліотеку Axios для здійснення HTTP-запитів.

3) Виконання обміну: Після введення користувачем необхідних даних та підтвердження зв'язку з підключеним гаманцем, компонент виконує

обмін криптовалют. Для цього він викликає функцію fetchDexSwap, яка викликає відповідний серверний маршрут.

4) Відображення статусу транзакції: Після виконання обміну компонент відображає статус транзакції - чи вона виконана успішно, чи наразі очікує підтвердження.

5) Динамічне оновлення даних: Компонент автоматично оновлює дані про баланс гаманця, ціни токенів та ціну за газ (gas fee) при зміні вхідних даних або параметрів.

6) Відображення допоміжних налаштувань: Компонент містить можливість зміни параметрів обміну, таких як рівень толерантності зносу (slippage) через відповідний поповер.

7) Відображення балансу гаманця та оцінка вартості в USD: Компонент відображає баланс користувача в обраних криптовалютах та оцінку їх вартості в доларах США.

Цей файл виконує важливу роль у взаємодії користувача з додатком, надаючи можливість зручного та безпечного обміну криптовалют.

Файл Tokens.jsx містить компонент React для відображення списку криптовалют за їх ринковою капіталізацією та іншими показниками (див. Додаток Б). Основні функції цього компонента включають:

1) Відображення списку криптовалют: Компонент відображає список криптовалют, відсортованих за їх ринковою капіталізацією. Кожна криптовалюта включає назву, символ, поточну ціну, зміну за останні 24 години та ринкову капіталізацію.

2) Пошук криптовалют: Користувач може використовувати поле введення для пошуку конкретної криптовалюти за назвою. Результати пошуку оновлюються динамічно.

3) Вибір валюти: Користувач може вибрати валюту для відображення цін криптовалют. Варіанти включають долар США (USD), євро (EUR) та українську гривню (UAH).

4) Навігація до окремих криптовалют: Кожна криптовалюта в списку є посиланням, яке перенаправляє користувача на окрему сторінку з додатковою інформацією про вибрану криптовалюту.

5) Динамічне оновлення даних: Компонент автоматично оновлює список криптовалют при зміні вхідних даних або параметрів, таких як зміна валюти.

Файл Tokens.jsx забезпечує користувачам зручний спосіб перегляду та аналізу ринку криптовалют, дозволяючи швидко знаходити та дізнаватися про різні криптовалюти залежно від їх ринкової капіталізації та інших ключових параметрів.

Файл Transactions.jsx містить компонент React для відображення списку транзакцій з пагінацією (див. Додаток В).. Основні функції цього компонента включають:

1) Відображення списку транзакцій: Компонент відображає таблицю з даними транзакцій, включаючи номер транзакції, користувача, початковий та кінцевий актив, вхідну та вихідну суму та статус транзакції.

2) Отримання даних транзакцій з сервера: Компонент використовує useEffect для виконання запиту на сервер за даними транзакцій під час завантаження компонента. Отримані дані обробляються та зберігаються у стані компонента для відображення.

3) Пагінація: Компонент використовує компонент пагінації з Ant Design для розділення списку транзакцій на сторінки. Користувач може переходити між сторінками, щоб переглянути інші транзакції.

4) Динамічне оновлення даних: Компонент автоматично оновлює список транзакцій при зміні сторінки або при отриманні нових даних з сервера.

Файл Transactions.jsx надає користувачам можливість переглядати та аналізувати дані транзакцій, використовуючи зручний інтерфейс з підтримкою пагінації. Це дозволяє ефективно керувати великими обсягами даних та забезпечує зручну навігацію користувачам.

Серверне програмне забезпечення: Серверне програмне забезпечення написане мовою програмування JavaScript з використанням фреймворка Express для створення серверної частини додатку. Також використовуються додаткові бібліотеки для забезпечення роботи сервера, такі як CORS для обробки запитів з різних джерел, Mongoose для взаємодії з базою даних MongoDB. Серверне програмне забезпечення забезпечує обробку запитів від

клієнтської частини, виконання різноманітних операцій з обробкою даних, включаючи збереження та оновлення даних в базі даних.

Вихідний код компонента server.js, який відповідає за обробку серверних запитів та взаємодію з базою даних (див. Додаток Г).

Серверне програмне забезпечення обробляє декілька ключових функцій. Серед них:

1) Отримання списку токенів: Сервер взаємодіє з зовнішнім API для отримання актуальної інформації про доступні токени. Це дозволяє клієнтській частині відображати користувачам актуальні дані про криптовалютні активи.

2) Отримання ціни токенів: Сервер виконує запити до зовнішнього API для отримання поточних цін токенів. Це необхідно для відображення користувачам актуальних курсів криптовалют.

3) Отримання балансу гаманця: Сервер обробляє запити на отримання балансу криптовалютних гаманців користувачів. Використовуючи API, сервер отримує інформацію про баланси різних токенів у вказаних гаманцях.

4) Перевірка дозволів: Сервер перевіряє дозволи на операції з токенами у вказаних гаманцях, що необхідно для забезпечення безпеки та коректності виконання транзакцій.

5) Виконання свопів: Сервер виконує операції обміну криптовалют, взаємодіючи з зовнішнім API для забезпечення точності та надійності обмінних операцій.

6) Збереження транзакцій: Сервер обробляє збереження інформації про транзакції в базі даних MongoDB, що дозволяє користувачам переглядати історію транзакцій.

Таким чином, серверне програмне забезпечення виконує важливу роль у забезпеченні функціональності веб-застосунку, обробляючи запити від клієнтської частини, взаємодіючи з зовнішніми сервісами, та зберігаючи дані у базі даних.

Детальний код кожного компонента можна знайти у відповідних додатках, що дозволяє повністю ознайомитися з реалізацією функціоналу веб-застосунку.

3.4 Результати реалізації інформаційної підсистеми

Веб-застосунок криптовалютної біржі, розроблений у рамках даного дослідження, належно втілює функціонал, що відповідає вимогам сучасних фінансових ринків та враховує специфіку використання криптовалютних активів у контексті цифрової економіки. Основними досягненнями даної підсистеми є інтеграція гаманця, можливість вибору криптовалютних пар, введення обмінних сум та отримання відповідної інформації про обмінні курси.

На рисунку 3.3 представлено інтерфейс, який дозволяє користувачам підключати свій криптовалютний гаманець до системи. Цей функціонал забезпечує безпечність зберігання криптовалютних активів користувачів та забезпечує доступ до них в рамках біржових операцій.

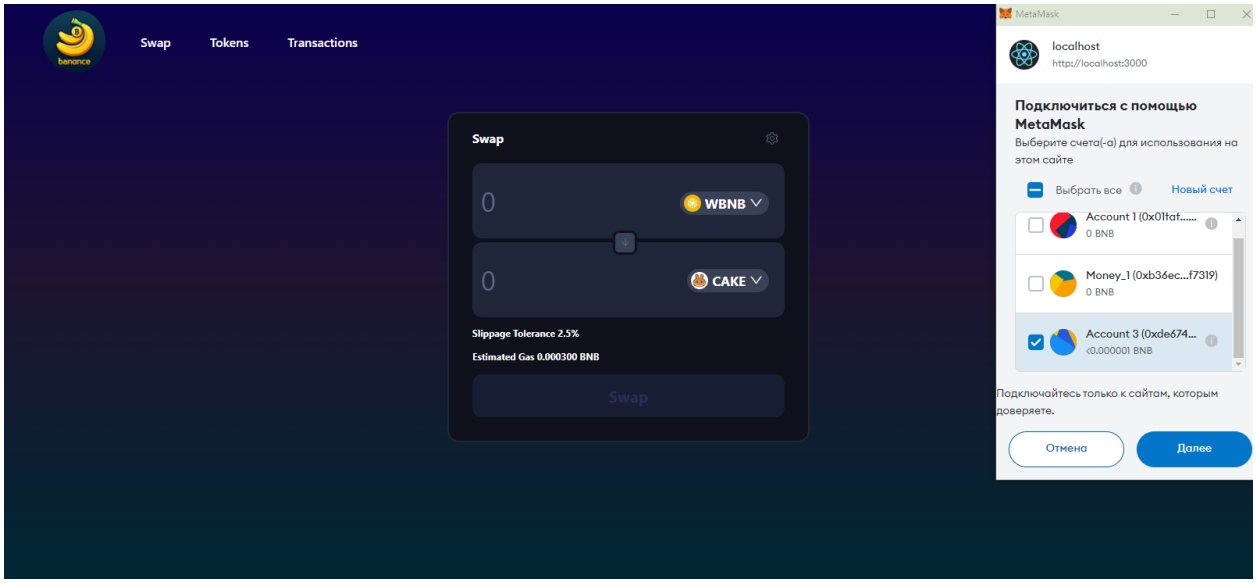


Рисунок 3.3 – Розроблений функціонал підключення гаманця

На рисунку 3.4 можна побачити інтерфейс для розрахунку обсягу обміну криптовалют. Користувачі можуть вибирати криптовалютні пари, вводити обмінні суми та отримувати відповідну інформацію про обмінні курси.

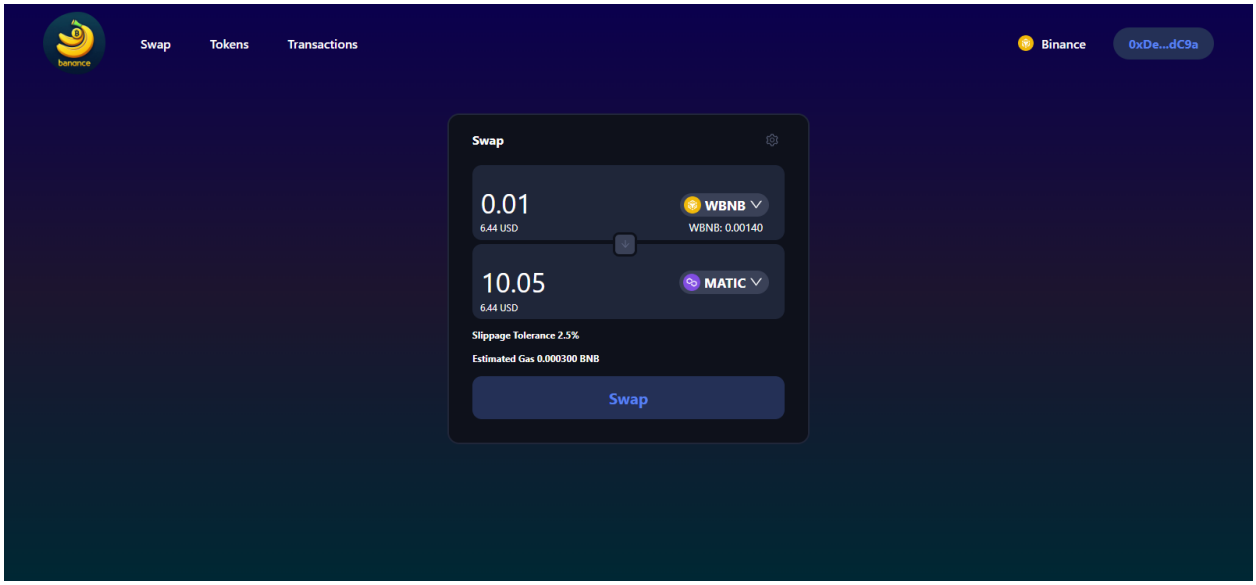


Рисунок 3.4 – Розроблений функціонал розрахунку обсягу обміну

Рисунок 3.5 відображає інтерфейс, де користувачі можуть переглядати історію своїх транзакцій. Він надає зручний інтерфейс для аналізу та відстеження фінансових операцій з криптовалютами.

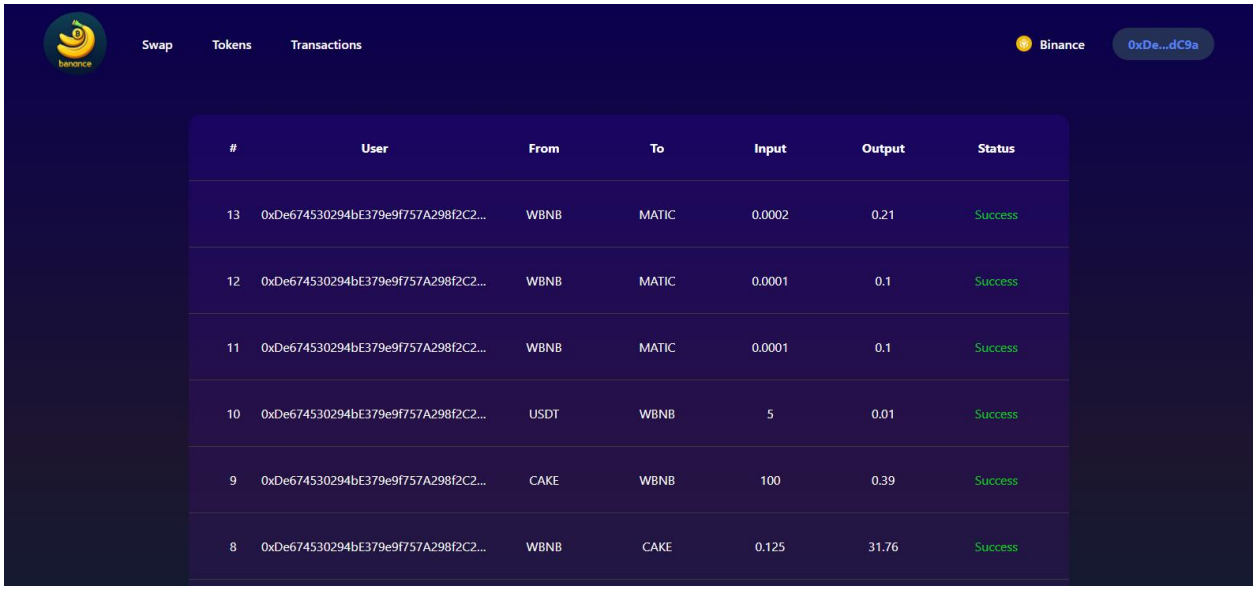


Рисунок 3.5 – Розроблений функціонал відображення історії транзакцій

На рисунку 3.6 показано інтерфейс, де користувачі можуть переглядати список доступних криптовалют. Це допомагає користувачам отримувати інформацію про різні криптовалютні активи та їх характеристики.

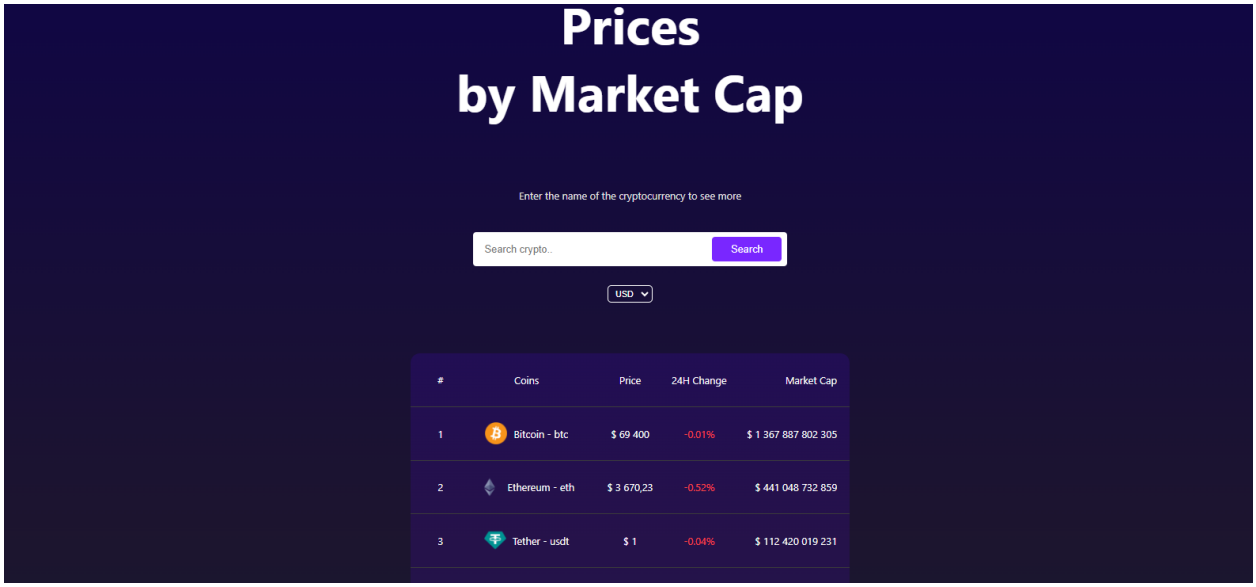


Рисунок 3.6 – Розроблений функціонал відображення списку криптовалют

Рисунок 3.7 відображає інтерфейс, який надає інформацію про конкретну криптовалюту. Користувачі можуть побачити графік цін на криптовалюту, отримати дані про ціну, обсяг, зміни за останні 24 години та інші параметри кожної криптовалюти.

Рисунок 3.7 – Розроблений функціонал відображення інформації по криптовалюті

Програмний продукт надає зручний інтерфейс для користувача, що дозволяє легко керувати фінансовими операціями з криптовалютами. Інтеграція гаманця забезпечує безпечність зберігання криптовалютних активів користувачів та забезпечує доступ до них в рамках біржових операцій.

Результати пілотного впровадження системи на тестовому середовищі виявили позитивні тенденції в її функціональності та ефективності. Зокрема, впровадження алгоритму пошуку найнижчої ціни на інших біржах дозволяє забезпечити оптимальні умови обміну для користувачів у разі низької ліквідності на внутрішній біржі.

В результаті тестування системи було здійснено транзакцію в мережі Binance Smart Chain (BSC), яка була записана в блокчейні. Результати апробації на контрольному прикладі демонструють можливість успішної реалізації функціональності системи в умовах реального фінансового ринку.

Результат транзакції можна побачити за допомогою ресурсу BscScan [13].

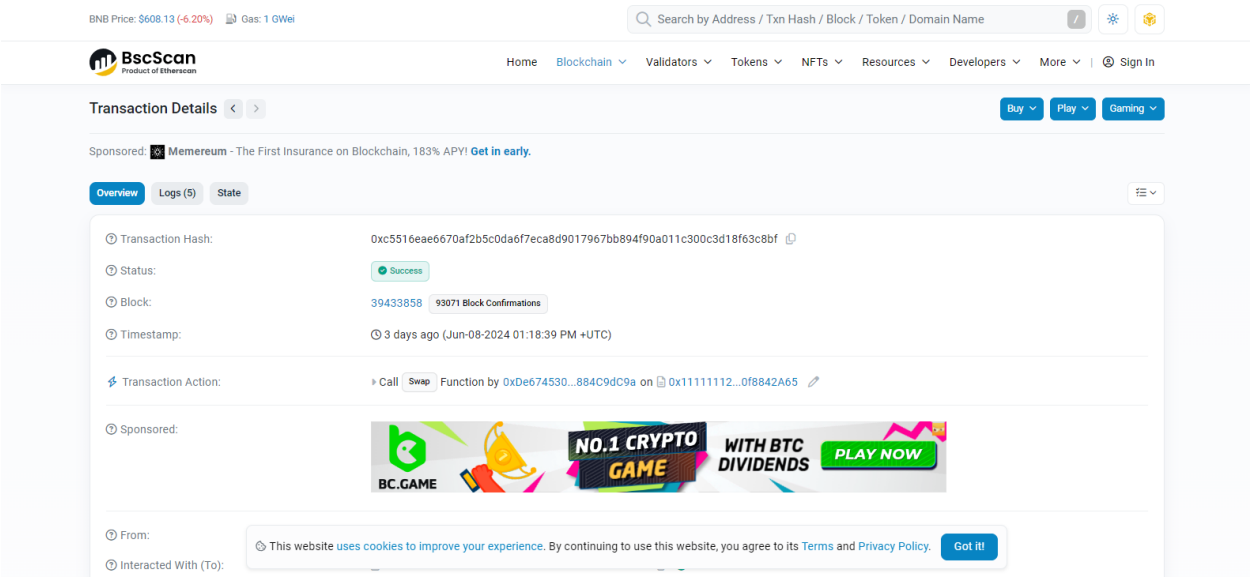


Рисунок 3.8 – Деталі транзакції на ресурсі BscScan

Подальші дослідження та удосконалення можуть бути спрямовані на розширення функціональності підсистеми, включаючи додаткові криптовалютні активи та інтеграцію з іншими фінансовими інструментами.

ВИСНОВКИ

В процесі виконання кваліфікаційного бакалаврського проєкту була розроблена автоматизована система для обміну криптовалют, яка дозволяє користувачам здійснювати операції за найнижчими цінами на ринку за допомогою API 1inch, відображати актуальну інформацію про криптовалюти через API CoinGecko, під'єднувати гаманці MetaMask, а також зберігати історію транзакцій у базі даних MongoDB. Основною мережею для обміну криптовалют є мережа Binance Smart Chain (BSC).

Проведений аналіз показав, що розроблена система забезпечує високу ефективність та безпеку обміну криптовалют. Використання API 1inch дозволяє отримувати найвигідніші курси обміну, що значно підвищує конкурентоспроможність системи. Інтеграція з API CoinGecko [6] надає користувачам доступ до актуальної інформації про курси та інші характеристики криптовалют, що сприяє прийняттю обґрунтованих рішень під час проведення транзакцій. Під'єднання гаманців MetaMask забезпечує зручність та безпеку здійснення операцій, завдяки чому користувачі можуть швидко та легко здійснювати обміни.

Важливою складовою системи є база даних MongoDB, яка дозволяє ефективно зберігати та обробляти історію транзакцій. Це забезпечує можливість подальшого аналізу та відстеження операцій, що є критично важливим для забезпечення прозорості та довіри до системи. Використання мережі Binance Smart Chain (BSC) для обміну криптовалют дозволяє досягти високої швидкості транзакцій та низьких комісій, що є вагомою перевагою в умовах сучасного ринку криптовалют.

Проєкт також виявив деякі обмеження та недоліки, які потребують подальшого вдосконалення. Зокрема, розширити функціонал системи, додавши можливість роботи з іншими мережами та платформами для обміну криптовалют.

Особистий вклад в розробку проєкту полягає в аналізі ринку криптовалют, проєктуванні системи, реалізації основних функціональних компонентів, а також тестуванні та оптимізації системи. Зокрема, я здійснив

інтеграцію API 1inch та CoinGecko, реалізував підключення гаманців MetaMask та організував збереження історії транзакцій у базі даних MongoDB. Результати роботи були впроваджені на практиці, що підтверджує ефективність та життєздатність запропонованих рішень.

Таким чином, виконаний кваліфікаційний бакалаврський проєкт продемонстрував можливість створення ефективної та безпечної системи для обміну криптовалют, яка відповідає сучасним вимогам ринку та потребам користувачів. Отримані результати можуть бути використані як основа для подальших досліджень та розробок у сфері фінансових технологій та криптовалют.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ghosh, A., Gupta, S., Dua, A., & Kumar, N. (2020). Security of Cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects. *J. Netw. Comput. Appl.*, 163, 102635. https://doi.org/10.1016/j.jnca.2020.102635.

2. Rezaeighaleh, H., & Zou, C., 2019. New Secure Approach to Backup Cryptocurrency Wallets. *2019 IEEE Global Communications Conference (GLOBECOM),* pp. 1-6. https://doi.org/10.1109/GLOBECOM38437.2019.9014007.

3. Baum, C., David, B., Frederiksen, T.K. (2021). P2DEX: Privacy-Preserving Decentralized Cryptocurrency Exchange. In: Sako, K., Tippenhauer, N.O. (eds) Applied Cryptography and Network Security. ACNS 2021. Lecture Notes in Computer Science(), vol 12726. Springer, Cham. https://doi.org/10.1007/978-3-030-78372-3_7

4. Mohanta, BK, Panda, SS, & Jena, D. (2018, липень). An Overview of Smart Contract and Use Cases in Blockchain Technology. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT) pp. 1-4 IEEE.

5. Wagmi Docs [Електронний ресурс] – Режим доступу до ресурсу: https://1.x.wagmi.sh/

6. CoinGecko API Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.coingecko.com/reference/introduction.

7. 1inch API Swap Docs [Електронний ресурс] – Режим доступу до ресурсу: https://portal.1inch.dev/documentation/swap/introduction

8. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/.

9. Binance [Електронний ресурс] – Режим доступу до ресурсу: https://www.binance.com/uk-UA.

10. Coinbase [Електронний ресурс] – Режим доступу до ресурсу: https://www.coinbase.com/.

11. Uniswap [Електронний ресурс] – Режим доступу до ресурсу: https://uniswap.org/.

12. SushiSwap [Електронний ресурс] – Режим доступу до ресурсу: https://www.sushi.com/swap.

13. BscScan [Електронний ресурс] – Режим доступу до ресурсу: https://bscscan.com/tx/0xc5516eae6670af2b5c0da6f7eca8d9017967bb894f90a011c300c3d18f63c8bf.

ДОДАТКИ

Додаток А

Вихідний код компонента Swap.jsx

```
import React, { useState, useEffect } from "react";
import { Input, Popover, Radio, Modal, message } from "antd";
import {
  ArrowDownOutlined,
  DownOutlined,
  SettingOutlined,
} from "@ant-design/icons";
import tokenList from "../tokenListBSC.json";
import axios from "axios";
import { useSendTransaction, useWaitForTransaction } from "wagmi";

function Swap(props) {
  const { address, isConnected } = props;
  const [messageApi, contextHolder] = message.useMessage();
  const [slippage, setSlippage] = useState(2.5);
  const [tokenOneAmount, setTokenOneAmount] = useState(null);
  const [tokenTwoAmount, setTokenTwoAmount] = useState(null);
  const [tokenOne, setTokenOne] = useState(tokenList[0]);
  const [tokenTwo, setTokenTwo] = useState(tokenList[1]);
  const [isOpen, setIsOpen] = useState(false);
  const [changeToken, setChangeToken] = useState(1);
  const [prices, setPrices] = useState(null);
  const [gasPrice, setGasPrice] = useState(null);
  const [balance, setBalance] = useState(null);
  const [txDetails, setTxDetails] = useState({
    to: null,
    data: null,
    value: null,
  });
```

```
const { data, sendTransaction } = useSendTransaction({
  request: {
    from: address,
    to: String(txDetails.to),
    data: String(txDetails.data),
    value: String(txDetails.value),
  },
});

const { isLoading, isSuccess } = useWaitForTransaction({
  hash: data?.hash,
});

function handleSlippageChange(e) {
  setSlippage(e.target.value);
}

function changeAmount(e) {
  setTokenOneAmount(e.target.value);
  if (
    e.target.value &&
    prices &&
    prices[tokenOne.address] &&
    prices[tokenTwo.address]
  ) {
    const priceOne = prices[tokenOne.address];
    const priceTwo = prices[tokenTwo.address];
    const tokenTwoEquivalent = (e.target.value * priceOne) / priceTwo;
    setTokenTwoAmount(tokenTwoEquivalent.toFixed(2));
  } else {
    setTokenTwoAmount(null);
```

```
  }
}

function switchTokens() {
  setPrices(null);
  setTokenOneAmount(null);
  setTokenTwoAmount(null);
  const one = tokenOne;
  const two = tokenTwo;
  setTokenOne(two);
  setTokenTwo(one);
  fetchPrices(two.address, one.address);
}

function openModal(asset) {
  setChangeToken(asset);
  setIsOpen(true);
}

function modifyToken(i) {
  setPrices(null);
  setTokenOneAmount(null);
  setTokenTwoAmount(null);
  if (changeToken === 1) {
    setTokenOne(tokenList[i]);
    fetchPrices(tokenList[i].address, tokenTwo.address);
  } else {
    setTokenTwo(tokenList[i]);
    fetchPrices(tokenOne.address, tokenList[i].address);
  }
  setIsOpen(false);
}
```

```
function formatGasPrice(gasPrice) {
  const formattedGasPrice = Number(gasPrice) / 10 ** 13;

  return formattedGasPrice.toFixed(6);
}

function displayBalance(token, address) {
  if (balance && balance[address] && balance[address][token.address]) {
    const formattedBalance =
      Number(balance[address][token.address]) / 10 ** 18;
    return (
      <>
        {token.ticker}: {formattedBalance.toFixed(5)}
      </>
    );
  }
  return null;
}

function calculateTokenOneValueInUSD(amount, prices) {
  if (!amount || !prices || !prices[tokenOne.address]) {
    return null;
  }

  const priceOne = prices[tokenOne.address];
  const tokenOneValue = amount * priceOne;
  return tokenOneValue.toFixed(2);
}

function formatWith18Decimals(tokenOneAmount) {
  const amountString = String(tokenOneAmount);
```

```
  const [integerPart, fractionalPart] = amountString.split(".");

  const paddedFractionalPart = fractionalPart
    ? fractionalPart.padEnd(18, "0")
    : "000000000000000000";

  const formattedAmount = integerPart + paddedFractionalPart;

  return formattedAmount;
}

async function fetchGasPrice() {
  const res = await axios.get("http://localhost:3001/api/gas-price");
  setGasPrice(res.data);
}

async function fetchWalletBalance() {
  try {
    const res = await axios.post("http://localhost:3001/walletBalance", {
      tokens: [tokenOne.address, tokenTwo.address],
      wallets: [address],
    });
    setBalance(res.data);
  } catch (error) {
    console.error("Error fetching wallet balance:", error);
  }
}

async function fetchPrices(one, two) {
  try {
    const addresses = `${one},${two}`;
    const res = await axios.get(`http://localhost:3001/tokenPricee`, {
```

```
      params: { addresses: addresses },
    });
    setPrices(res.data);
  } catch (error) {
    console.error("Error fetching prices:", error);
  }
}

async function fetchDexSwap() {
  try {
    const allowanceResponse = await axios.get(
      `http://localhost:3001/allowance`,
      {
        params: {
          tokenAddress: tokenOne.address,
          walletAddress: address,
        },
      }
    );

    if (allowanceResponse.data.allowance === "0") {
      const approveResponse = await axios.get(
        `http://localhost:3001/transaction`,
        {
          params: {
            tokenAddress: tokenOne.address,
          },
        }
      );

      setTxDetails(approveResponse.data);
      console.log("Not approved");
```

```
      return;
    }
    console.log("Approved");
    const formattedAmount = formatWith18Decimals(tokenOneAmount);
    const performSwap = await axios.get(`http://localhost:3001/swap`, {
      params: {
        fromToken: tokenOne.address,
        toToken: tokenTwo.address,
        amount: formattedAmount,
        walletAddress: address,
        slippage: slippage,
      },
    });
    setTxDetails(performSwap.data.tx);
  } catch (error) {
    console.error("Error approving data:", error.message);
  }
}

function calculateTokenTwoValueInUSD(amount, prices) {
  if (!amount || !prices || !prices[tokenTwo.address]) {
    return null;
  }

  const priceTwo = prices[tokenTwo.address];
  const tokenTwoValue = amount * priceTwo;
  return tokenTwoValue.toFixed(2);
}

useEffect(() => {
  fetchWalletBalance(tokenList[0].address);
}, []);
```

```
useEffect(() => {
  fetchGasPrice();
}, []);

useEffect(() => {
  fetchPrices(tokenList[0].address, tokenList[1].address);
}, []);

useEffect(() => {
  if (txDetails.to && isConnected) {
    sendTransaction();
  }
}, [txDetails]);

useEffect(() => {
  messageApi.destroy();

  if (isLoading) {
    messageApi.open({
      type: "loading",
      content: "Transaction is Pending...",
      duration: 3,
    });
    axios
      .post("http://localhost:3001/add-transaction", {
        user: address,
        from: tokenOne.ticker,
        to: tokenTwo.ticker,
        input: tokenOneAmount,
        output: tokenTwoAmount,
        status: "Success",
```

```
      })
      .then((response) => {
        console.log("Transaction added:", response.data);
        setTokenOneAmount(null);
        setTokenTwoAmount(null);
      })
      .catch((error) => {
        console.error("Error adding transaction:", error);
      });
  }
}, [isLoading]);

useEffect(() => {
  messageApi.destroy();

  if (isSuccess) {
    console.log("Is Successful");
    messageApi.open({
      type: "success",
      content: "Transaction Successful",
      duration: 1.5,
    });
  }
}, [isSuccess]);

const settings = (
  <>
    <div>Slippage Tolerance</div>
    <div>
      <Radio.Group value={slippage} onChange={handleSlippageChange}>
        <Radio.Button value={0.5}>0.5%</Radio.Button>
        <Radio.Button value={2.5}>2.5%</Radio.Button>
```

```
        <Radio.Button value={5}>5.0%</Radio.Button>
      </Radio.Group>
    </div>
  </>
);
return (
  <>
    {contextHolder}
    <Modal
      open={isOpen}
      footer={null}
      onCancel={() => setIsOpen(false)}
      title="Select a token"
    >
      <div className="modalContent">
        {tokenList?.map((e, i) => {
          return (
            <div
              className="tokenChoice"
              key={i}
              onClick={() => modifyToken(i)}
            >
              <img src={e.img} alt={e.ticker} className="tokenLogo" />
              <div className="tokenChoiceNames">
                <div className="tokenName">{e.name}</div>
                <div className="tokenTicker">{e.ticker}</div>
              </div>
            </div>
          );
        })}
      </div>
    </Modal>
```

```
          <div className="tradeBox">
            <div className="tradeBoxHeader">
              <h4>Swap</h4>
              <Popover
                content={settings}
                title="Settings"
                trigger="click"
                placement="bottomRight"
              >
                <SettingOutlined className="cog" />
              </Popover>
            </div>
            <div className="inputs">
              <Input
                placeholder="0"
                value={tokenOneAmount}
                onChange={changeAmount}
                disabled={!prices}
              />
              <Input placeholder="0" value={tokenTwoAmount} disabled={true} />
              <div className="switchButton" onClick={switchTokens}>
                <ArrowDownOutlined className="switchArrow" />
              </div>
              <div className="assetOne" onClick={() => openModal(1)}>
                                <img src={tokenOne.img} alt="assetOneLogo"
classNameName="assetLogo" />
                {tokenOne.ticker}
                <DownOutlined />
              </div>
              <div className="assetTwo" onClick={() => openModal(2)}>
                                <img src={tokenTwo.img} alt="assetOneLogo"
classNameName="assetLogo" />
```

```
            {tokenTwo.ticker}
            <DownOutlined />
          </div>
          <div className="assetOneDisplaydInUSD">
            {prices &&
              calculateTokenOneValueInUSD(tokenOneAmount, prices) !== null
&& (
                <p>{calculateTokenOneValueInUSD(tokenOneAmount, prices)}
USD</p>
              )}
          </div>
          <div className="assetTwoDisplaydInUSD">
            {prices &&
              calculateTokenTwoValueInUSD(tokenTwoAmount, prices) !== null
&& (
                <p>{calculateTokenTwoValueInUSD(tokenTwoAmount, prices)}
USD</p>
              )}
          </div>

          <div className="assetOneWalletBalance">
            {displayBalance(tokenOne, address)}
          </div>
          <div className="assetTwoWalletBalance">
            {displayBalance(tokenTwo, address)}
          </div>
          <div className="slippageContainer">
            Slippage Tolerance {slippage}%
          </div>
          <div className="slippageContainer">
           Estimated Gas {gasPrice && formatGasPrice(gasPrice.standard)} BNB
          </div>
```

```
        </div>
        <div
          className="swapButton"
          disabled={!tokenOneAmount || !isConnected}
          onClick={fetchDexSwap}
        >
          Swap
        </div>
      </div>
    </>
  );
}

export default Swap;
```

```
import React, { useContext, useEffect, useState } from "react";
import { CoinContext } from "../context/CoinContext";
import { Link } from "react-router-dom";
import "./../styles/Tokens.css";

function Tokens() {
  const { allCoin, currency, setCurrency } = useContext(CoinContext);
  const [displayCoin, setDisplayCoin] = useState([]);
  const [input, setInput] = useState("");

  const inputHandler = (event) => {
    setInput(event.target.value);
    if (event.target.value === "") {
      setDisplayCoin(allCoin);
    }
  };

  const searchHandler = async (event) => {
    event.preventDefault();
    const coins = await allCoin.filter((item) => {
      return item.name.toLowerCase().includes(input.toLowerCase());
    });

    setDisplayCoin(coins);
  };

  useEffect(() => {
    setDisplayCoin(allCoin);
  }, [allCoin]);
```

```
const currencyHandler = (event) => {
  switch (event.target.value) {
    case "usd": {
      setCurrency({ name: "usd", symbol: "$" });
      break;
    }
    case "eur": {
      setCurrency({ name: "eur", symbol: "€" });
      break;
    }
    case "uah": {
      setCurrency({ name: "uah", symbol: "₴" });
      break;
    }
    default: {
      setCurrency({ name: "usd", symbol: "$" });
      break;
    }
  }
};

return (
  <div className="home">
    <div className="hero">
      <h1>
        Cryptocurrency Prices <br /> by Market Cap{" "}
      </h1>
      <p>Enter the name of the cryptocurrency to see more</p>
      <form onSubmit={searchHandler}>
        <input
          onChange={inputHandler}
          value={input}
```

```
      list="coinlist"
      type="text"
      placeholder="Search crypto.. "
      required
    />

    <datalist id="coinlist">
      {allCoin.map((item, index) => (
        <option key={index} value={item.name} />
      ))}
    </datalist>
    <button type="submit">Search</button>
  </form>
  <div className="currency">
    <select onChange={currencyHandler}>
      <option value="usd">USD</option>
      <option value="eur">EUR</option>
      <option value="uah">UAH</option>
    </select>
  </div>
</div>
<div className="crypto-table">
  <div className="table-layout">
    <p>#</p>
    <p className="coins-column">Coins</p>
    <p>Price</p>
    <p style={{ textAlign: "center" }}>24H Change</p>
    <p className="market-cap">Market Cap</p>
  </div>
  {displayCoin.slice(0, 15).map((item, index) => (
    <Link to={`/coin/${item.id}`} className="table-layout" key={index}>
      <p>{item.market_cap_rank}</p>
```

```
            <div>
              <img src={item.image} alt="" />
              <p>{item.name + " - " + item.symbol}</p>
            </div>
            <p>
              {currency.symbol} {item.current_price.toLocaleString()}
            </p>
            <p
             className={item.price_change_percentage_24h > 0 ? "green" : "red"}
            >
              {Math.floor(item.price_change_percentage_24h * 100) / 100}%
            </p>
            <p className="market-cap">
              {currency.symbol} {item.market_cap.toLocaleString()}
            </p>
          </Link>
        ))}
      </div>
    </div>
  );
}

export default Tokens;
```

```
import React, { useEffect, useState } from "react";
import "./../styles/Transactions.css";
import axios from "axios";
import { Pagination } from "antd";

const Transactions = () => {
  const [transactionsData, setTransactionsData] = useState([]);
  const [page, setPage] = useState(1);
  const [transactionsPerPage] = useState(8);

  useEffect(() => {
    axios
      .get(`http://localhost:3001/transactions`)
      .then((response) => {
        if (Array.isArray(response.data)) {
          const reversedData = response.data.reverse();
          setTransactionsData(reversedData);

        } else {
          console.error("Data is not an array", response.data);
        }
      })
      .catch((error) => {
        console.error("There was an error fetching the transactions!", error);
      });
  }, []);

  const handlePageChange = (page) => {
    setPage(page);
  };
```

```
const indexOfLastTransaction = page * transactionsPerPage;
    const indexOfFirstTransaction = indexOfLastTransaction - transactionsPerPage;
const currentTransactions = transactionsData.slice(
  indexOfFirstTransaction,
  indexOfLastTransaction
);

return (
  <div className="transactions-home">
    <div className="transactions-table">
      <div className="transactions-table-layout header">
        <p>#</p>
        <p>User</p>
        <p>From</p>
        <p>To</p>
        <p>Input</p>
        <p>Output</p>
        <p className="transactions-status">Status</p>
      </div>
      {currentTransactions.map((transaction) => (
        <div className="transactions-table-layout" key={transaction._id}>
          <p>{transaction.number}</p>
          <p>{transaction.user}</p>
          <p>{transaction.from}</p>
          <p>{transaction.to}</p>
          <p>{transaction.input}</p>
          <p>{transaction.output}</p>
          <p
                                    className={`transactions-status ${transaction.status.toLowerCase()}`}
```

```
            >
              {transaction.status}
            </p>
          </div>
        ))}
      </div>
      <div className="pagination-container">
        <Pagination
          className="ant-pagination"
          current={page}
          pageSize={transactionsPerPage}
          total={transactionsData.length}
          onChange={handlePageChange}
        />
      </div>
    </div>
  );
};

export default Transactions;
```

```
const express = require("express");
const axios = require("axios");
const cors = require("cors");
const mongoose = require("mongoose");
const Transaction = require("./models/transaction");
require("dotenv").config();

const app = express();
const port = 3001;

const API_KEY = process.env.API_KEY;
const MONGODB_URL = process.env.MONGODB_URL;

const tokenList = "https://api.1inch.dev/swap/v6.0/56/tokens";

const headers = {
  accept: "application/json",
  Authorization: `Bearer ${API_KEY}`,
};

app.use(
  cors({
    origin: "http://localhost:3000",
  })
);

app.get("/tokenList", (req, res) => {
  axios
    .get(tokenList, { headers })
    .then((response) => {
```

```
      res.json(response.data);
    })
    .catch((error) => {
      console.error(error);
      res.status(500).json({ error: "Internal Server Error" });
    });
});

app.use(express.json());

app.get("/tokenPricee", async (req, res) => {
  const { addresses } = req.query;

  async function fetchTokenPrices(addresses) {
    const url = `https://api.1inch.dev/price/v1.1/56/${addresses}`;
    const config = {
      headers: {
        Authorization: `Bearer ${API_KEY}`,
      },
      params: {
        currency: "USD",
      },
    };

    try {
      const response = await axios.get(url, config);
      const prices = response.data;


      Object.keys(prices).forEach((token) => {
        console.log(`Price of ${token} compared to USD:`, prices[token]);
      });
```

```
      return prices;
    } catch (error) {
      if (error.response && error.response.status === 429) {
        console.error(
          "Rate limit fetching price exceeded:",
          error.response.data
        );
      } else {
        console.error("Error fetching prices:", error.message);
      }
      return null;
    }
  }

  const tokenPrices = await fetchTokenPrices(addresses);
  res.json(tokenPrices);
});

app.post("/walletBalance", async (req, res) => {
  console.log("Received request for /walletBalance");
  const { tokens, wallets } = req.body;

  console.log("Waiting for 2000ms...");
  await delay(2000);
  console.log("Timer finished, continuing execution...");

  async function getWalletBalance() {
    const url =
      "https://api.1inch.dev/balance/v1.2/56/balances/multiple/walletsAndToke
ns";
```

```
  const config = {
    headers: {
      Authorization: `Bearer ${API_KEY}`,
    },
  };

  const body = {
    tokens: tokens,
    wallets: wallets,
  };

  console.log("Making API request...");
  try {
    await delay(5000);
    const response = await axios.post(url, body, config);
    return response.data;
  } catch (error) {
    if (error.response && error.response.status === 429) {
      console.error(
        "Rate limit fetching balance exceeded:",
        error.response.data
      );
    } else {
      console.error("Error fetching balance:", error.message);
    }
    return null;
  }
}

console.log("Calling getWalletBalance function...");
const walletBalance = await getWalletBalance();
console.log("Wallet balance:", walletBalance);
```

```
  res.json(walletBalance);
});

app.get("/api/gas-price", async (req, res) => {
  res.set("Cache-Control", "no-store");
  res.set("Cache-Control", "no-store, max-age=0");
  res.set("Pragma", "no-cache");
  res.set("Expires", "0");

  async function fetchGasPrice() {
    const url = "https://api.1inch.dev/gas-price/v1.5/56";
    const config = {
      headers: {
        Authorization: `Bearer ${API_KEY}`,
      },
      params: {},
    };

    try {
      await delay(8000);
      const response = await axios.get(url, config);
      const gasPrice = response.data;
      console.log(gasPrice);
      return gasPrice;
    } catch (error) {
      if (error.response && error.response.status === 429) {
        console.error(
          "Rate limit fetching balance exceeded:",
          error.response.data
        );
      } else {
        console.error("Error fetching balance:", error.message);
```

```
    }
    return null;
  }
 }

 const gasPrice = await fetchGasPrice();
 res.json(gasPrice);
});

app.get("/allowance", async (req, res) => {
 const { tokenAddress, walletAddress } = req.query;

 async function checkAllowance(tokenAddress, walletAddress) {
  const url = "https://api.1inch.dev/swap/v6.0/56/approve/allowance";

  const config = {
   headers: {
    Authorization: `Bearer ${API_KEY}`,
   },
   params: {
    tokenAddress: tokenAddress,
    walletAddress: walletAddress,
   },
  };

  const response = await axios.get(url, config);
  return response.data;
 }

 try {
  const allowance = await checkAllowance(
   req.query.tokenAddress,
```

```
      req.query.walletAddress
    );
    res.json(allowance);
    console.log(allowance);
  } catch (error) {
    if (error.response && error.response.status === 429) {
      console.error("Rate limit while allowing exceeded:", error.response.data);
    } else {
      console.error("Error validating allowance:", error.message);
    }
    res
      .status(500)
      .json({ error: "An error occurred while fetching allowance" });
  }
});

function delay(ms) {
  return new Promise((resolve) => setTimeout(resolve, ms));
}

app.get("/transaction", async (req, res) => {
  const { tokenAddress } = req.query;
  let approvalExecuted = false;

  async function approveTokenForSwap(tokenAddress) {
    const url = "https://api.1inch.dev/swap/v6.0/56/approve/transaction";

    const config = {
      headers: {
        Authorization: `Bearer ${API_KEY}`,
      },
      params: {
```

```
            tokenAddress: tokenAddress,
          },
        };

        const response = await axios.get(url, config);
        console.log(
          `Made request with tokenAddress ${tokenAddress}, received status code ${response.status}`
        );
        return response.data;
      }

      try {
        const transaction = await approveTokenForSwap(tokenAddress);
        res.json(transaction);
        approvalExecuted = true;
      } catch (error) {
        console.error("Error making axios request:", error.message);
        if (error.response && error.response.status === 429) {
          await delay(2000);
          const transaction = await approveTokenForSwap(tokenAddress);
          res.json(transaction);
          approvalExecuted = true;
        } else if (error.response) {
          console.error("Error response data:", error.response.data);
          console.error("Error response status:", error.response.status);
          console.error("Error response headers:", error.response.headers);
          res.status(error.response.status || 500).json({
            error:
              error.response.data ||
              "An error occurred while executing transaction.",
          });
```

```
    } else {
      res
        .status(500)
        .json({ error: "An error occurred while executing transaction." });
    }
  }
});

app.get("/swap", async (req, res) => {
  const { fromToken, toToken, amount, walletAddress, slippage } = req.query;
  let swapExecuted = false;

  async function executeSwap() {
    const url = "https://api.1inch.dev/swap/v6.0/56/swap";

    const config = {
      headers: {
        Authorization: `Bearer ${API_KEY}`,
      },
      params: {
        src: fromToken,
        dst: toToken,
        amount: amount,
        from: walletAddress,
        slippage: slippage,
      },
    };

    try {
      const response = await axios.get(url, config);
      console.log(response.data);
      const formattedResponse = {
```

```
        fromToken: response.data.fromToken,
        toToken: response.data.toToken,
        toAmount: response.data.amount,
        tx: response.data.tx,
      };
      res.json(formattedResponse);
      swapExecuted = true;
    } catch (error) {
      console.error(error);
      if (error.response && error.response.status === 400) {
        const errorData = {
          error: "Bad Request",
          description: error.response.data.description,
          statusCode: error.response.status,
          requestId: error.response.headers["x-request-id"],
          meta: [],
        };
        res.status(400).json(errorData);
      } else if (error.response && error.response.status === 429) {
        await new Promise((resolve) => setTimeout(resolve, 2000));
        return executeSwap();
      } else {

        res.status(500).json({ error: "Failed to execute swap" });
      }
    }
  }
  await new Promise((resolve) => setTimeout(resolve, 2000));
  executeSwap();
});

app.get("/transactions", async (req, res) => {
```

```
  try {
    const events = await Transaction.find();
    res.json(events);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});


app.post("/add-transaction", async (req, res) => {
  try {
    const { number, user, from, to, input, output, status } = req.body;
    const newTransaction = new Transaction({
      number,
      user,
      from,
      to,
      input,
      output,
      status,
    });

    const savedTransaction = await newTransaction.save();
    res.status(201).json(savedTransaction);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

async function startServer() {
  try {
    await mongoose.connect(MONGODB_URL).then(() => {
```

```
      console.log("Connected to MongoDB");
      app.listen(port, () => {
        console.log(`Server running at http://localhost:${port}`);
      });
    });
  } catch (error) {
    console.error("Error connecting to MongoDB:", error);
    process.exit(1);
  }
}

startServer();
```

UNICHECK
by Turnitin

Ім'я користувача:
Інформаційних систем в економіці Шкуратовська Те...

ID перевірки:
1016352800

Дата перевірки:
12.06.2024 22:09:22 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
13.06.2024 11:21:34 EEST

ID користувача:
100005745

Назва документа: Сорока_Д_ІН-404

Кількість сторінок: 54 Кількість слів: 8544 Кількість символів: 70634 Розмір файлу: 1.42 MB ID файлу: 1016156241

7.34%
Схожість

Найбільша схожість: 2.04% з джерелом з Бібліотеки (ID файлу: 1015259415)

2.63% Джерела з Інтернету 232 ..... Сторінка 56

7.23% Джерела з Бібліотеки 391 ..... Сторінка 57

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 6

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА
Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці

ОСВІТНЬО ПРОФЕСІЙНА ПРОГРАМА «КОМП'ЮТЕРНІ НАУКИ»
галузь знань 12 «Інформаційні технології»
спеціальність 122 «Комп'ютерні науки»

Форма навчання: денна

КВАЛІФІКАЦІЙНИЙ БАКАЛАВРСЬКИЙ ПРОЄКТ

на тему:

«Розробка веб-застосунку криптовалютної біржі»

здобувача Сороки Дмитра Сергійовича

Науковий керівник:
к.е.н., доцент
Ситник Н.В.

Проєкт допущений до захисту
перед екзаменаційною комісією з
атестації здобувачів вищої освіти
завідувач кафедри:
к.е.н., доцент.
Тішков Б.О.

Київ 2024