МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Розробка Web3 застосунку на базі блокчейну з застосуванням штучного інтелекту»

Студент 4 курсу групи КН-20к-1	Керівник дипломної роботи
	_____
	(посада керівника)
Орловський Владислав Віталійович	Кириченко Віктор Вікторович
(прізвище, ім'я та по-батькові студента)	(прізвище, ім'я та по-батькові керівника)
	До захисту
	(резолюція «До захисту»)
_____	10.06.2024 _____
(підпис студента)	(дата) (підпис викладача)

Київ, 2024 рік

Скорочення

DeFi (Decentralized Finance) - децентралізована фінансова система.

NFT (Non-Fungible Token) - незамінний токен, термін у криптовалюті.

AI (Artificial Intelligence) - штучний інтелект.

IoT (Internet of Things) - інтернет речей, мережа фізичних об'єктів з датчиками для обміну даними.

UX (User Experience) - користувацький досвід, взаємодія користувача з продуктом.

UI (User Interface) - користувацький інтерфейс, зовнішній вигляд та елементи керування програмою.

ENSIP-11 (Ethereum Name Service Improvement Proposal 11) - Пропозиція покращення служби імен Ethereum номер 11.

API (Application Programming Interface) - програмний інтерфейс прикладного програмування

Зміст

Вступ

У сучасному світі технології розвиваються з надзвичайною швидкістю, і це особливо помітно у сфері інформаційних технологій. Одним з найбільш перспективних напрямків є розвиток Web3 технологій, які обіцяють кардинально змінити спосіб взаємодії користувачів з Інтернетом. Web3 представляє собою нове покоління Інтернету, яке базується на принципах децентралізації, прозорості та безпеки, що дозволяє користувачам контролювати свої дані та активи без посередників.

Одна з ключових переваг Web3 технологій полягає в тому, що вони відкривають нові можливості для автоматизації та інтеграції різних процесів у цифровому середовищі. Це може включати DeFi, які дозволяють здійснювати фінансові операції без участі банків, а також різноманітні смарт-контракти, що забезпечують автоматичне виконання угод при дотриманні певних умов.

Важливою складовою Web3 екосистеми є криптовалютні гаманці, які дозволяють користувачам безпечно зберігати та керувати своїми цифровими активами. Однак, використання таких гаманців супроводжується певними труднощами, такими як складність запам'ятовування seed-фраз, висока вартість транзакцій та необхідність роботи з кількома блокчейн-мережами. Це створює потребу у розробці більш зручних та безпечних рішень для управління криптовалютами.

Проте, впровадження Web3 технологій супроводжується низкою викликів. Серед них – складність розробки та інтеграції децентралізованих систем, необхідність забезпечення високого рівня безпеки та конфіденційності даних, а також потреба в кваліфікованих фахівцях, здатних працювати з новітніми технологіями.

Попри ці виклики, переваги використання Web3 технологій значно переважають. Вони дозволяють створювати більш надійні, безпечні та ефективні системи, що можуть значно покращити якість життя користувачів та

підвищити продуктивність бізнесу. У майбутньому можливості для розвитку цих технологій є практично безмежними, і вони обіцяють забезпечити новий рівень взаємодії в цифровому світі.

РОЗДІЛ 1. Аналіз існуючої інформації щодо гаманців Web3

1.1. Порівняльний аналіз існуючої інформації та рішень

Розвиток Web3 технологій супроводжується появою численних рішень для управління цифровими активами. Найпопулярнішими серед них є гаманці, такі як MetaMask, Trust Wallet та Coinbase Wallet. Ці гаманці забезпечують доступ до DeFi, NFT та інших додатків на базі блокчейн[1].

MetaMask є одним із найбільш популярних Web3 гаманців, що дозволяє користувачам взаємодіяти з децентралізованими додатками безпосередньо через браузер. Основні функції MetaMask включають зберігання та управління криптовалютами, підписання транзакцій і взаємодію з різними блокчейнами. Однак, MetaMask має суттєві недоліки, такі як складність використання для новачків та високі транзакційні витрати в мережі Ethereum.

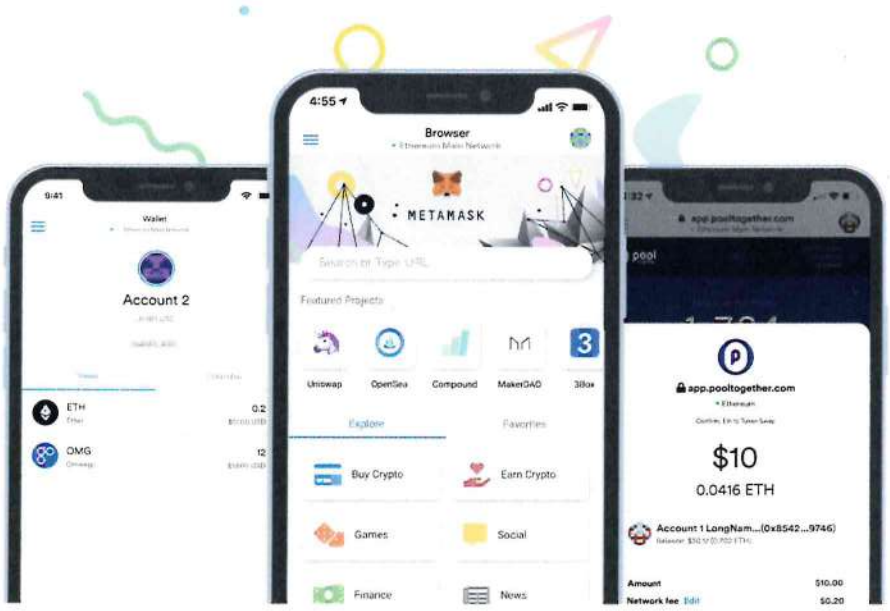

Рис. 1.1. Криптогаманець MetaMask

Trust Wallet є другим по популярності гаманцем і підтримує великий спектр криптовалют і токенів. Він має мобільний додаток для iOS та Android, що робить його зручним для користувачів смартфонів. Trust Wallet також підтримує взаємодію з децентралізованими додатками через вбудований браузер. Основні

недоліки Trust Wallet включають ті ж самі проблеми, що і у MetaMask – необхідність запам'ятовування seed-фраз і приватних ключів, а також високі витрати на транзакції.

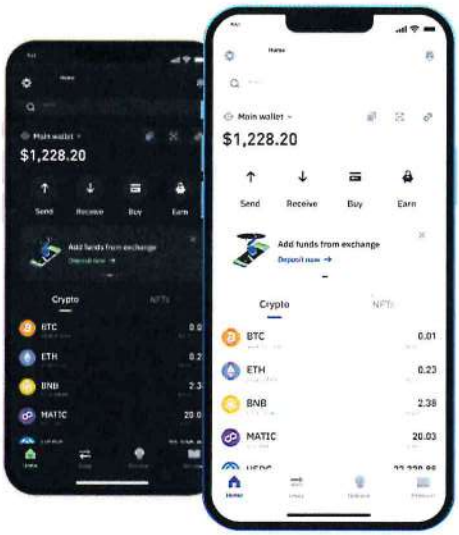

Рис. 1.2. Криптогаманець Trust Wallet

Coinbase Wallet є частиною екосистеми Coinbase і забезпечує зручний інтерфейс для зберігання та управління криптовалютами. Він також підтримує взаємодію з децентралізованими додатками та DeFi протоколами. Незважаючи на зручність використання, Coinbase Wallet також стикається з проблемами безпеки через необхідність зберігання seed-фраз.

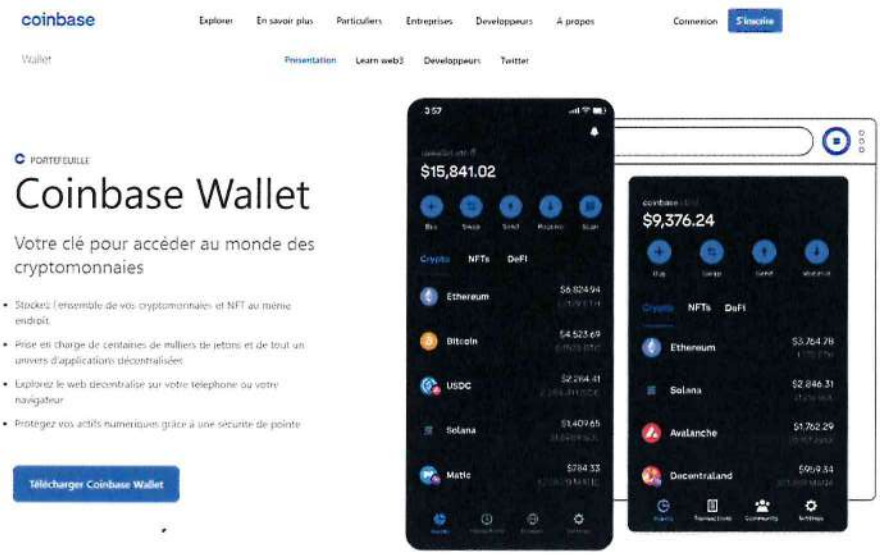

Рис. 1.3. Криптогаманець Coinbase Wallet

Інші гаманці, такі як MyEtherWallet та Argent, також мають свої особливості та обмеження. MyEtherWallet дозволяє користувачам взаємодіяти з Ethereum блокчейном через інтерфейс веб-браузера, але вимагає від користувачів запам'ятовування приватних ключів. Argent, з іншого боку, використовує концепцію "social recovery", що дозволяє користувачам відновлювати доступ до гаманця за допомогою довірених контактів, але цей метод також має свої ризики.

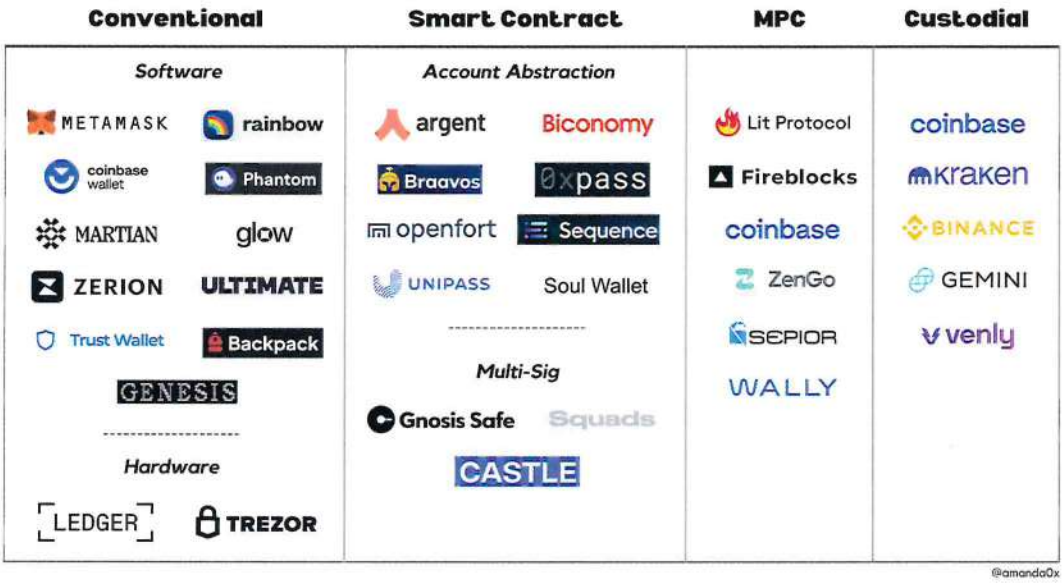

Рис. 1.4. Табличка з різними криптогаманцями

1.2. Відомі проблеми та виклики

Основними проблемами, з якими стикаються користувачі Web3 гаманців, є складність запам'ятовування та зберігання seed-фраз та приватних ключів. Seed-фрази зазвичай складаються з 12-24 слів, які користувач повинен запам'ятати і зберігати у безпечному місці. Втрата цієї інформації призводить до втрати доступу до гаманця і всіх його активів.

Висока вартість транзакцій у мережах, таких як Ethereum, створює бар'єр для використання Web3 гаманців, особливо для дрібних транзакцій. Багато користувачів мають активи на різних блокчейнах, і управління ними може бути

складним через необхідність використовувати різні гаманці та інтерфейси. Це ускладнює процес моніторингу і управління активами.

Недружній користувацький інтерфейс також є значною проблемою. Багато Web3 гаманців мають складний інтерфейс, який відлякує нових користувачів. Для успішного використання гаманця користувач повинен мати певний рівень технічної підготовки і розуміння принципів роботи блокчейнів.

Існують також проблеми, пов'язані з безпекою. Хоча блокчейн технологія сама по собі забезпечує високий рівень безпеки, користувачі можуть стати жертвами фішингових атак, зловмисного програмного забезпечення або втрати приватних ключів. Крім того, зростання популярності DeFi додатків призвело до збільшення кількості атак на смарт-контракти.

1.3. Існуючі рішення та їх обмеження

На ринку існують різні рішення, які намагаються вирішити ці проблеми. Апаратні гаманці, такі як Ledger і Trezor, забезпечують високий рівень безпеки за рахунок зберігання приватних ключів на фізичному пристрої. Вони захищені від хакерських атак, оскільки ключі ніколи не покидають пристрій. Однак, ці гаманці все ще вимагають від користувачів запам'ятовування seed-фраз, що не вирішує проблему їх складності.

Рис. 1.5. Фізичні криптогаманці Ledger та Trezor

Мультипідписні гаманці дозволяють підвищити безпеку, вимагаючи кількох підписів для проведення транзакцій. Це знижує ризик втрати доступу до активів у випадку компрометації одного з ключів. Однак, такі гаманці ускладнюють процес управління, оскільки вимагають координації між кількома користувачами.

Смарт-контракти надають можливість створення гаманців з гнучкими налаштуваннями підписантів та іншими функціями безпеки. Вони можуть автоматично виконувати певні дії при виконанні заданих умов. Однак, такі рішення часто стикаються з проблемами масштабованості та високими витратами на транзакції.

Іншим важливим аспектом є взаємодія користувачів з різними блокчейн-мережами. Наприклад, деякі гаманці підтримують лише одну мережу, що обмежує можливості користувачів у управлінні активами. У той же час, мультичейн гаманці дозволяють працювати з кількома мережами, що значно розширює їх функціонал, але ускладнює процес налаштування та використання.

1.4. Використання технологій штучного інтелекту у Web3

AI відіграє важливу роль у розвитку Web3 технологій. Використання AI дозволяє автоматизувати багато процесів, покращити безпеку та зручність використання гаманців. Наприклад, технології машинного навчання можуть використовуватись для аналізу поведінки користувачів та виявлення підозрілих транзакцій, що дозволяє запобігти шахрайству.

Інший аспект використання AI у Web3 - це автоматизація процесів управління активами. AI може допомагати користувачам приймати рішення щодо інвестування, аналізуючи ринкові тенденції та прогнозуючи зміни у вартості активів. Це дозволяє знизити ризики та підвищити ефективність управління портфелем.

Розвиток технологій AI також відкриває нові можливості для створення інтелектуальних контрактів, які можуть автоматично виконувати певні дії при виконанні заданих умов. Це може включати автоматичне управління активами, виплату дивідендів, виконання фінансових операцій та інші функції, що значно спрощує процеси для користувачів.

Як може змінитися користувацький досвід у майбутньому завдяки інтеграції AI у Web3 гаманці? Можливо, ми побачимо повну автоматизацію багатьох процесів, що дозволить користувачам зосередитись на більш важливих аспектах управління активами та зменшить необхідність у ручному виконанні рутинних завдань.

1.5. Перспективи розвитку та виклики впровадження Web3 гаманців

Впровадження Web3 гаманців супроводжується численними викликами, включаючи технічні, економічні та соціальні аспекти. Одним з головних викликів є забезпечення високого рівня безпеки та конфіденційності даних користувачів. Іншим важливим аспектом є забезпечення зручності використання гаманців для широкої аудиторії.

Технічні виклики включають необхідність забезпечення масштабованості системи, здатність обробляти велику кількість транзакцій без затримок та забезпечення сумісності з різними блокчейнами. Це потребує розробки нових алгоритмів та технологій, що можуть ефективно вирішувати ці завдання.

Економічні виклики включають високу вартість транзакцій у деяких мережах, що може обмежувати використання гаманців для дрібних транзакцій. Крім того, необхідно розробити механізми для зниження цих витрат, щоб зробити використання криптовалют більш доступним для користувачів

Соціальні виклики включають необхідність навчання користувачів новим технологіям та забезпечення їх довіри до системи. Це може вимагати

проведення інформаційних кампаній та розробки зручних та інтуїтивно зрозумілих інтерфейсів, що полегшують процес використання гаманців.

Перспективи розвитку Web3 гаманців включають інтеграцію з новими технологіями, такими як AI та IoT. Це може значно розширити функціональність гаманців та забезпечити нові можливості для користувачів. Наприклад, інтеграція з IoT може дозволити автоматичне виконання фінансових операцій на основі даних, отриманих від різних пристроїв.

Як може змінитися ринок цифрових активів у майбутньому завдяки розвитку Web3 гаманців? Уявімо, що кожен користувач має доступ до простих та безпечних інструментів для управління своїми активами. Це може призвести до значного зростання кількості користувачів криптовалют, підвищення ефективності фінансових операцій та створення нових економічних можливостей.

РОЗДІЛ 2. Проектні і технічні рішення

2.1 Визначення основної ідеї

Основною ідеєю створення Web3 застосунків є децентралізація сервісів та додатків, що дозволяє забезпечити більшу прозорість, безпеку та контроль над даними для користувачів. У Web3 застосунках користувачі мають можливість володіти та контролювати свої дані та активи без необхідності довіряти посередникам, таким як банки або великі корпорації. Це досягається за допомогою блокчейн-технологій, що дозволяють створювати децентралізовані платформи, де інформація зберігається у розподілених реєстрах, що є доступними для всіх учасників мережі[2].

Блокчейн як основа Web3 застосунків забезпечує низку ключових переваг. Перш за все, це підвищена безпека. Децентралізовані системи менш уразливі до хакерських атак, оскільки відсутній єдиний точковий збій. Кожен вузол мережі зберігає копію даних, що забезпечує їхню незмінність та надійність. Наприклад, у випадку атаки на один вузол, решта мережі продовжує функціонувати без збоїв, і дані залишаються недоторканими[3].

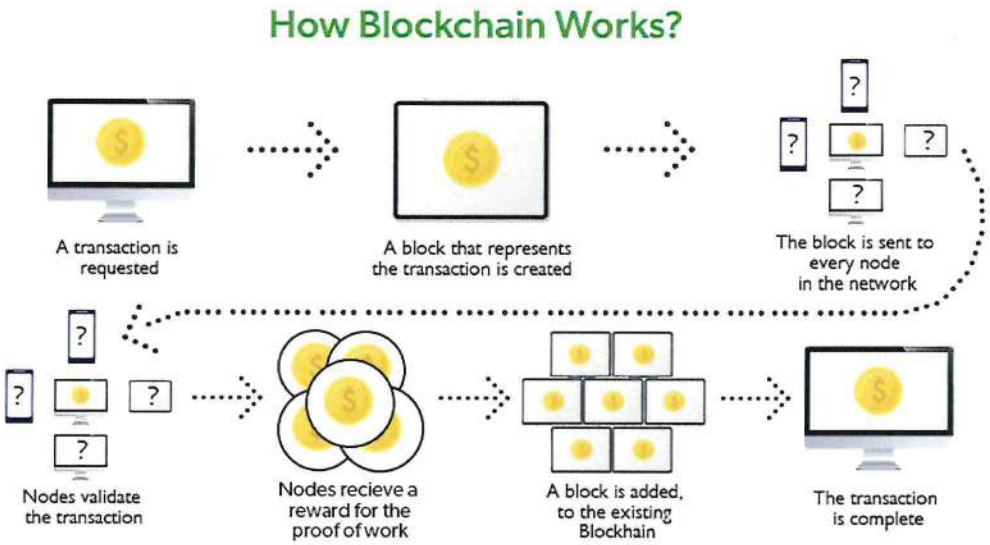

Рис. 2.1. Процес роботи блокчейну

Другим важливим аспектом є прозорість. Всі транзакції та дії в блокчейні є публічними та можуть бути перевірені будь-яким учасником мережі. Це сприяє

підвищенню довіри користувачів до системи та забезпечує можливість незалежного аудиту. Користувачі можуть самостійно перевіряти історію транзакцій, що підвищує рівень довіри до платформ та сервісів.

Контроль над даними є ще однією суттєвою перевагою Web3 застосунків. У традиційних централізованих системах користувачі часто втрачають контроль над своїми даними, коли передають їх третім сторонам. У децентралізованих системах користувачі зберігають контроль над своїми даними, що забезпечує їхню конфіденційність та захист від зловживань. Наприклад, користувачі можуть самостійно керувати своїми криптовалютними гаманцями, не побоюючись втрати доступу через проблеми на стороні постачальника послуг.

Web3 застосунки можуть застосовуватися в різних сферах. У фінансовій галузі додатки DeFi дозволяють користувачам здійснювати транзакції, позики, кредити та інші фінансові операції без посередників. Це знижує витрати та підвищує ефективність, а також забезпечує доступ до фінансових послуг для людей, які не мають доступу до традиційних банківських систем. Наприклад, користувачі можуть отримувати кредити без необхідності проходити складні процедури перевірки, що властиво традиційним банкам.

У сфері мистецтва платформи для створення та торгівлі NFT дозволяють художникам монетизувати свою творчість та захищати свої права. Наприклад, художники можуть створювати цифрові картини та продавати їх як унікальні токени, що підтверджують автентичність та право власності. Це відкриває нові можливості для монетизації творчості та захисту інтелектуальної власності.

Децентралізовані соціальні мережі забезпечують користувачам можливість спілкування без контролю з боку централізованих платформ. Це сприяє свободі слова та захисту особистих даних, оскільки користувачі самі контролюють свої дані та контент. Наприклад, користувачі можуть створювати та публікувати контент без побоювань, що він буде видалений або змінений без їх згоди[4].

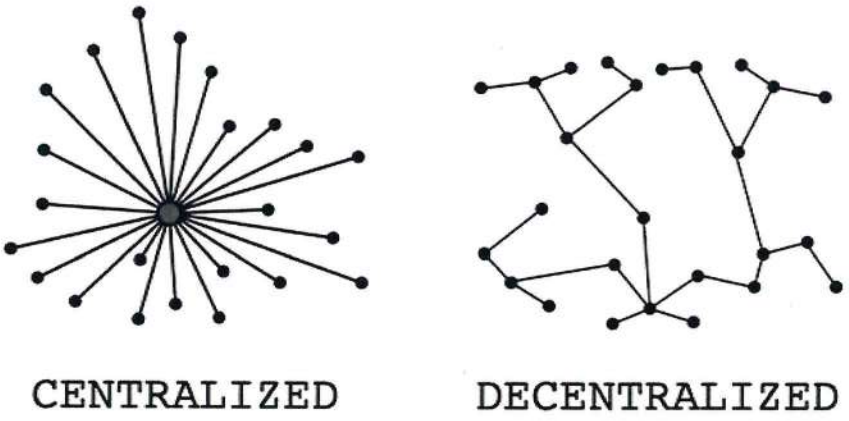

Рис. 2.2. Порівняння централізованої та децентралізованої мереж

2.2 Аналіз ринку та потреб користувачів

Ринок Web3 застосунків стрімко зростає завдяки підвищеному інтересу до криптовалют, DeFi, NFT та токенів. Користувачі прагнуть до більшої приватності, безпеки та контролю над своїми фінансовими активами. Основні потреби користувачів включають безпеку, прозорість, контроль над активами та доступність.

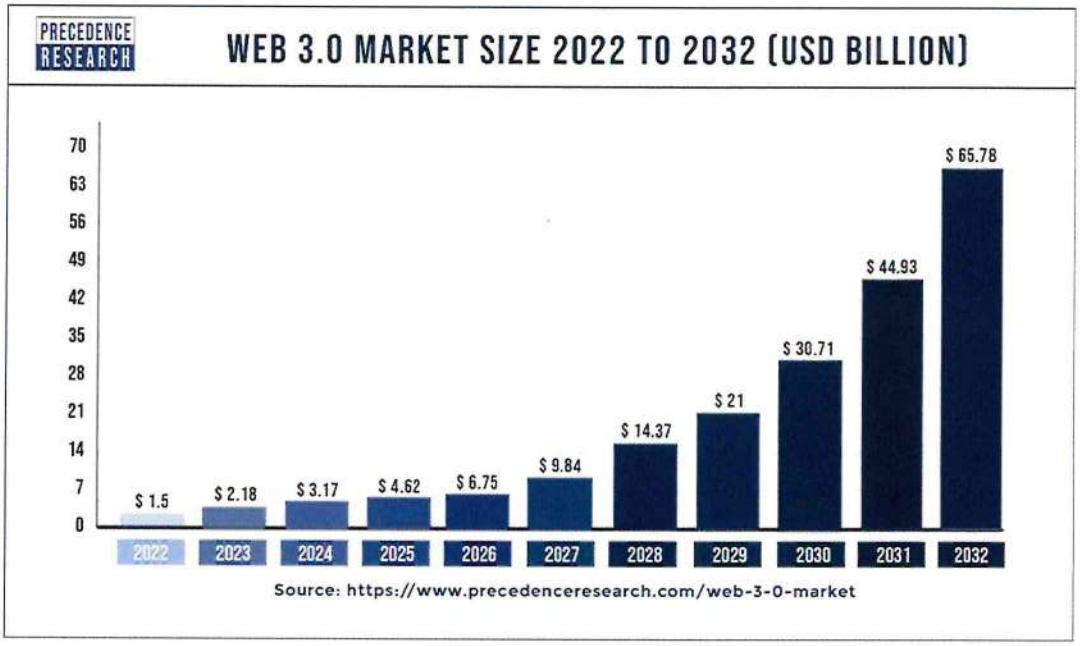

Рис. 2.3. Прогноз зростання ринку Web3 (млрд доларів США)

Безпека є однією з найважливіших потреб користувачів Web3 застосунків. Вони хочуть бути впевненими, що їхні кошти та особисті дані надійно захищені від зловмисників. Це включає використання багатофакторної аутентифікації, шифрування даних та регулярні аудити безпеки . Наприклад, криптовалютні гаманці використовують зашифровані приватні ключі для доступу до активів, що забезпечує високий рівень захисту.

Прозорість також є важливим аспектом для користувачів. Можливість перевірки та аудиту всіх транзакцій підвищує довіру до системи. Користувачі хочуть мати доступ до повної історії транзакцій, щоб мати змогу перевіряти їхню коректність та уникати шахрайства. Це забезпечується використанням розподілених реєстрів, які є публічними та доступними для перевірки.

Контроль над активами є ще однією ключовою потребою користувачів. Вони хочуть мати повний контроль над своїми фінансовими активами без необхідності довіряти третім сторонам. Це означає, що користувачі повинні мати можливість управляти своїми активами самостійно, без посередників, що забезпечує більшу незалежність та знижує ризики втрати коштів через банкрутство або зловживання з боку централізованих організацій.

Доступність також є важливою для користувачів Web3 застосунків. Зручний інтерфейс та простота використання дозволяють широкому колу користувачів взаємодіяти з системою. Користувачі хочуть мати можливість легко користуватися застосунками без необхідності вивчати складні технічні деталі. Це включає створення інтуїтивно зрозумілих інтерфейсів, адаптованих для різних пристроїв та платформ.

Одним з найважливіших типів Web3 застосунків є криптовалютні гаманці, які дозволяють користувачам зберігати, управляти та використовувати свої цифрові активи. Гаманці можуть бути різних типів:

- Апаратні гаманці: Фізичні пристрої, що зберігають приватні ключі в автономному режимі, забезпечуючи високий рівень безпеки. Приклади: Ledger, Trezor.
- Програмні гаманці: Додатки для комп'ютерів або мобільних пристроїв, що зберігають приватні ключі у зашифрованому вигляді на пристрої користувача. Приклади: MetaMask, Trust Wallet.
- Мобільні гаманці: Додатки для смартфонів, що забезпечують зручність використання та доступ до активів у будь-який час. Приклади: Coinbase Wallet, Mycelium.
- Веб-гаманці: Онлайн сервіси, що дозволяють зберігати приватні ключі на віддалених серверах. Вони забезпечують доступ до активів з будь-якого пристрою з підключенням до інтернету. Приклади: Blockchain.com, BitGo.

Ці різновиди гаманців забезпечують користувачам різний рівень безпеки та зручності, що дозволяє вибрати найкращий варіант залежно від їхніх потреб. Наприклад, апаратні гаманці надають максимальний рівень безпеки за рахунок зберігання приватних ключів в автономному режимі, але можуть бути менш зручними у використанні порівняно з програмними або мобільними гаманцями.

2.3 Вибір технологій і інструментів

Розробка Web3 застосунків включає використання різних технологій та інструментів. Вибір конкретних технологій залежить від вимог проекту, обраної блокчейн-платформи та функціональних можливостей, які мають бути реалізовані. Ось деякі з основних технологій та інструментів, які використовуються в розробці Web3 застосунків:

Блокчейн-платформи:

- Ethereum: Одна з найбільш популярних платформ для розробки децентралізованих додатків завдяки своїй гнучкості та підтримці смарт-контрактів. Solidity є основною мовою програмування для розробки

смарт-контрактів на Ethereum. Ethereum підтримує широкий спектр додатків, включаючи DeFi, NFT та DAO. Крім того, Ethereum постійно вдосконалюється, впроваджуючи нові функції та оптимізації, такі як Ethereum 2.0, що покращує масштабованість та продуктивність мережі.

- Optimism: Рішення другого рівня для масштабування Ethereum, яке використовує Optimistic Rollups для зниження транзакційних витрат та підвищення пропускної здатності. Це дозволяє обробляти більше транзакцій за меншу вартість, зберігаючи при цьому безпеку та децентралізацію основної мережі. Optimism сумісний з Ethereum, що дозволяє легко інтегрувати існуючі додатки та використовувати інструменти розробки, створені для Ethereum. Ця платформа особливо корисна для додатків, які потребують високої пропускної здатності та низьких витрат на транзакції.
- Avalanche: Високопродуктивна блокчейн-платформа, яка підтримує смарт-контракти та забезпечує швидкі та дешеві транзакції. Avalanche використовує унікальний консенсусний алгоритм, що дозволяє досягати високої пропускної здатності та низьких затримок. Платформа також підтримує створення індивідуальних блокчейнів, що забезпечує гнучкість та масштабованість. Avalanche активно використовується в різних сферах, включаючи фінансові сервіси, NFT та децентралізовані біржі.

Мови програмування:

- Solidity: Високорівнева мова програмування, що використовується для написання смарт-контрактів на Ethereum. Solidity дозволяє створювати складні логічні структури та інтерактивні додатки на блокчейні. Мова підтримує типізацію та велику кількість вбудованих функцій для роботи з даними. Завдяки своїй гнучкості та потужності, Solidity є основним вибором для розробників, що працюють з Ethereum та іншими сумісними блокчейнами[5].

- Vyper: Альтернатива Solidity, яка має простіший синтаксис та забезпечує більшу безпеку за рахунок зменшення складності коду. Vyper підходить для написання безпечних та надійних смарт-контрактів, зменшуючи ризик вразливостей завдяки обмеженій функціональності. Мова створена для забезпечення високої прозорості та легкості перевірки коду, що робить її привабливою для проектів, що вимагають високих стандартів безпеки.

Фреймворки для розробки:

- Truffle: Один з найпопулярніших фреймворків для розробки, тестування та деплойменту смарт-контрактів на Ethereum. Truffle надає інструменти для автоматизації процесу розробки та забезпечення безперервної інтеграції. Він включає в себе власний засіб для міграції контрактів, налагоджувач та тестовий середовищ. Truffle також має інтеграцію з іншими інструментами, такими як Ganache, що дозволяє створювати локальні блокчейн-мережі для тестування.
- Hardhat: Інший потужний фреймворк для розробки та тестування смарт-контрактів. Hardhat відомий своєю гнучкістю та розширюваністю, що дозволяє розробникам налаштовувати середовище розробки відповідно до своїх потреб. Hardhat підтримує різні плагіни, що розширюють його функціональність, та інтегрується з популярними бібліотеками для тестування, такими як Mocha та Chai. Крім того, Hardhat має потужний інструментарій для налагодження та профілювання смарт-контрактів, що робить його ідеальним вибором для великих та складних проектів[6].
- React: Бібліотека JavaScript для побудови користувацьких інтерфейсів. React дозволяє створювати динамічні та інтуїтивно зрозумілі інтерфейси для взаємодії з користувачами. Вона забезпечує компонентний підхід до розробки, що спрощує повторне використання коду та його підтримку. React використовує віртуальний DOM для оптимізації оновлень

інтерфейсу, що робить застосунки більш швидкими та відзивними[7].

2.4 Архітектура системи

Архітектура Web3 застосунків суттєво відрізняється від традиційних централізованих додатків через децентралізовану природу зберігання даних та обробки транзакцій. Основні компоненти архітектури Web3 застосунку включають фронтенд, бекенд, смарт-контракти та блокчейн.

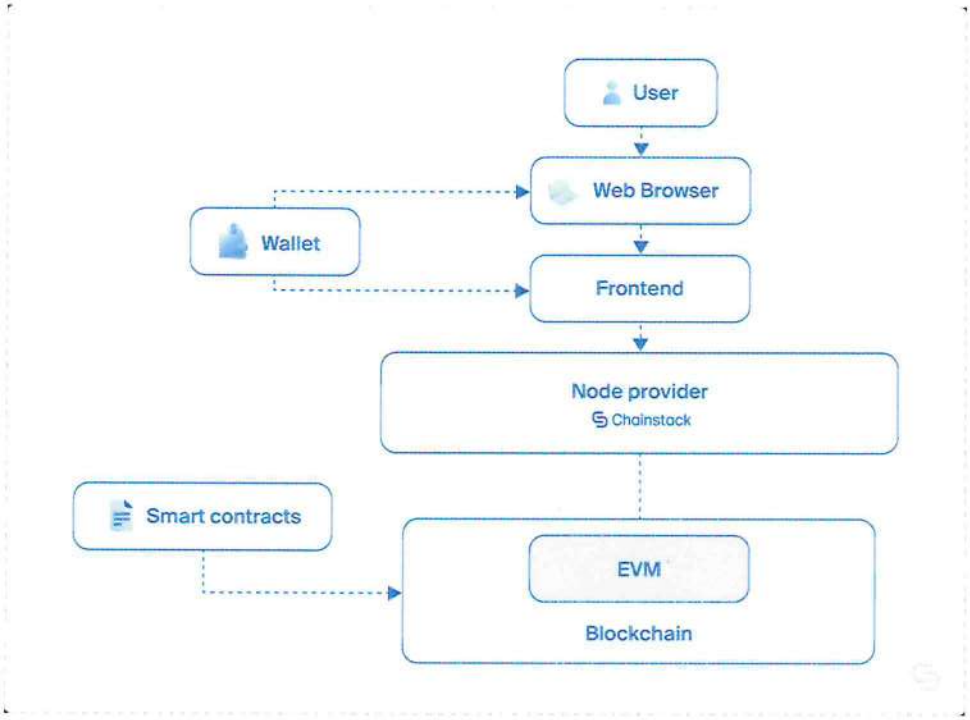

Рис. 2.4. Схема архітектури Web3 застосунку

Фронтенд є інтерфейсом користувача, який безпосередньо взаємодіє з користувачем та відправляє запити до смарт-контрактів. Він створюється за допомогою сучасних JavaScript-фреймворків, таких як React, Angular або Vue.js. Основна роль фронтенду полягає у забезпеченні зручного доступу до функціональності застосунку. Завдяки інтуїтивно зрозумілому інтерфейсу, користувачі можуть легко взаємодіяти з блокчейн-системою, отримуючи необхідні послуги та виконуючи транзакції.

У деяких випадках Web3 застосунки можуть мати бекенд для обробки даних, що не потребують децентралізації. Цей бекенд створюється за

допомогою технологій, таких як Node.js або Express. Він відповідає за обробку запитів до бази даних, керування автентифікацією користувачів та виконання інших завдань, що не вимагають зберігання даних у блокчейні. Наприклад, бекенд може забезпечувати збереження та обробку тимчасових даних, які не мають критичного значення для безпеки системи.

Смарт-контракти є ядром логіки бізнес-процесів у Web3 застосунках. Вони виконуються на блокчейні та забезпечують виконання транзакцій і обробку даних у децентралізованій мережі. Смарт-контракти містять програмний код, який автоматично виконується при виконанні певних умов, що дозволяє автоматизувати та забезпечити безпеку багатьох бізнес-процесів. Наприклад, вони можуть використовуватися для укладання угод, виплати дивідендів або управління активами, гарантуючи виконання умов без участі посередників.

Блокчейн є основою децентралізованої мережі, де зберігаються дані та виконуються смарт-контракти. Він забезпечує безпеку, прозорість та незмінність даних. Всі транзакції записуються в блоки, які з'єднуються в ланцюжок, забезпечуючи незмінність та доступність інформації для всіх учасників мережі. Блокчейн також використовує криптографічні алгоритми для захисту даних та підтвердження транзакцій, що гарантує високу ступінь безпеки та довіри до системи.

Таким чином, архітектура Web3 застосунку поєднує у собі сучасні технології фронтенду, гнучкість бекенду для невеликої обробки даних, силу смарт-контрактів для автоматизації процесів та безпеку децентралізованого блокчейну. Це дозволяє створювати надійні та прозорі системи, що відповідають високим вимогам користувачів щодо приватності, контролю та доступності[8].

2.5 Розробка смарт-контрактів

Смарт-контракти є ключовим елементом Web3 застосунків, оскільки вони забезпечують виконання транзакцій та обробку даних у децентралізованій мережі. Розробка смарт-контрактів вимагає особливої уваги до безпеки та оптимізації, оскільки будь-які помилки в коді можуть призвести до значних фінансових втрат[9].

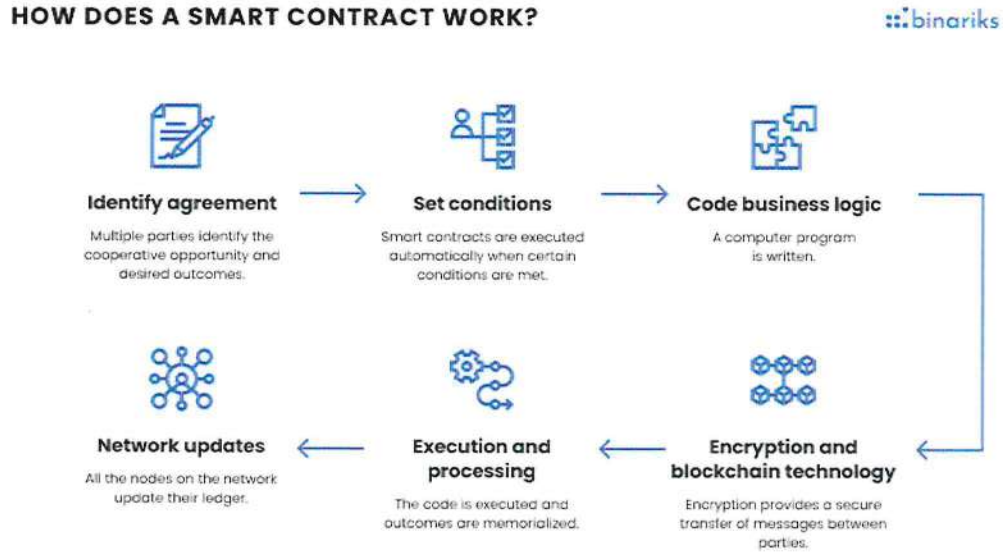

Рис. 2.5. Процес роботи смарт-контракту

Основні етапи розробки смарт-контрактів включають проектування, написання коду, тестування, аудит та деплоймент. На етапі проектування визначаються вимоги до смарт-контрактів та розробляється їх архітектура. Важливо врахувати всі можливі сценарії використання та потенційні загрози безпеці. Наприклад, смарт-контракти для фінансових додатків повинні забезпечувати надійний захист коштів користувачів та виключати можливість несанкціонованого доступу до них.

Написання коду смарт-контрактів здійснюється за допомогою мов програмування, таких як Solidity або Vyper. Код має бути написаний з урахуванням найкращих практик безпеки та оптимізації. Наприклад, необхідно

уникати використання небезпечних конструкцій та забезпечувати захист від відомих вразливостей, таких як “reentrancy attack” або “integer overflow”.

Тестування є важливим етапом розробки смарт-контрактів, оскільки воно дозволяє виявити та виправити помилки до деплойменту на блокчейн. Проведення всебічного тестування включає юніт-тести, інтеграційні тести та тестування безпеки. Це допомагає забезпечити надійність та коректність роботи смарт-контрактів у реальних умовах.

Аудит смарт-контрактів проводиться для виявлення потенційних вразливостей та забезпечення відповідності вимогам безпеки. Аудит здійснюється незалежними експертами або за допомогою AI, які перевіряють код та надають рекомендації щодо його покращення. Це дозволяє знизити ризики та підвищити довіру користувачів до застосунку.

Деплоймент смарт-контрактів включає їх розгортання на блокчейн та налаштування необхідних параметрів. Після деплойменту смарт-контракти стають незмінними, тому важливо забезпечити їхню безпеку та надійність до цього етапу. Наприклад, можна використовувати патерн проксі-контрактів, що дозволяє оновлювати логіку смарт-контрактів без зміни їх адреси на блокчейні.

2.6 Інтеграція штучного інтелекту

Інтеграція AI в Web3 застосунки може значно підвищити їхню функціональність та ефективність. AI може використовуватися для автоматизації процесів, аналізу даних, підвищення безпеки та створення більш інтелектуальних користувацьких інтерфейсів.

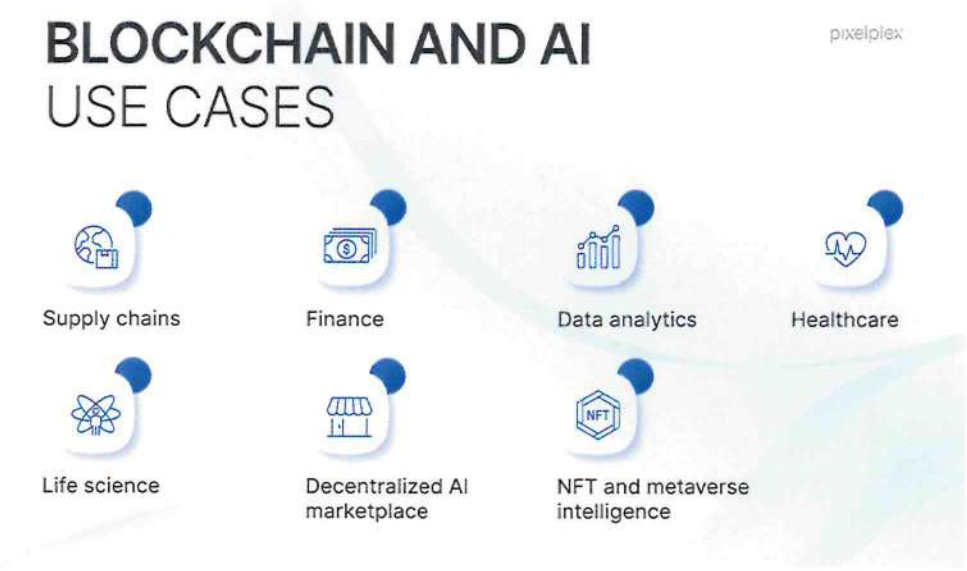

Рис. 2.7. Використання AI: приклади застосування

Основні напрями використання AI в Web3 застосунках включають автоматизацію процесів, аналіз даних, підвищення безпеки та інтелектуальні інтерфейси. Використання AI для автоматизації рутинних завдань, таких як обробка транзакцій, верифікація даних та управління користувацькими акаунтами, допомагає знизити витрати та підвищити ефективність. Наприклад, AI може автоматично обробляти транзакції, перевіряти їх на коректність та виявляти підозрілі дії.

Аналіз даних за допомогою машинного навчання та алгоритмів AI дозволяє обробляти великі обсяги даних, збережених на блокчейні. Це може допомогти виявити закономірності, тренди та аномалії, що можуть бути корисними для прийняття рішень. Наприклад, AI може аналізувати транзакції на блокчейні та виявляти потенційні шахрайські дії.

Підвищення безпеки є ще одним важливим напрямом використання AI. Алгоритми машинного навчання можуть аналізувати поведінку користувачів та виявляти підозрілі дії, що можуть свідчити про спробу шахрайства або атаки на мережу. Це дозволяє вчасно виявляти та запобігати загрозам, підвищуючи загальний рівень безпеки застосунків.

Інтелектуальні інтерфейси, створені за допомогою AI, можуть забезпечити більш інтуїтивні та персоналізовані взаємодії з користувачами. Наприклад, AI може аналізувати поведінку користувачів та надавати рекомендації або автоматично налаштовувати інтерфейс під потреби конкретного користувача. Це допомагає створювати більш зручні та ефективні користувацькі інтерфейси, що сприяє підвищенню задоволеності користувачів.

Для інтеграції AI в Web3 застосунки можна використовувати різні інструменти та платформи, такі як TensorFlow, PyTorch, OpenAI та інші. Важливо забезпечити сумісність між AI-моделями та децентралізованими системами, що може вимагати додаткової розробки та налаштування.

2.7 Забезпечення безпеки

Безпека є одним з найважливіших аспектів розробки Web3 застосунків. Децентралізовані системи мають свої унікальні загрози та виклики, що вимагають спеціальних методів захисту. Основні загрози безпеці Web3 застосунків включають:

Вразливості смарт-контрактів:

- Помилки в коді смарт-контрактів можуть призвести до втрати коштів або інших активів. Для запобігання цьому необхідно проводити всебічне тестування та аудит смарт-контрактів. Однією з найбільших проблем у безпеці смарт-контрактів є можливість reentrancy-атак, коли зловмисник може багаторазово викликати функцію контракту до завершення першого виклику, що може призвести до втрати коштів. Для захисту від таких атак слід використовувати патерни програмування, що забезпечують безпеку, такі як замикання стану контракту на час виконання критичних операцій.

Атаки на мережу:

- DDoS-атаки та інші види атак на блокчейн-мережу можуть призвести до збоїв у роботі застосунків. Для захисту від таких атак необхідно

використовувати захищені інфраструктури та протоколи. Наприклад, для захисту від DDoS-атак можна використовувати розподілені мережеві фільтри, що блокують підозрілу активність, або спеціалізовані сервіси для захисту від DDoS-атак, що забезпечують високу стійкість мережі до таких загроз.

Шахрайство та соціальна інженерія:

- Користувачі можуть стати жертвами фішингових атак або інших видів шахрайства. Важливо забезпечити освітню роботу серед користувачів та надавати інструменти для захисту від таких загроз. Наприклад, можна використовувати двофакторну аутентифікацію для захисту облікових записів користувачів, а також впроваджувати механізми верифікації транзакцій, що допомагають виявляти підозрілі операції.

Рис. 2.8. Основні ризики Web3

Основні методи забезпечення безпеки включають:

- Аудит смарт-контрактів: Залучення незалежних експертів або AI для перевірки коду смарт-контрактів та виявлення потенційних вразливостей. Аудит допомагає забезпечити відповідність кодексу вимогам безпеки та знизити ризики.

- Використання криптографічних протоколів: Застосування сучасних криптографічних алгоритмів для шифрування даних та забезпечення безпечної передачі інформації. Це допомагає захистити дані від несанкціонованого доступу та змін.
- Захист від DDoS-атак: Використання спеціалізованих рішень для запобігання DDoS-атакам та забезпечення стабільної роботи мережі. Це може включати використання мережевих фільтрів, захищених серверів та інших методів.
- Освітні програми для користувачів: Проведення навчальних заходів та розробка матеріалів для підвищення обізнаності користувачів про можливі загрози та методи захисту. Це допомагає знизити ризик шахрайства та інших видів атак.

2.8 Тестування та налагодження

Тестування та налагодження є важливими етапами розробки Web3 застосунків, оскільки вони дозволяють виявити та виправити помилки до релізу продукту. Юніт-тестування перевіряє окремі компоненти застосунку, щоб гарантувати їх коректну роботу, виявляючи помилки на ранніх етапах розробки та забезпечуючи стабільність коду. Наприклад, юніт-тести можуть перевіряти коректність функцій смарт-контрактів, таких як передача токенів або обробка транзакцій. Інтеграційне тестування, в свою чергу, перевіряє взаємодію між різними компонентами системи, дозволяючи виявити проблеми, що виникають при об'єднанні різних модулів. Це тестування гарантує, що фронтенд правильно взаємодіє зі смарт-контрактами та бекендом, забезпечуючи коректну роботу всієї системи в цілому.

Тестування безпеки є критичним для виявлення вразливостей та потенційних загроз. Воно включає проведення пентестів (penetration tests) та інших видів тестування безпеки, які дозволяють пентестерам спробувати зламати смарт-контракти або знайти вразливості у фронтенді. Це допомагає

виявити та виправити потенційні загрози до релізу продукту. Тестування продуктивності перевіряє систему на здатність витримувати високі навантаження та забезпечувати стабільну роботу під час пікових навантажень. Це дозволяє виявити вузькі місця та оптимізувати продуктивність. Наприклад, стрес-тестування може симулювати високе навантаження на блокчейн-систему для перевірки її стійкості до перевантажень.

Для проведення тестування можна використовувати різні інструменти та фреймворки, такі як Truffle, Hardhat, Mocha, Chai та інші. Важливо забезпечити автоматизацію тестування, що дозволяє швидко виявляти та виправляти помилки на всіх етапах розробки. Налагодження включає процес виправлення виявлених помилок та оптимізації коду. Для налагодження можна використовувати дебагери, логери та інші інструменти, що дозволяють аналізувати роботу застосунку та виявляти причини помилок. Наприклад, дебагери дозволяють розробникам крок за кроком аналізувати виконання коду та виявляти місця, де виникають помилки або непередбачувана поведінка.

2.9 Взаємодія з користувачами та UX/UI дизайн

UX/UI є важливими аспектами розробки Web3 застосунків, оскільки вони визначають, наскільки зручно та інтуїтивно зрозуміло користувачам взаємодіяти з системою. Основні принципи UX/UI дизайну для Web3 застосунків включають інтуїтивність, прозорість, безпеку та доступність.

Інтуїтивність передбачає, що інтерфейс має бути зрозумілим та простим у використанні, навіть для користувачів, які не мають технічного досвіду. Кнопки та елементи управління повинні бути логічно розташовані, а інформація доступною для розуміння. Прозорість означає, що користувачі мають отримувати чітку інформацію про свої дії та наслідки цих дій. Це особливо важливо в фінансових додатках, де кожна транзакція може мати значні наслідки. Наприклад, користувачі повинні отримувати підтвердження перед виконанням важливих дій, таких як переведення коштів або зміна налаштувань безпеки.

Безпека в UX/UI дизайні передбачає захист особистих даних та фінансових активів користувачів. Це включає використання двофакторної аутентифікації, шифрування даних та інших методів захисту. Наприклад, інтерфейс може вимагати підтвердження особистості користувача перед виконанням важливих дій. Доступність інтерфейсу має забезпечувати зручність використання для широкого кола користувачів, включаючи людей з обмеженими можливостями. Це включає підтримку різних мов, адаптацію для людей з вадами зору та іншими потребами. Використання великих шрифтів та контрастних кольорів допомагає забезпечити зручність використання для всіх користувачів.

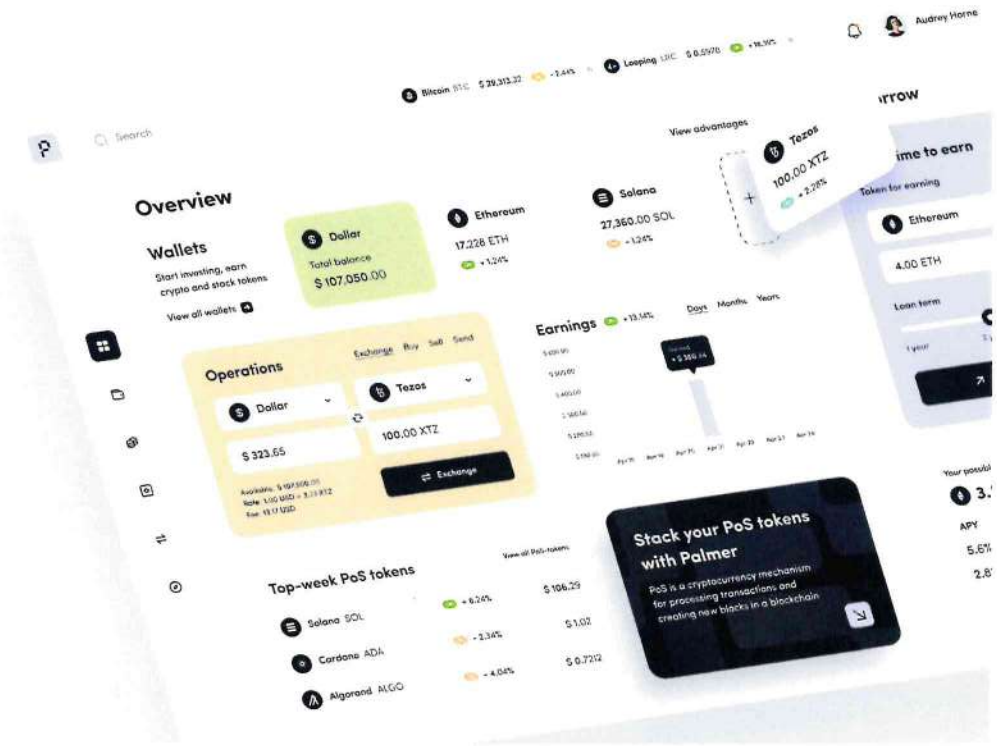

Рис. 2.9. Приклад UX/UI дизайну для Web3 застосунків

Для розробки UX/UI дизайну можна використовувати різні інструменти, такі як Figma, Sketch, Adobe XD та інші. Важливо проводити тестування інтерфейсу з користувачами на різних етапах розробки, щоб виявити можливі проблеми та покращити користувацький досвід.

РОЗДІЛ 3. Процес створення та розробка Web3 застосунку InWallet

3.1 Концепція та планування проекту

Проект InWallet розпочався з концептуальної фази, де було визначено основну мету та задачі. Основна ідея полягала у створенні децентралізованого гаманця, який би спрощував процес взаємодії з криптовалютами, забезпечуючи при цьому високий рівень безпеки та зручності. Основні проблеми, які планувалося вирішити, включали складність використання seed-фраз, високу вартість транзакцій та потребу в мульти-чейн функціоналі.

На початковому етапі був проведений ринковий аналіз, який включав дослідження існуючих рішень на ринку криптовалютних гаманців. Аналіз охоплював такі популярні гаманці, як MetaMask, Trust Wallet та Coinbase Wallet. Виявлені проблеми та обмеження цих гаманців допомогли визначити пріоритети для InWallet. Зокрема, потребувалося більш зручний та безпечний спосіб управління криптовалютами, а також у можливості роботи з кількома блокчейн-мережами.

Було визначено, що користувачі цінують простоту використання та безпеку більше, ніж інші функції. Це стало основою для розробки концепції InWallet, яка включала:

- Безпека: Забезпечення високого рівня безпеки за рахунок використання біометричної автентифікації та Passkey.
- Простота використання: Мінімізація необхідності запам'ятовувати складні паролі та seed-фрази.
- Мульти-чейн функціонал: Підтримка кількох блокчейн-мереж для зручного управління активами.

Планування проекту включало кілька етапів, кожен з яких мав свої завдання та терміни. Першим кроком було створення детального плану проекту, що включав:

- Розробка технічного завдання: Визначення основних функцій та вимог до системи.
- Вибір технологічного стеку: Вибір інструментів та технологій, які будуть використовуватися для розробки.
- Встановлення термінів: Визначення часових рамок для кожного етапу проекту.

Було вирішено використовувати сучасний стек технологій, який включав Next.js та React для фронтенду, Foundry та Solidity для розробки смарт-контрактів, а також Wagmi та Viem для взаємодії з блокчейном[10].

На цьому етапі також було розроблено перші прототипи інтерфейсу користувача. Вони допомогли визначити, як користувачі взаємодіятимуть із системою, і які елементи інтерфейсу потребують покращення. Прототипи були протестовані з декількома людьми, що дозволило виявити та виправити проблеми на ранніх етапах розробки.

3.2 Архітектура та дизайн системи

Розробка InWallet почалася з проектування архітектури системи. Основною вимогою було забезпечення підтримки мульти-чейн функціоналу, що дозволяє користувачам керувати своїми активами в різних блокчейн-мережах. Було вирішено використовувати архітектуру, яка включає такі компоненти: фронтенд, бекенд, смарт-контракти та блокчейн.

Фронтенд є інтерфейсом користувача і був створений за допомогою Next.js. Основна мета фронтенду – забезпечити зручний доступ до функціональності застосунку. Для створення інтерфейсу використовувалися бібліотеки Tailwind CSS та shadcn/ui, що дозволяють створювати сучасні та зручні інтерфейси. Крім того, інтеграція з Wagmi та Viem забезпечила безшовну взаємодію з блокчейном. На етапі проектування інтерфейсу було створено декілька прототипів, які включали базові функції гаманця, такі як перегляд балансу, відправка та отримання криптовалюти. Наприклад, один з перших макетів

інтерфейсу включав в себе ідею, як користувачі можуть бачити свій загальний баланс та баланси в окремих мережах. Цей макет допоміг зрозуміти, які елементи інтерфейсу потребують покращення для забезпечення кращого користувацького досвіду[11].

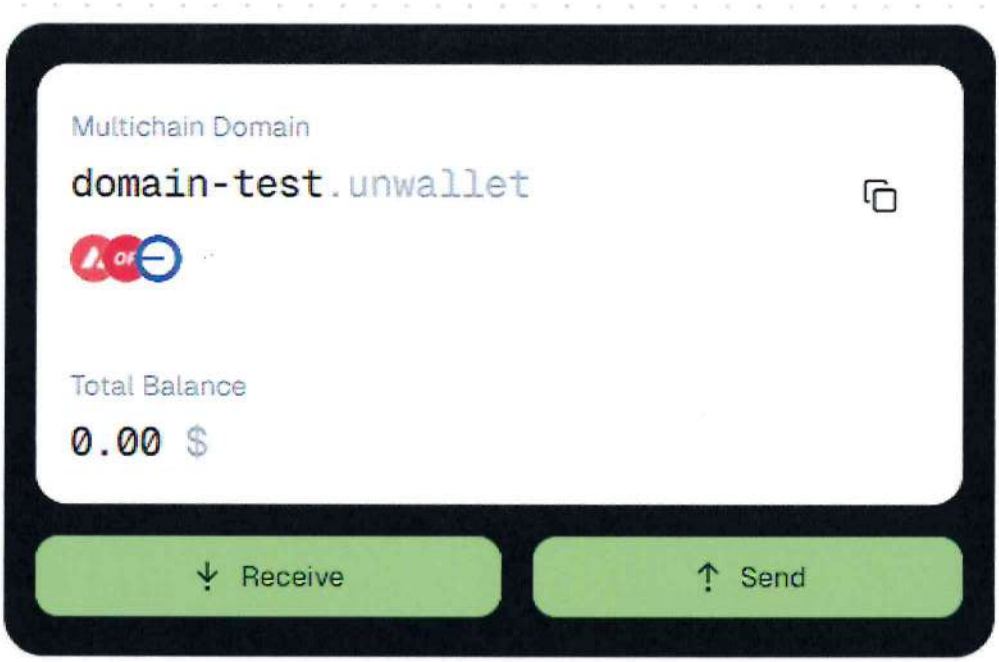

Рис. 3.1. Макет інтерфейсу гаманця InWallet

Після проведення тестувань прототипів були внесені відповідні зміни до дизайну інтерфейсу. Було додано нові функції, такі як можливість перегляду детальної інформації про транзакції та налаштування профілю користувача. Це зробило інтерфейс більш зручним та інтуїтивно зрозумілим.

Для обробки даних, що не потребують децентралізації, був створений бекенд на базі Node.js з Express. Бекенд відповідає за обробку запитів до бази даних, керування автентифікацією користувачів та виконання інших завдань, які не вимагають зберігання даних у блокчейні. Це дозволяє розвантажити блокчейн від зайвих транзакцій та знизити витрати на газ. Основними задачами бекенду було створення API для взаємодії з фронтендом та забезпечення безпеки даних користувачів. Було реалізовано кілька маршрутів для обробки запитів на автентифікацію, отримання балансу та історії транзакцій. Одним із

ключових маршрутів було забезпечення захисту даних користувачів через шифрування та використання токенів доступу[12].

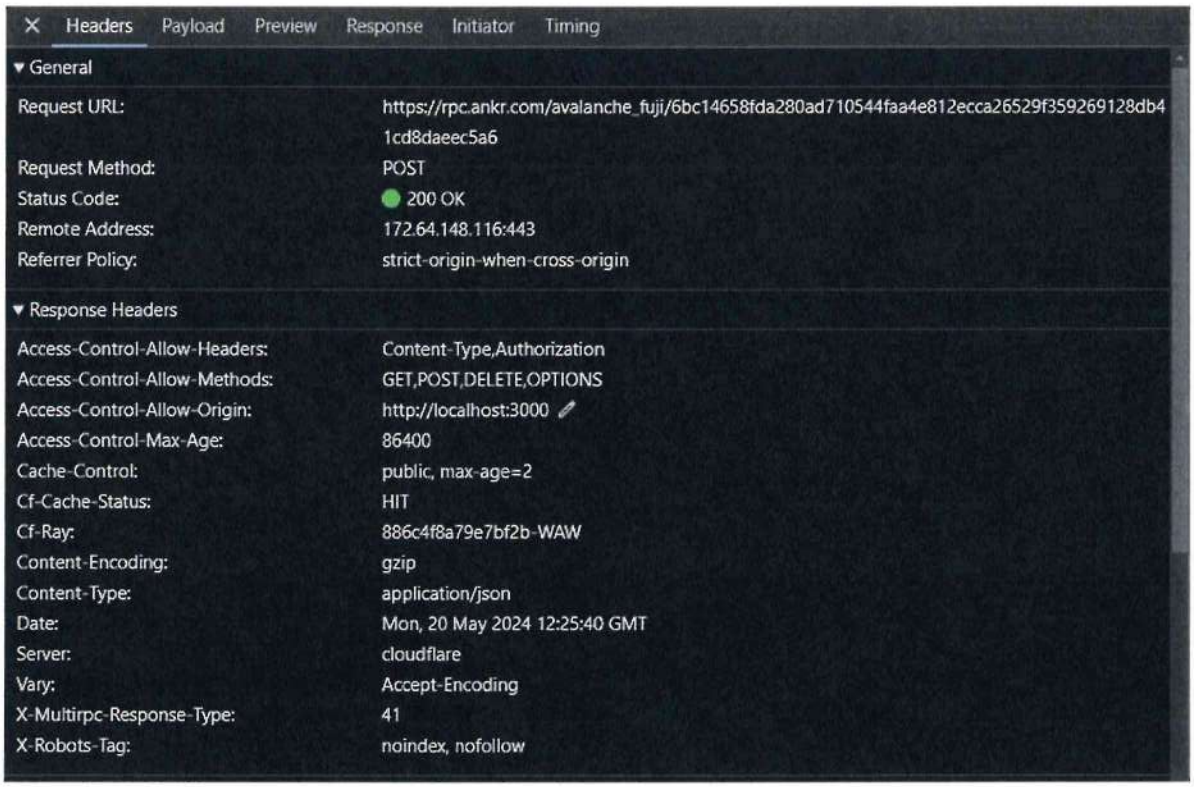

Рис. 3.2. Запит до API для отримання балансу користувача

Для інтеграції з базою даних було обрано KV від Vercel як основну базу даних для зберігання інформації про користувачів та їх транзакції. Це рішення було прийнято через високу продуктивність та масштабованість KV[13].

Смарт-контракти є основою бізнес-логіки InWallet. Вони були написані мовою Solidity з використання Foundry для автоматизації процесу розробки. Основні функції смарт-контрактів включають управління доменами, створення смарт-гаманців та взаємодію між різними блокчейн-мережами. Велику увагу приділяли безпеці смарт-контрактів, щоб уникнути можливих вразливостей та забезпечити надійність системи. Розробка смарт-контрактів почалася з визначення основних вимог та функцій. Спочатку використовувалися шаблони контрактів, які забезпечували базову функціональність. Потім ці шаблони були адаптовані під потреби проекту. Було створено кілька основних

смарт-контрактів для реєстрації доменів, управління гаманцями та обробки транзакцій[14].

```solidity
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

import "../utils/CCIPReceiverBase.sol";
import "./ens-original/FIFSRegistrar.sol";

/**
 * The CCIP enabled ENS registry contract for Hub side.
 */
contract FIFSRegistrarCCIP is FIFSRegistrar, CCIPReceiverBase {
    constructor(ENS ensAddr, bytes32 node, address _router) FIFSRegistrar(ensAddr, node) CCIPReceiverBase(_router) {}

    function _executeFunction(bytes4 func, bytes memory params) internal override {
        if (func == this.register.selector) {
            (bytes32 label, address owner) = abi.decode(params, (bytes32, address));
            register(label, owner);
        } else {
            revert("Unknown function selector");
        }
    }

    function _msgSender() internal view virtual override(Context, CCIPReceiverBase) returns (address) {
        return CCIPReceiverBase._msgSender();
    }

    function isCCIPWhitelisted(uint64 sourceChainSelector, address sender) public pure override returns (bool) {
        return true;
    }
}
```

Рис. 3.2. Смарт-контракт реєстрації домену

Особлива увага приділялася безпеці смарт-контрактів. Було використано найкращі практики програмування, такі як модульне тестування, статичний аналіз коду та аудит безпеки. Для забезпечення надійності системи проводилися зовнішні аудити коду за допомогою AI.

Для забезпечення децентралізованої природи застосунку використовувалися кілька блокчейн-мереж, включаючи Ethereum, Avalanche Fuji, Base Sepolia та Optimism Sepolia. Це дозволяє користувачам керувати своїми активами в різних мережах з одного інтерфейсу. Було використано стандарти, такі як ENSIP-11, для забезпечення єдиного іменування та адресації в різних мережах. Інтеграція мульти-чейн функціоналу вимагала створення спеціальних смарт-контрактів для кожної підтримуваної мережі. Основна задача полягала у забезпеченні безперебійної взаємодії між різними мережами. Для

цього використовувалися інструменти, які дозволяють автоматично оновлювати адреси гаманців та забезпечувати єдине іменування в усіх мережах.

Для підтримки різних блокчейн-мереж було розроблено систему автоматичного оновлення адрес гаманців та їх конфігурацій. Це дозволило користувачам з легкістю перемикатися між різними мережами та керувати своїми активами.

3.3 Розробка та впровадження функціоналу гаманця

Процес реєстрації та налаштування доменів є одним із найважливіших етапів для користувачів. Було створено систему, яка дозволяє користувачам вибирати доменні імена або генерувати їх за допомогою AI. Використання OpenAI API дозволяє автоматично генерувати креативні ідеї для доменів, що значно спрощує процес вибору для користувачів[15].

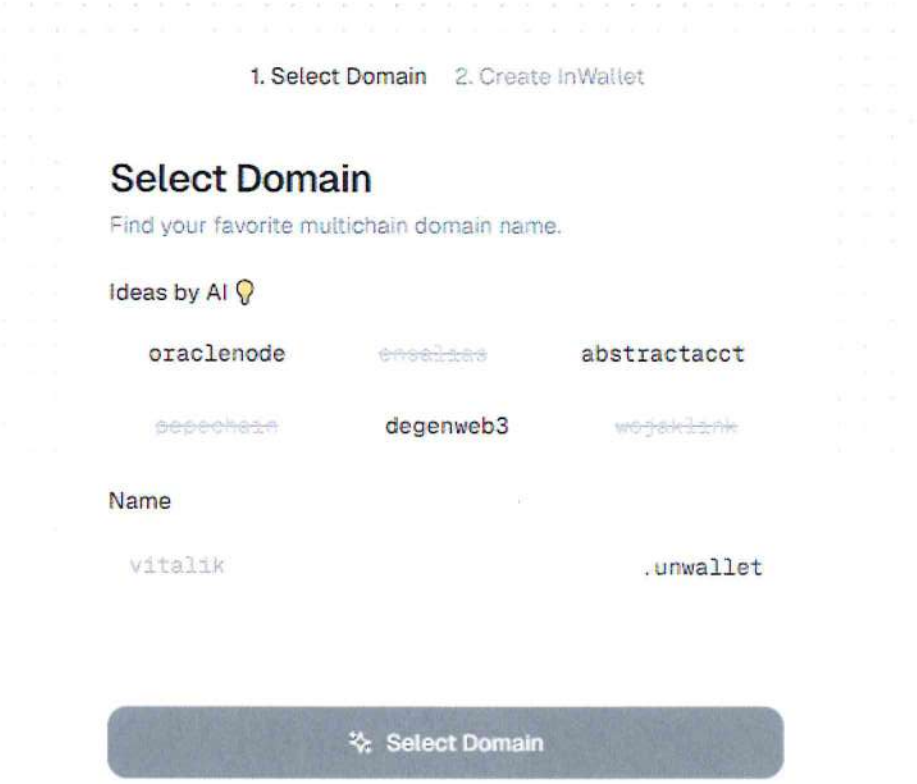

Рис. 3.3. Процес вибору доменного імені за допомогою штучного інтелекту

Користувач може вибрати доменне ім'я або отримати його автоматично за допомогою OpenAI. Запит на генерацію домену надсилається через API, де використовується спеціальний промпт для генерації імен. Після отримання результату домен автоматично реєструється в системі та налаштовується для мульти-чейн використання. Після реєстрації домену користувачі можуть переглядати та копіювати свій домен через інтуїтивно зрозумілий інтерфейс.

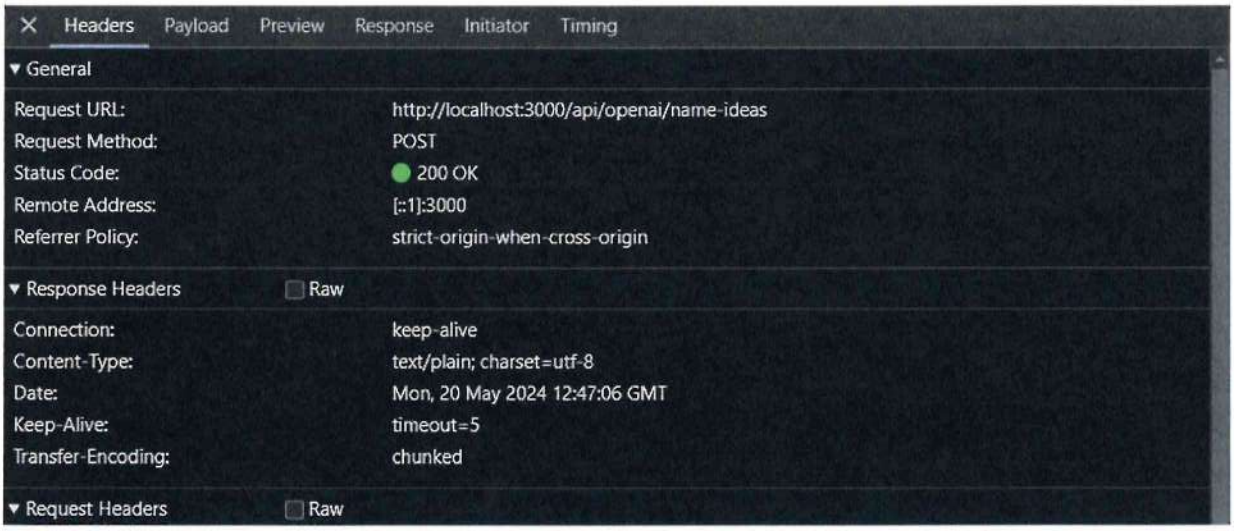

Рис. 3.4. Запит до API для генерації доменного імені за допомогою OpenAI

InWallet дозволяє користувачам створювати та управляти смарт-гаманцем в різних блокчейн-мережах. Гаманець має унікальну адресу, яка автоматично налаштовується для роботи з усіма підтримуваними мережами. Процес створення гаманців включає генерування ключів, реєстрацію адреси в системі та налаштування мульти-чейн функціоналу. Для цього використовуються спеціальні смарт-контракти, які забезпечують автоматичне оновлення адреси та її управлінням.

Після проходження реєстрації користувач має можливість переглядати свій гаманець у кожній з доступних мереж, легко копіювати адресу гаманця, а також переходити в блокчейн-сканер, який динамічно змінюється в залежності від вибраної мережі. Наприклад, користувач може переглядати свій баланс в Base Sepolia, Optimism Sepolia, та Avalanche Fuji. Ця функціональність значно

спрощує управління активами, оскільки дозволяє зосередити всі операції в одному зручному інтерфейсі.

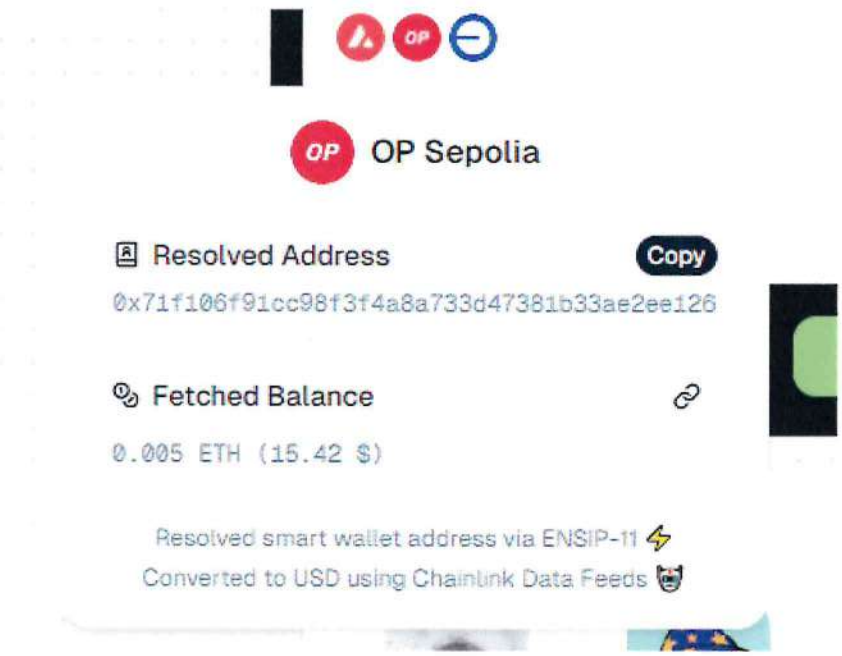

Рис. 3.5. Інтерфейс InWallet для перегляду балансу та адреси в Optimism Sepolia

Крім того, інтеграція блокчейн-сканерів дозволяє користувачам легко відстежувати транзакції та перевіряти їх статус у реальному часі. Це забезпечує додатковий рівень прозорості та контролю, оскільки користувачі можуть швидко переходити до відповідного блокчейн-сканера з інтерфейсу InWallet і переглядати всі деталі своїх транзакцій.

3.4 Інтеграція біометричної автентифікації та Passkey

Однією з ключових особливостей InWallet є використання біометричної автентифікації з технологією Passkey. Це дозволяє користувачам входити в систему без необхідності запам'ятовувати складні паролі або seed-фрази, що значно підвищує зручність використання. Інтеграція Passkey вимагала додаткової розробки для забезпечення сумісності з існуючими протоколами автентифікації. Було створено систему, яка дозволяє користувачам реєструвати свої біометричні дані та використовувати їх для входу в гаманець. Це забезпечує

високий рівень безпеки, оскільки зменшує ризик фішингових атак та втрати доступу до акаунтів[16].

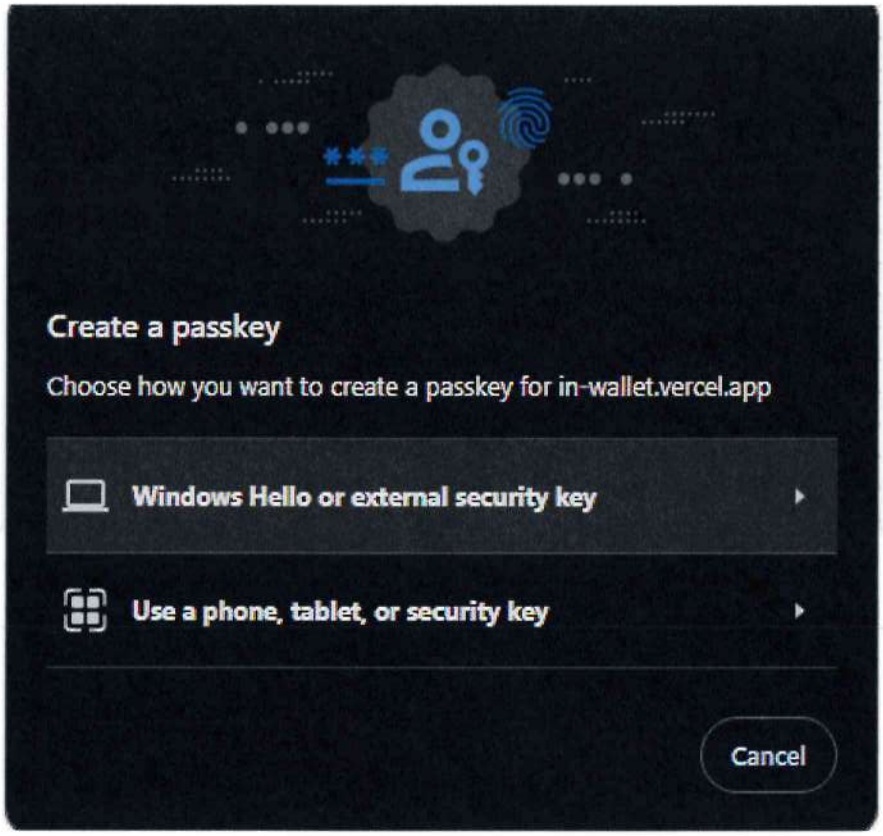

Рис. 3.6. Створення ключа доступу (Passkey)

Процес інтеграції Passkey почався з аналізу існуючих рішень для біометричної автентифікації та вибору найбільш підходящої технології. Було вирішено використовувати Turnkey, яка забезпечує високу надійність та сумісність з іншими компонентами системи. Далі був розроблений механізм реєстрації біометричних даних користувачів. Користувач може зареєструвати свої біометричні дані, такі як відбиток пальця або розпізнавання обличчя, які зберігаються в безпечному середовищі. Під час входу в систему ці дані перевіряються та використовуються для автентифікації, що значно спрощує процес входу та підвищує рівень безпеки[17].

Для завершення інтеграції біометричної автентифікації було проведено всебічне тестування системи. Це включало функціональні тести для перевірки правильності роботи всіх компонентів, роботу Passkey на різних платформах та пристроях, таких як iOS та Android.

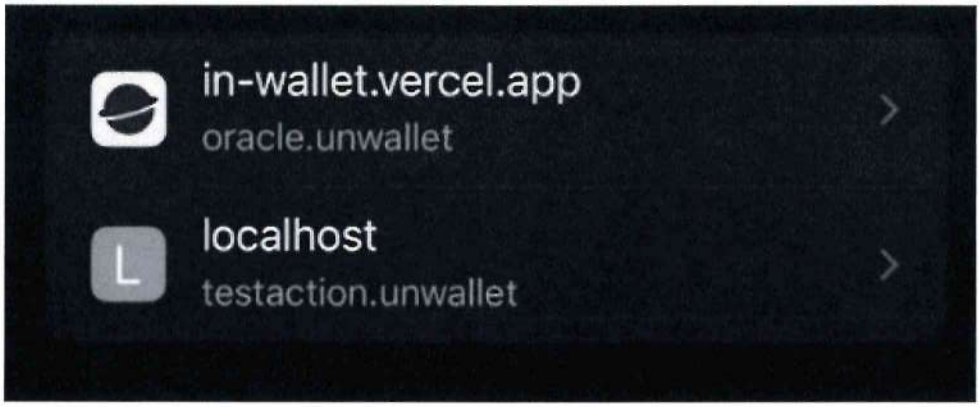

Рис. 3.7. Приклад використання Passkey на iPhone

3.5 Тестування, аудит та деплоймент

Для забезпечення високої якості та безпеки застосунку були проведені всебічні тести та аудити. Тестування включало юніт-тести та інтеграційні тести для перевірки правильності роботи системи. Було використано інструмент Hardhat для автоматизації процесу тестування та виявлення помилок[15].

Тестування та аудит були важливими етапами для забезпечення надійності InWallet. Спочатку проводилися юніт-тести, щоб перевірити окремі функції застосунку. Юніт-тести допомагають виявити помилки на ранніх стадіях розробки та забезпечують правильну роботу кожного компонента системи. Далі проводилися інтеграційні тести, щоб перевірити, як різні компоненти взаємодіють між собою і чи немає конфліктів між ними.

Аудит смарт-контрактів був проведений з використанням інструментів AI для перевірки коду на наявність вразливостей. Аудит включав перевірку коду на наявність відомих вразливостей та оцінку загальної безпеки смарт-контрактів. Результати аудиту показали високу безпеку смарт-контрактів, що дозволило впевнено переходити до етапу деплойменту. Після завершення розробки та тестування проект перейшов до етапу деплойменту. Смарт-контракти були розгорнуті на основних та тестових мережах, а фронтенд був розміщений на Vercel для забезпечення швидкого та надійного доступу до застосунку.

Процес деплойменту включав розробку автоматизованих скриптів для розгортання смарт-контрактів та фронтенду. Це включало налаштування середовища, завантаження контрактів у мережу та налаштування сервера для розміщення фронтенду.

3.6 Складова програми

Переходячи до заключної частини, важливо розглянути ключові складові програми InWallet, що забезпечують її функціональність та зручність у використанні. Основні екрани та інтерфейсні компоненти, розроблені для взаємодії користувача з системою, є важливими елементами, що дозволяють забезпечити високий рівень користувацького досвіду. Нижче наведено основні кроки, які проходить користувач під час роботи з InWallet.

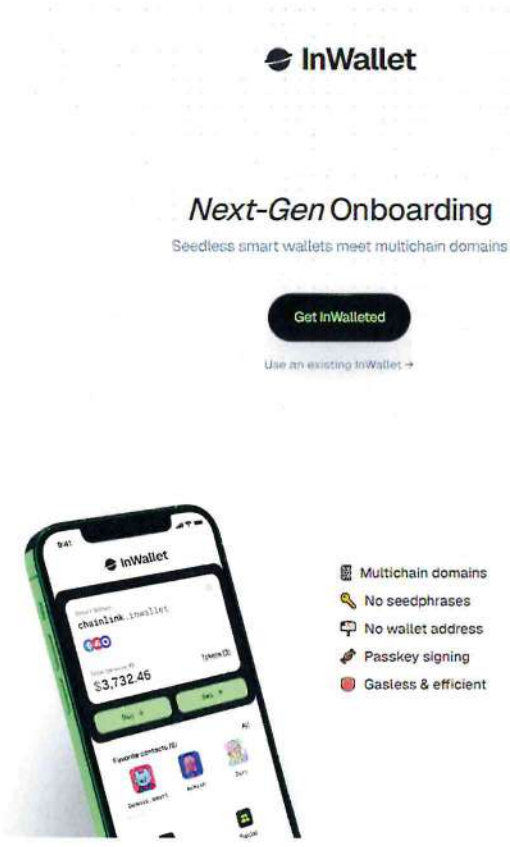

Рис. 3.8. Головне меню екрану

Головне меню екрану є першим екраном, який користувач бачить після запуску InWallet. Воно надає основні опції для початку роботи з застосунком,

такі як створення нового гаманця або використання існуючого. Інтерфейс простий та зрозумілий, що дозволяє новим користувачам швидко зрозуміти, як розпочати роботу.

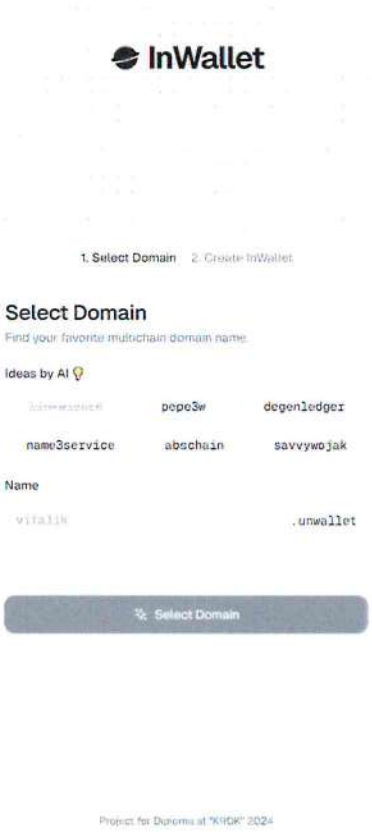

Рис. 3.9. Меню з вибором імені домену

Наступний крок після початкового екрану – це вибір імені домену для нового гаманця. Користувач може обрати ім'я домену з автоматично згенерованих варіантів або ввести власне ім'я. Цей екран також пропонує користувачу підказки на основі AI для полегшення вибору.

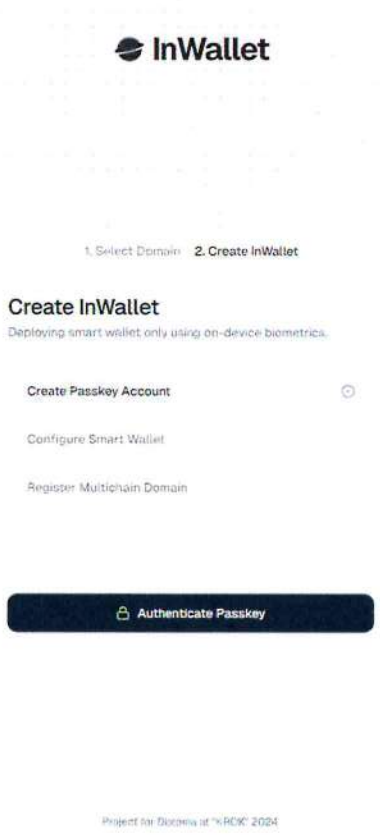

Рис. 3.10. Меню з створенням Passkey, створенням гаманця

Після вибору імені домену користувач переходить до створення гаманця. Цей процес включає створення облікового запису з використанням біометричної автентифікації Passkey, налаштування смарт-гаманця та реєстрацію мульти-чейн домену. Інтерфейс цього меню дозволяє користувачу легко пройти через всі етапи створення гаманця.

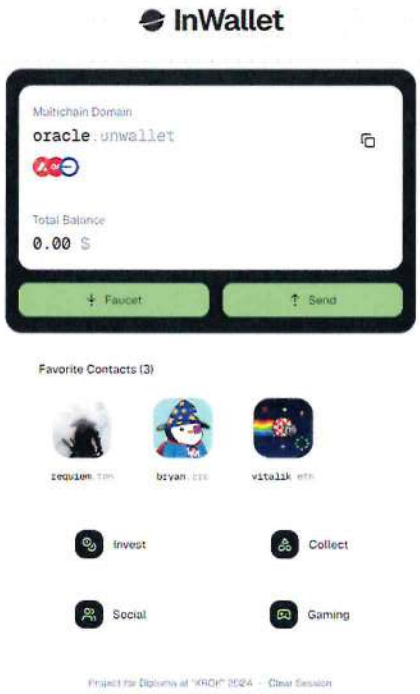

Рис. 3.11. Меню гаманця користувача

Головне меню гаманця користувача відображає основну інформацію про стан рахунку, такі як баланс, улюблені контакти та доступні дії (наприклад, інвестування, збирання, соціальні активності, перейти на coinmarketcap.com). Цей екран забезпечує зручний доступ до всіх функцій гаманця та дозволяє користувачам ефективно керувати своїми криптовалютними активами.

Рис. 3.12. Сторінка 404, якщо сторінку не знайдено

Сторінка 404 відображається, якщо користувач намагається перейти на неіснуючу або недоступну сторінку. Вона повідомляє користувача про помилку та пропонує повернутися на головну сторінку або скористатися іншими функціями застосунку.

Висновок

Розвиток технологій Web3 відкриває нові можливості для створення більш прозорих, безпечних та децентралізованих систем. Використання блокчейну дозволяє створювати додатки, які забезпечують високий рівень контролю над даними і активами користувачів без участі посередників. У цьому контексті криптовалютні гаманці грають ключову роль, оскільки вони надають користувачам можливість безпечно зберігати та керувати своїми цифровими активами.

Основними викликами, з якими стикаються користувачі Web3 гаманців, є складність управління безпекою, необхідність забезпечення інтуїтивно зрозумілого інтерфейсу та інтеграція з різними блокчейн-мережами. Ці проблеми потребують інноваційних підходів для їх вирішення. Розробка зручних та безпечних рішень для управління криптовалютами є важливим напрямком для подальшого розвитку цієї технології.

Впровадження децентралізованих фінансових систем та інших Web3 застосунків дозволяє досягти значних покращень у безпеці, ефективності та прозорості фінансових операцій. Це створює передумови для нових бізнес-моделей і способів взаємодії користувачів із цифровими активами, що раніше були неможливі в традиційних централізованих системах.

Подальший розвиток Web3 також залежить від спрощення користувацького досвіду, що включає створення інтуїтивно зрозумілих інтерфейсів та забезпечення зручності використання децентралізованих застосунків. Це дозволить залучити ширшу аудиторію користувачів, які зможуть легко адаптуватися до нових технологій.

У майбутньому можна очікувати подальшого розвитку та вдосконалення Web3 гаманців, що сприятиме їх масовому впровадженню. Це відкриває нові можливості для створення децентралізованих фінансових систем, підвищуючи

ефективність та прозорість фінансових операцій, і забезпечуючи новий рівень взаємодії у цифровому світі.

Список використаних джерел

1. Що таке гаманці Web3? [Електронний ресурс] – Режим доступу до ресурсу: https://academy.binance.com/en/articles/what-are-web3-wallets
2. Що таке Web3 застосунки? DApp - децентралізована програма [Електронний ресурс] – Режим доступу до ресурсу: https://www.exodus.com/support/en/articles/8598734-what-are-dapps-and-web3-apps
3. Що таке блокчейн? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/blockchain
4. Централізація проти децентралізації. Прийняття ефективних Орг. рішень [Електронний ресурс] – Режим доступу до ресурсу: https://corporatefinanceinstitute.com/resources/management/centralization/
5. Документація Solidity [Електронний ресурс] – Режим доступу до ресурсу: https://docs.soliditylang.org/en/v0.8.25/
6. Документація Hardhat [Електронний ресурс] – Режим доступу до ресурсу: https://hardhat.org/docs
7. Документація React [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/docs/getting-started.html
8. Архітектура Web3 застосунків. Глибокий аналіз [Електронний ресурс] – Режим доступу до ресурсу: https://www.preethikasireddy.com/post/the-architecture-of-a-web-3-0-application
9. Смарт-контракт [Електронний ресурс] – Режим доступу до ресурсу: https://www.coinbase.com/ru/learn/crypto-basics/what-is-a-smart-contract
10. Документація Viem [Електронний ресурс] – Режим доступу до ресурсу: https://viem.sh/docs/getting-started
11. Документація Next.js [Електронний ресурс] – Режим доступу до ресурсу: https://nextjs.org/docs
12. Документація Node.js [Електронний ресурс] – Режим доступу до

ресурсу: https://nodejs.org/docs/latest/api/

13. База даних KV від Vercel [Електронний ресурс] – Режим доступу до ресурсу: https://vercel.com/docs/storage/vercel-kv
14. Документація Foundry [Електронний ресурс] – Режим доступу до ресурсу: https://book.getfoundry.sh/
15. Документація OpenAI [Електронний ресурс] – Режим доступу до ресурсу: https://platform.openai.com/docs/api-reference
16. Документація Passkey [Електронний ресурс] – Режим доступу до ресурсу: https://developer.chrome.com/docs/identity/passkeys?hl=en
17. Документація Turnkey [Електронний ресурс] – Режим доступу до ресурсу: https://docs.turnkey.com/

Додатки

Додаток А. Код з деталями мережі користувача

```
import { FC } from 'react'
import { Link } from 'lucide-react';

import { publicResolverCcipAbi, publicResolverCcipAddress } from '@/wagmi.generated'
import { BookUser, Coins } from 'lucide-react'
import { Chain, namehash, zeroAddress } from 'viem'
import { avalancheFuji } from 'viem/chains'
import { useReadContract } from 'wagmi'

import { DomainContext } from '@/app/atoms'
import ChainIcon from '@/components/chain-icon'
import { Button } from '@/components/ui/button'
import { HoverCard, HoverCardContent, HoverCardTrigger } from '@/components/ui/hover-card'
import { Separator } from '@/components/ui/separator'
import { useChainlinkPriceFeeds } from '@/hooks/use-chainlink-price-feeds'
import { convertEVMChainIdToCoinType } from '@/utils/coin-type'
import { copyToClipboard } from '@/utils/copy-to-clipboard'

import { useDomainMultichainBalances } from '../_hooks/use-domain-multichain-balances'

interface ChainDetailsHoverCardProps {
 chain: Chain
 domainContext: DomainContext
}
export const ChainDetailsHoverCard: FC<ChainDetailsHoverCardProps> = (props) => {
 return (
  <HoverCard openDelay={0} closeDelay={50}>
   <HoverCardTrigger>
    <button
```

```
        type="button"
        className="flex rounded-full outline-none ring-offset-background transition-transform
hover:scale-110 focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2"
      >
        <ChainIcon chain={props.chain} />
      </button>
    </HoverCardTrigger>
    <HoverCardContent className="w-[22.2rem] max-w-full overflow-hidden" sideOffset={12}>
      <div className="flex flex-col gap-4">
        <div className="flex items-center justify-center gap-2">
          <ChainIcon size={32} chain={props.chain} />
          <h4 className="font-semibold leading-none">{props.chain.name}</h4>
        </div>
        <div className="flex flex-col gap-3 rounded-sm border bg-muted p-2 pb-3">
          <DomainChainResolvedAddress {...props} />
          <Separator className="-mx-2 w-auto" />
          <DomainChainFetchedBalance {...props} />
        </div>
        <div className="flex flex-col gap-1">
          <div className="text-center text-xs text-muted-foreground">
            Resolved smart wallet address via ENSIP-11 ⚡
          </div>
          <div className="text-center text-xs text-muted-foreground">
            Converted to USD using Chainlink Data Feeds 🤖
          </div>
        </div>
      </div>
    </HoverCardContent>
  </HoverCard>
 )
}
```

```
let address: string | undefined = undefined;


const DomainChainResolvedAddress: FC<ChainDetailsHoverCardProps> = ({ chain,
domainContext }) => {
 // Resolve address via ENSIP-11
 const { domain } = domainContext
 const query = useReadContract({
  chainId: avalancheFuji.id,
  address: publicResolverCcipAddress[avalancheFuji.id],
  abi: publicResolverCcipAbi,
  functionName: 'addr',
  args: [namehash(domain), BigInt(convertEVMChainIdToCoinType(chain.id))],
 })

 address = query.data;


 return (
  <div className="flex flex-col gap-1.5">
   <div className="flex items-center gap-1.5">
    <BookUser size={14} />
    <h5 className="text-sm font-medium">Resolved Address</h5>
    <Button size="xs" className="ml-auto" onClick={() => copyToClipboard(query.data)}>
     Copy
    </Button>
   </div>
   <div className="h-[13.5px] font-mono text-xs text-muted-foreground">
    {!query.isLoading && (
     <span className="animate-in fade-in-0">{query.data || zeroAddress}</span>
    )}
   </div>
  </div>
 )
```

```
}

const DomainChainFetchedBalance: FC<ChainDetailsHoverCardProps> = ({ chain, domainContext
}) => {
 const balances = useDomainMultichainBalances(domainContext, [chain], false);
 const balancesWithPrices = useChainlinkPriceFeeds(balances || []);

 const getExplorerUrl = (chain: number, address: string | undefined) => {
  switch (chain) {
   case 43113:
    return `https://testnet.snowtrace.io/address/${address}`;
   case 84532:
    return `https://base-sepolia.blockscout.com/address/${address}`;
   case 11155420:
    return `https://optimism-sepolia.blockscout.com/address/${address}`;
  }
 };

 const explorerUrl = getExplorerUrl(chain.id, address);

 return (
  <div className="flex flex-col gap-1.5">
   <div className="flex items-center justify-between">
    <div className="flex items-center gap-1.5">
     <Coins size={14} />
     <h5 className="text-sm font-medium">Fetched Balance</h5>
    </div>
    <button
     onClick={() => window.open(explorerUrl, '_blank')}
     className="flex items-center justify-center p-2 rounded hover:bg-gray-200"
     aria-label="Open in Explorer"
    >
```

```
          <Link size={14} />
        </button>
      </div>
      <div className="h-[13.5px] font-mono text-xs text-muted-foreground">
        {!!balances?.length && (
          <span className="animate-in fade-in-0">
            {balances[0].formatted || '0'} {balances[0].symbol}
          </span>
        )}
        {!!balances?.length &&
          !balancesWithPrices.isLoading &&
          !!balancesWithPrices.data?.totalFormattedInUSD && (
            <span className="animate-in fade-in-0">
              {' '}
              ({balancesWithPrices.data.totalFormattedInUSD} $)
            </span>
          )}
      </div>
    </div>
  );
};
```

Додаток В. Код головної сторінки InWallet

```
import Image from 'next/image'

import ShimmerButton from '@/components/magicui/shimmer-button'

import { DashboardButton } from './_components/dashboard-button'
import { SigninButton } from './_components/signin-button'
import phoneImg from '/public/phone.png'

export default function HomePage() {
 return (
  <>
   <main className="mt-10 flex grow flex-col items-center justify-center">
    <div className="flex grow flex-col items-center justify-center gap-10 hxl:gap-14">

     {/* Title & Description */}
     <div className="flex max-w-prose flex-col gap-2 text-center">
      <h2 className="text-4xl font-medium tracking-tight">
       <em>Next-Gen</em> Onboarding
      </h2>
      <p className="text-lg tracking-tight text-muted-foreground">
       Seedless smart wallets meet multichain domains
      </p>
     </div>

     <div className="flex flex-col items-center justify-center gap-4 text-center">
      <div className="flex justify-center gap-3">
       {/* Start Setup */}
       <ShimmerButton
        href="/setup"
        className="h-14 shadow-2xl"
        shimmerColor="hsl(var(--brand))"
```

```
            shimmerSize="0.1em"
          >
            <span className="whitespace-pre-wrap px-1 text-center text-base font-medium
leading-none tracking-tight text-brand dark:from-white dark:to-slate-900/10">
              Get InWalleted
            </span>
          </ShimmerButton>

          {/* Dashboar Button */}
          <DashboardButton />
        </div>

        {/* Sign-in */}
        <SigninButton className="text-sm font-medium text-muted-foreground
hover:text-foreground" />
      </div>
    </div>

    {/* Phone Image with Features */}
    <div className="mt-10 grid grid-cols-2 items-center justify-center">
      <div className="-ml-12 h-[300px] overflow-hidden sm:h-[350px] md:h-[400px] hlg:!h-auto
hlg:!overflow-auto">
        <Image
          src={phoneImg}
          alt="InWallet App Screenshot"
          width={450}
          className="select-none"
          priority
        />
      </div>
      <ul className="mb-[20%] flex flex-col gap-2 whitespace-nowrap font-medium
hlg:!mb-[60%]">
        <li>⛓  Multichain domains</li>
```

```
            <li>🔑&amp;nbsp;&amp;nbsp;No seedphrases</li>
            <li>📪&amp;nbsp;&amp;nbsp;No wallet address</li>
            <li>🧬&amp;nbsp;&amp;nbsp;Passkey signing</li>
            <li>👛&amp;nbsp;&amp;nbsp;Gasless &amp; efficient</li>
          </ul>
        </div>
      </main>

      {/* Marquee with latest domains */}
      {/* <LatestDomainsMarquee /> */}
    </>
  )
}
```

Додаток С. Код з промптом для генерації домену

```
import { NextRequest, NextResponse } from 'next/server'

import { kv } from '@vercel/kv'
import { OpenAIStream, StreamingTextResponse } from 'ai'
import { z } from 'zod'

import { openai } from '@/config/openai'
import { getAndValidateRequestData } from '@/utils/get-and-validate-request-data'

export const runtime = 'edge'

const getUserPrompt = (
  amount: number,
) => `Custom Instruction: Craft Web3 usernames that can be loosely related to one of the following projects:
- Chainlink (the decentralized Web3 oracle network),
- ENS (the Ethereum Name Service),
- Account Abstraction (the new wallet standard on EVM networks),
- Crypto Memes (e.g. pepe, degen, wojak, etc.).

Disallowed Results: linkoracle,enshandle,chainfeed,ethnamer,ensvision,linknode,oraclechain

Amount: ${amount}`

const SYSTEM_PROMPT = `# Instruction

You are a helpful assistant that replies with a list of creative social handles. You are always strictly following the requirements and defined output format below.

# Requirements
```

Usernames

- Usernames should be between 4 and 12 characters long
- Usernames should only contain alphanumerical characters (lowercased) and optionally hyphens in between
- Usernames should adhere to the "Custom Instruction" provided
- Usernames should never contain the word "username" (this is just the example format)

List

- The length of the list should be exactly equal to the "Amount" provided
- Comma-seperated list of usernames
- DO NOT include spaces after commas
- DO NOT use line breaks
- DO NOT enumerate the list

Format

username1,username2,username3,username4,username5,...

Examples

Example 1

User

Custom Instruction: Craft usernames that resonate with trending topics in blockchain, web3, and the cultural nuances of the crypto space, ensuring a modern and insightful connection to the digital frontier.

Input: web3

Quantity: 5

Assistant

web3-degen,w3bthree,wizard3,hodl3r,web69

Example 2

User

Custom Instruction: Craft usernames that resonate with trending topics in blockchain, web3, and the cultural nuances of the crypto space, ensuring a modern and insightful connection to the digital frontier.

Quantity: 9

Assistant

min3r,chainrider,cryptoninja,ethlord,digitrailblazer,tokenwhisperer,degen3,web3guru,decentralizoor

```
const schema = z.object({
 amount: z.number().int().min(1).max(10),
})
export async function POST(req: NextRequest) {
 try {
  // Get and validate the request data.
  const { data, error } = await getAndValidateRequestData(req, schema)
  if (error) {
   console.error(error)
   return NextResponse.json({ error: 'Bad Request' }, { status: 400 })
  }
  const { amount } = data
```

```
// Check if ideas are cached
const cacheKey = `name-ideas-${amount}`
const cached = await kv.get(cacheKey)
if (cached) {
 const chunks = (cached as string).split('')
 const stream = new ReadableStream({
  async start(controller) {
   for (const chunk of chunks) {
    const bytes = new TextEncoder().encode(chunk)
    controller.enqueue(bytes)
   }
   controller.close()
  },
 })
 return new StreamingTextResponse(stream)
}

// Stream the ideas via OpenAI
const USER_PROMPT = getUserPrompt(amount)

const response = await openai.createChatCompletion({
 model: 'gpt-4-1106-preview',
 stream: true,
 temperature: 0.75,
 max_tokens: 250,
 frequency_penalty: 0.25,
 top_p: 1,
 presence_penalty: 0,
 n: 1,
 messages: [
  {
```

```
        role: 'system',
        content: SYSTEM_PROMPT,
      },
      {
        role: 'user',
        content: USER_PROMPT,
      },
    ],
  })

  // Convert the response to a text stream
  const stream = OpenAIStream(response, {
    onCompletion: async (completion: string) => {
      // Cache the ideas
      await kv.set(cacheKey, completion)
      await kv.expire(cacheKey, 10) // Expire in 10 seconds
    },
  })

  return new StreamingTextResponse(stream)
 } catch (error) {
  console.error(error)
  return NextResponse.json({ error: 'Internal Server Error' }, { status: 400 })
 }
}
```