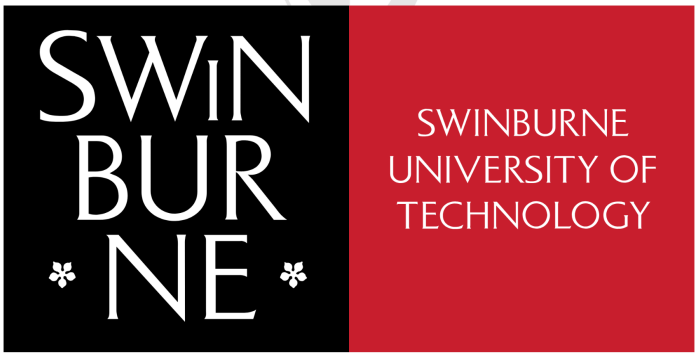
COS30049 - Computing Technology Innovation Project

PROJECT DESIGN REPORT - ASSIGNMENT 02

GROUP SET 3 - CryptoPath



Group Member:

1. Nguyen Minh Duy 104974743
2. Phan Cong Hung: 104995595
3. Le Nguyen Dang Duy: 105028557

Lecturer: Mr. Tristan Nguyen

Campus: Ho Chi Minh City

CryptoPath Project Report..... 2
1. Introduction..... 2
2. Working Phase..... 3
3. Project Overview..... 5
4. Team and Project Structure..... 6
4.1 Team Members..... 6
4.2 Diagrams..... 7
User Stories from Assignment 1..... 7
Homepage & Navigation..... 7
Search & Wallet Information..... 8
Transaction History & Details..... 8
User Authentication (Login & Signup)..... 8
Security & Data Handling..... 8
4.3 Technical Architecture..... 9
Frontend..... 9
Graph Database..... 10
API Integration and Real-Time Data..... 10
5. Website Details..... 11
Landing Page:..... 11
Market Overview Page:..... 18
PriceTable Page:..... 20
Transactions Page:..... 21
Faucet Page:..... 23
NFT Page:..... 24
PATH Marketplace:..... 24
Collection Explorer:..... 26
Search Page:..... 27
Search Result:..... 28
Login Page:..... 31
6. Core Functionalities..... 31
6.1 User Authentication and Session Management..... 31
6.2 Wallet and Portfolio Management..... 31
6.3 Blockchain Data Visualization..... 31
6.4 Market Insights and Analytics..... 32
6.5 NFT Marketplace..... 32
6.6 Faucet Integration..... 32
6.7 Search and Explore..... 32
6.8 Support and Accessibility..... 32
7. Backend Overview..... 32
7.1 RESTful API for Blockchain Data Retrieval..... 32

7.2 Data Transformation & Caching..... 32
7.3 Authentication & Rate Limiting..... 33
7.4 Graph Database Integration..... 33
7.5 API Integration..... 33
7.6 Real-Time Notifications..... 33
7.7 NFT Marketplace Integration..... 34
7.8 User Personalization..... 34
7.9 Supabase Details..... 34
7.10 Data Storage..... 35
7.11 Smart Contract..... 36
8. Backend Code..... 37
Setup Instructions..... 45
API Design..... 46
1. Wallet API..... 46
Query Parameters:..... 46
Response:..... 46
2. NFT Stats API..... 46
Response:..... 47
3. DeFi TVL API..... 47
Query Parameters:..... 47
Response:..... 47
4. CoinMarketCap Listings API..... 47
Query Parameters:..... 48
Response:..... 48
5. Whale Alerts API..... 48
Query Parameters:..... 48
Response:..... 48
6. Gas Prices API..... 49
Response:..... 49
7. Portfolio API..... 49
Query Parameters:..... 49
Response:..... 49
8. Trending Coins API..... 50
Response:..... 50
9. Fear & Greed Index API..... 50
Response:..... 50
References..... 51

CryptoPath Project Report

1. Introduction

Blockchain technology has transformed data management by providing decentralization, transparency, and increased security; yet, these same characteristics also present issues. While decentralization and anonymity promote privacy and creativity, they can also be used by criminals for money laundering, frauds, and other cybercrimes. Furthermore, the massive volume of blockchain transactions, combined with their intricate flow, makes it difficult to trace and detect suspicious activity.

CryptoPath was designed to meet these challenges front on. Our revolutionary transaction tracing engine converts complex blockchain data into an interactive and user-friendly visualization tool. CryptoPath simplifies the interpretation, navigation, and analysis of blockchain transactions by allowing users to query Ethereum wallet addresses and view comprehensive transaction histories via a dynamic graphical interface. Investigators may use real-time data analytics, detailed reports, and intuitive graphs to search across different blockchains and swiftly uncover potential dangers, while also benefiting from blockchain's transparency and better security.

Website link: https://cryptopathabcd.vercel.app

Example Wallet for testing:

NFT holder: 0x2F1D9c7ba44F5B72f0AD1bad2af5F9dF7EE7d508

Receiving: 0x0000000000000068f116a894984e2db1123eb395View

Neo4j offchain: 0xb0606f433496bf66338b8ad6b6d51fc4d84a44cd
0x3089df0e2349faea1c8ec4a08593c137da10fe2d

Dev Wallet:

viable congress target wheel index bicycle fury drop ranch butter paper cheese
3ffc4efc5badcdb8637b8ce7c70b4d34e8cf0ce0c3240c11249630cb5ff23224

2. Working Phase

Phase 1: Foundation & Architecture

- Core Infrastructure Setup
  - Project initialization with Next.js framework
  - Tailwind CSS styling system integration
  - Component library development (UI/UX foundations)
  - Basic state management implementation
- Authentication System
  - Web3 wallet connectivity with multiple wallet options
  - Supabase authentication integration

    - User profile management foundations
- Development Environment
    - API endpoints configuration
    - Environment variable management
    - Development toolchain setup

Phase 2: Core Functionality Development

- Blockchain Data Integration
    - Integration with Etherscan API
    - On-chain transaction tracking capabilities
    - Basic block explorer functionality
    - Gas price tracking and optimization
- Search & Discovery
    - Address/transaction search implementation
    - Token tracking system
    - Basic analytics dashboard
- User Experience
    - Responsive UI implementation
    - Loading states and error handling
    - Initial dark theme implementation

Phase 3: Feature Completion & Enhancement

- Advanced On-Chain Analytics
    - Real-time price data integration (ETH/BNB)
    - Gas fee prediction and estimation
    - Transaction history visualization
    - Portfolio tracking capabilities
- NFT Integration
    - Collection browsing system
    - Collection social features (comments, watchlisting)
    - Creator profiles and verification
    - NFT metadata display and analysis
- UX Refinement
    - Advanced search with type filtering
    - Watchlist functionality
    - Off-chain data integration
    - Interactive data visualizations

Phase 4: Platform Optimization & Security

- Performance Optimization
    - Component code splitting
    - Server-side rendering optimization
    - API response caching
    - Database query optimization
- Security Enhancements
    - Wallet connection security auditing
    - Data encryption implementation (using bcryptjs)

    - API rate limiting
    - Cross-site request forgery protection
- Cross-Chain Expansion
    - Support for additional EVM networks
    - Cross-chain transaction tracking
    - Multi-chain portfolio aggregation

Phase 5: Advanced Features & Ecosystem Growth

- DeFi Analytics Integration
    - Yield farming analytics
    - Liquidity pool tracking
    - DeFi protocol integration
    - Risk assessment tools
- Enterprise Features
    - API key generation for developers
    - Customizable dashboards
    - Batch transaction analysis
    - Export functionality for reports
- Community & Growth
    - User profiles and social features
    - Developer documentation
    - Analytics sharing capabilities
    - Mobile optimization

Each phase builds upon the previous one, starting with the core infrastructure and progressively integrating more advanced features while continuously enhancing performance and security. Our approach follows industry best practices for Web3 application development, ensuring a structured and efficient progression from initial development to a fully-featured blockchain explorer platform.

3. Project Overview

CryptoPath is a blockchain transaction visualization system that emphasizes clarity and ease of use. The platform focuses on several core functionalities:

Title	Type	Description
Wallet search bar	Front-end	A search bar that allows users to enter a cryptocurrency wallet address to obtain information about it. After pressing Enter, the website takes you to the proper wallet details page.
Wallet overview	Front-end	A part that displays basic information about the searched wallet, such as its balance, number of transactions, total amount transferred and received, and last active time.

Interactive transaction graph	Front-end	A directed graph of transactions involving the searched wallet. Each node represents a wallet, whereas each edge represents a transaction. Clicking on a node in the graph takes the user to the wallet information page.
Transaction history table	Front-end	A table containing detailed information on transactions involving the searched wallet, including the transaction ID, from, to, amount, and date.
Responsive design	Front-end	Ensure that the page elements are appropriately shown across multiple devices (mobile, tablet, and desktop).
Graph database	back-end	Create a graph database to hold all transaction data supplied in the assignment.
Server-side processing	back-end	Implement the backend and required APIs for the platform's functioning (communicate with the database to retrieve and process blockchain data).
User-friendly error handling	Front-end	Ensure that any errors are handled appropriately and that the user receives nice error messages explaining what went wrong (and what they may do, if relevant).
Efficient Data Storage	back-end	In addition to the dataset supplied in the assignment, the website obtains transaction data directly from the blockchain, providing users with accurate and up-to-date information.
Real-Time Updates	back-end	Integration with blockchain APIs and WebSocket technology guarantees that the platform's transaction information is up to date.
Detailed data analytics & visualization	back-end	Multiple charts are provided (not only the wallet overview, transaction graph, and table) to offer users with a comprehensive picture and deep insights into the searched wallet.
Direct blockchain data fetching	back-end	A dropdown menu allows users to pick which they wish to search on.

These functionalities align with the project’s primary objective—to demystify blockchain data and present it in an accessible, intuitive format.

4. Team and Project Structure

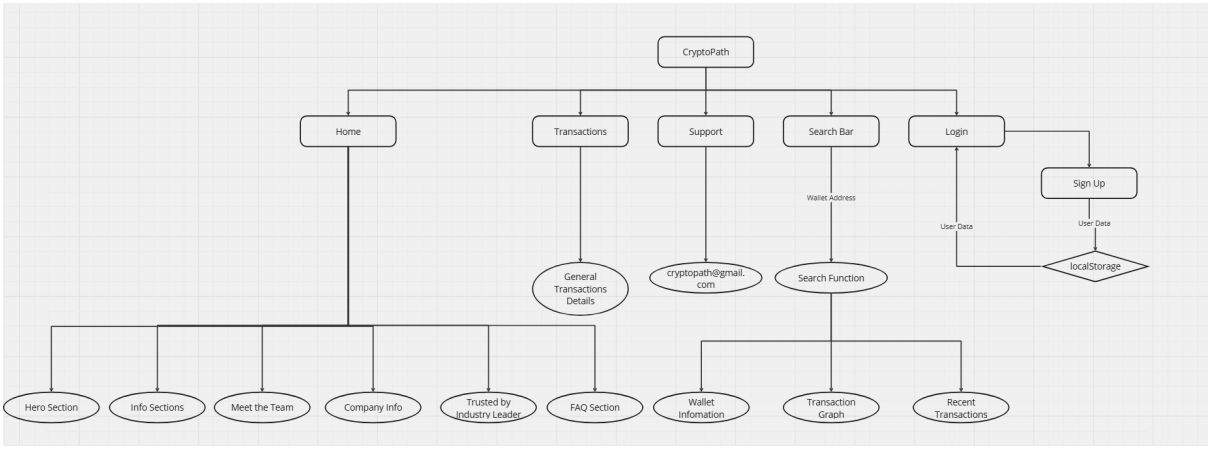
4.1 Team Members

CryptoPath is a collaborative effort led by a our team of developers:

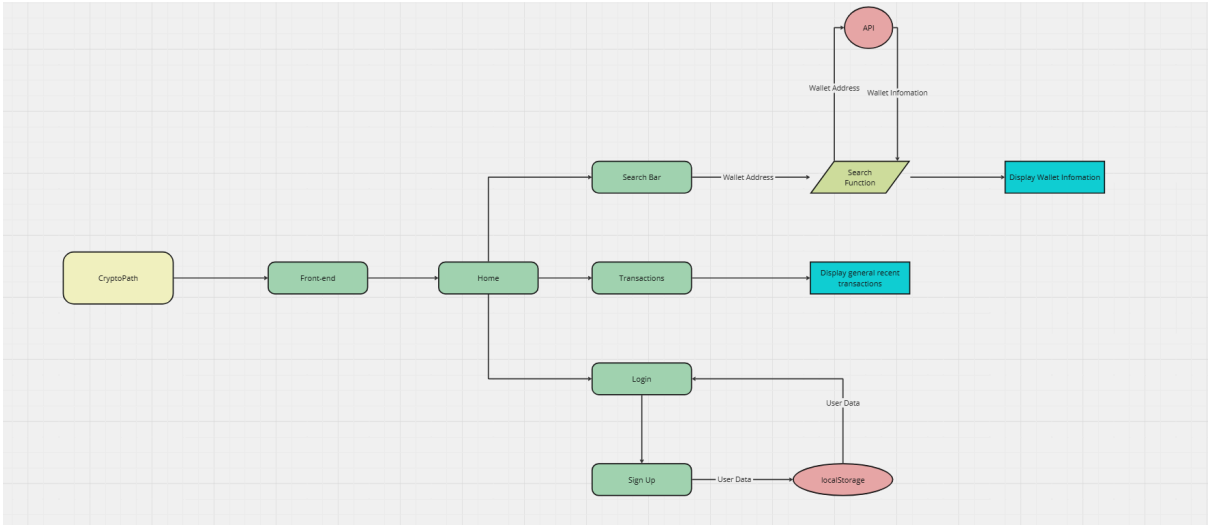
- **Le Nguyen Dang Duy (105028557):** Frontend Lead and Graph Visualization Specialist
- **Phan Cong Hung (104995595):** Backend & Frontend Developer / API Integration
- **Nguyen Minh Duy (104974743):** Team Leader / Full-Stack Developer / Product Architect

4.2 Diagrams

Architecture Diagram from Assignment 1



Workflow Diagram from Assignment 1



User Stories from Assignment 1

Homepage & Navigation

- **As a user,** I want to access the homepage so that I can explore the key features of the platform and navigate easily to different sections.

Search & Wallet Information

- **As a blockchain user,** I want to search for a wallet address using the search bar so that I can retrieve detailed information about its transaction history and interactions.
- **As an analyst,** I want the search function to connect with an API so that I can get real-time wallet details and transaction data.
- **As a user,** I want to see wallet details displayed in a structured format so that I can better understand the financial activities associated with a given address.

Transaction History & Details

- **As a blockchain enthusiast,** I want to browse general transaction history so that I can monitor network activity and identify trends.
- **As a user,** I want transaction data to be displayed clearly, including recent transactions, so that I can analyze the flow of assets.

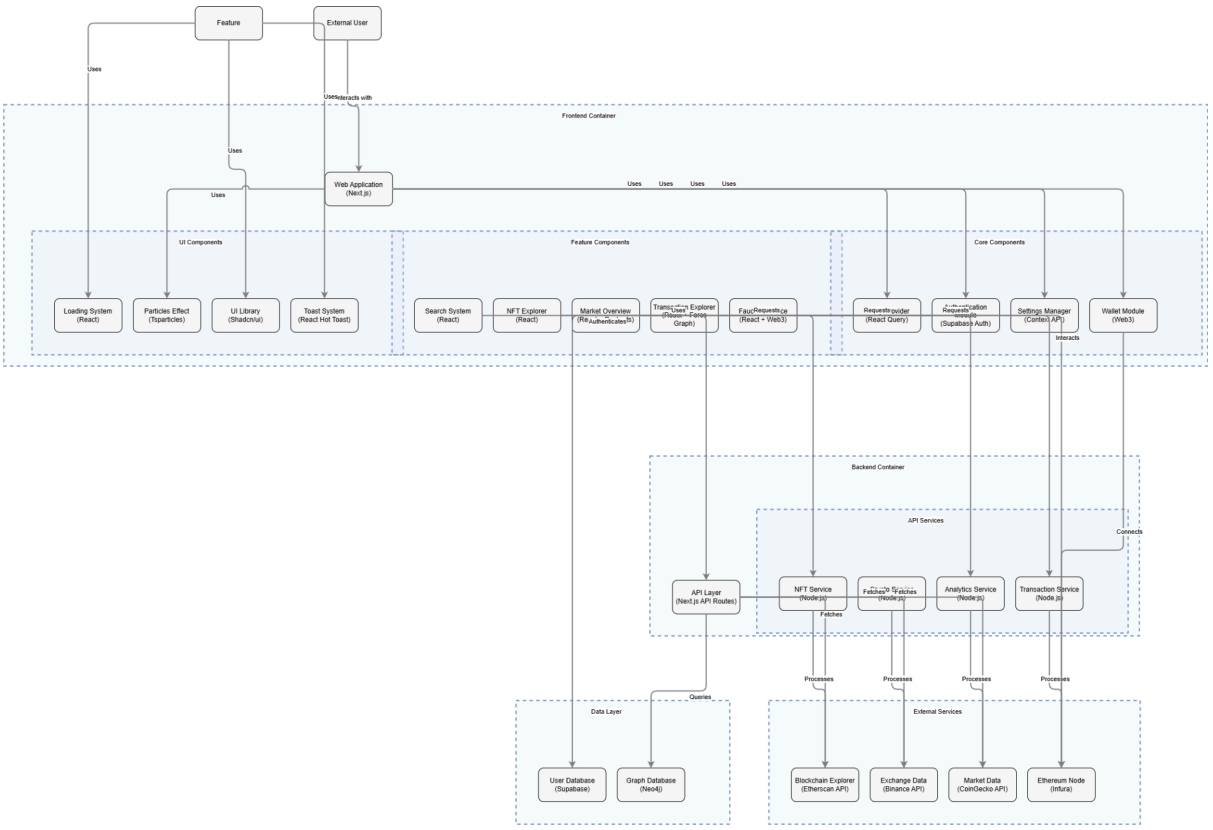
User Authentication (Login & Signup)

- **As a new user,** I want to sign up on the platform so that I can create an account and store my preferences.
- **As a registered user,** I want to log in securely so that I can access my saved data and interact with the platform seamlessly.
- **As a returning user,** I want my login credentials and user preferences to be stored in localStorage so that I don’t have to log in every time I visit the platform.

Security & Data Handling

- **As a user,** I want my account details to be stored securely in localStorage so that I can retain my preferences without compromising security.
- **As a system administrator,** I want authentication processes to be secure so that user data remains protected.

Architecture Diagram from Assignment 2



4.3 Technical Architecture

Frontend

- **Framework:** Developed using Next.js 14 and TypeScript, ensuring robust and scalable code.
- **Styling:** Tailwind CSS and shadcn/ui components provide a modern, responsive design with both light and dark mode support.
- **Visualization:** The use of React-based libraries ensures that the interactive transaction graph is smooth and dynamic.**Backend**

Main design choice	Reason
Dark theme	A black backdrop offers a sleek, sophisticated, and high-tech look while also highlighting crucial items on the page.
Orange accent color	This colour is frequently associated with money and evokes a digital, matrix-like aura, redolent of coding. It is also lively, sustaining an effervescent motif throughout the page without being overly flamboyant.

Tailwind CSS	This CSS framework facilitates rapid development, particularly in teams. Its utility classes can be directly embedded into HTML without the need to create distinct CSS files or determine appropriate class names, which also mitigates the risk of conflicting class names among developers. Finally, the integration of numerous contemporary UI libraries is simplified by the use of Tailwind. (No date, Tailwind CSS)
React Icons	This library offers a comprehensive collection of icons that can be effortlessly imported and utilized as React components. (React Icons, no date)
Next.js	This framework is constructed on the foundation of React, and it is pre-integrated with TypeScript and Tailwind CSS. It offers practical features such as built-in optimizations and file-based routing. Additionally, it provides a plethora of tools for a full-stack web application, including server-side rendering and server actions. (No date, Next.js)

- **API Development:** A RESTful API serves as the backbone for retrieving and processing blockchain data. This API handles data transformation, caching, error handling, and authentication.
- **Data Integration:** The backend interacts with blockchain sources (e.g., Etherscan API) to fetch wallet details and transaction histories, ensuring data accuracy and timeliness.

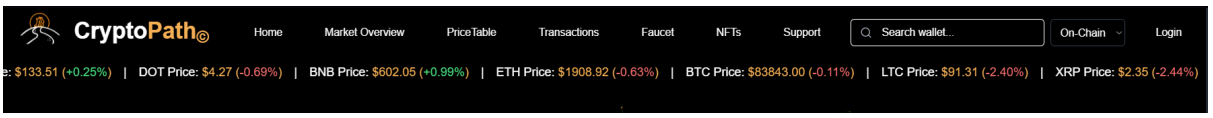
Graph Database

- **Neo4j Integration:** To efficiently map transaction relationships, CryptoPath uses Neo4j as its primary database. The graph schema is designed for optimal query performance, facilitating quick path discovery and data visualization.

API Integration and Real-Time Data

- **Blockchain API:** Integration with Ethereum blockchain APIs (such as Etherscan and Infura) ensures that wallet information and transaction data are retrieved accurately.
- **Real-Time Updates:** The use of WebSocket technology allows for live updates, keeping the data current and providing users with a dynamic viewing experience.

5. Website Details



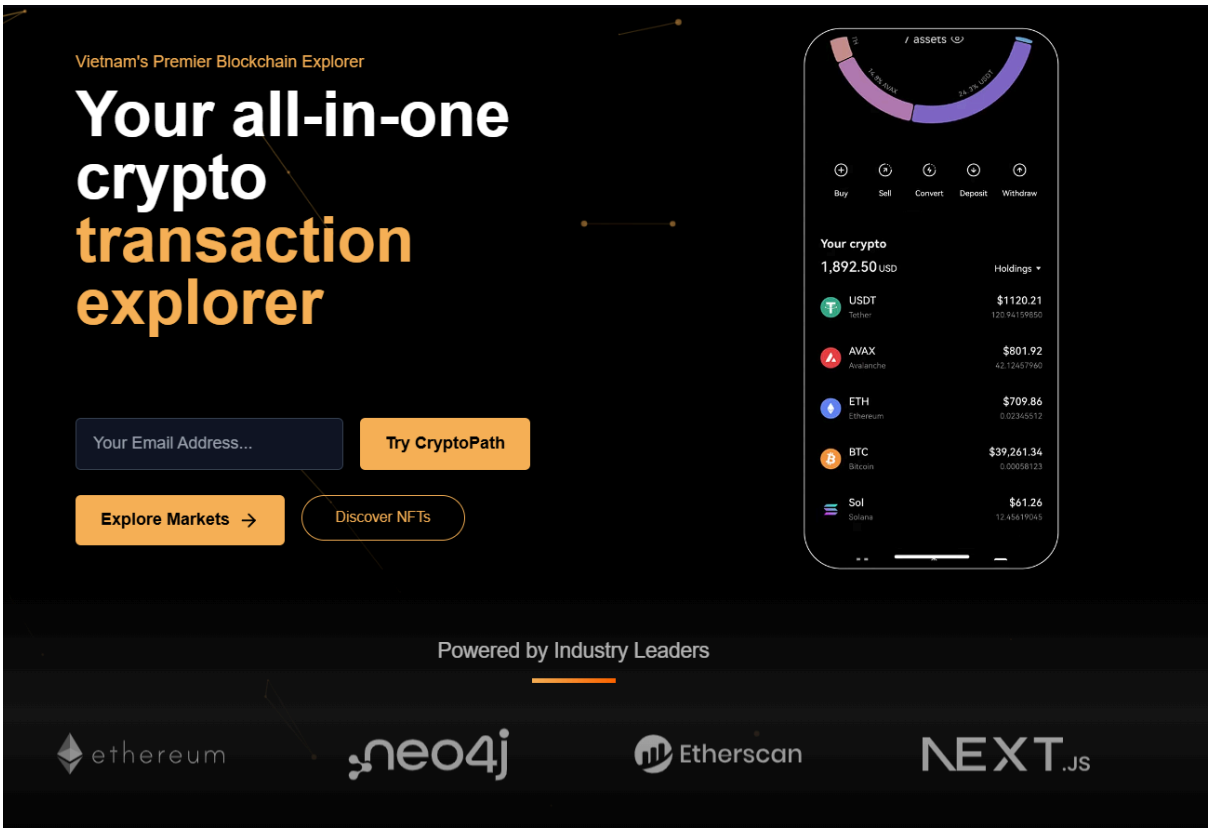
The header section of this website highlights the brand “CryptoPath” alongside a clear navigation bar featuring options such as Home, Market Overview, Price Rate, Transactions, Feeds, NFTs, and Support. This layout instantly communicates the breadth of the platform, emphasizing its focus on providing a comprehensive crypto experience. The dark background, accented by subtle design elements, conveys a sophisticated, modern aesthetic, while consistent fonts, colors, and icons create a cohesive visual identity.

Positioned just below the navigation, a row of real-time cryptocurrency prices serves as a central feature. Displaying well-known assets like BTC, DOT, BNB, ETH, LTC, BCH, and XRP, along with their current values and percentage changes, this element provides immediate market insights. By placing these metrics front and center, CryptoPath encourages users to engage with the data and explore the platform’s functionalities, underscoring its commitment to delivering a seamless, user-friendly experience.

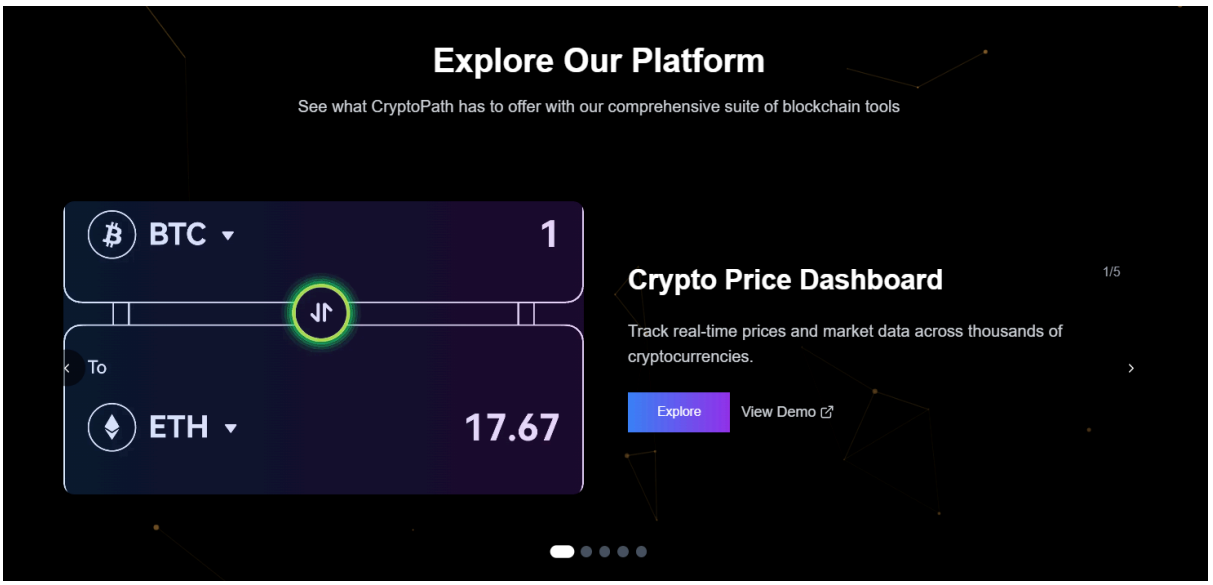
Landing Page:

The explorer section of this website introduces "CryptoPath Explorer," providing a streamlined search bar where users can look up blockchain data by address, transaction hash, block, or token. This feature enhances accessibility, allowing users to navigate the blockchain efficiently. The dark-themed design maintains a professional and futuristic aesthetic, reinforcing the platform’s credibility.

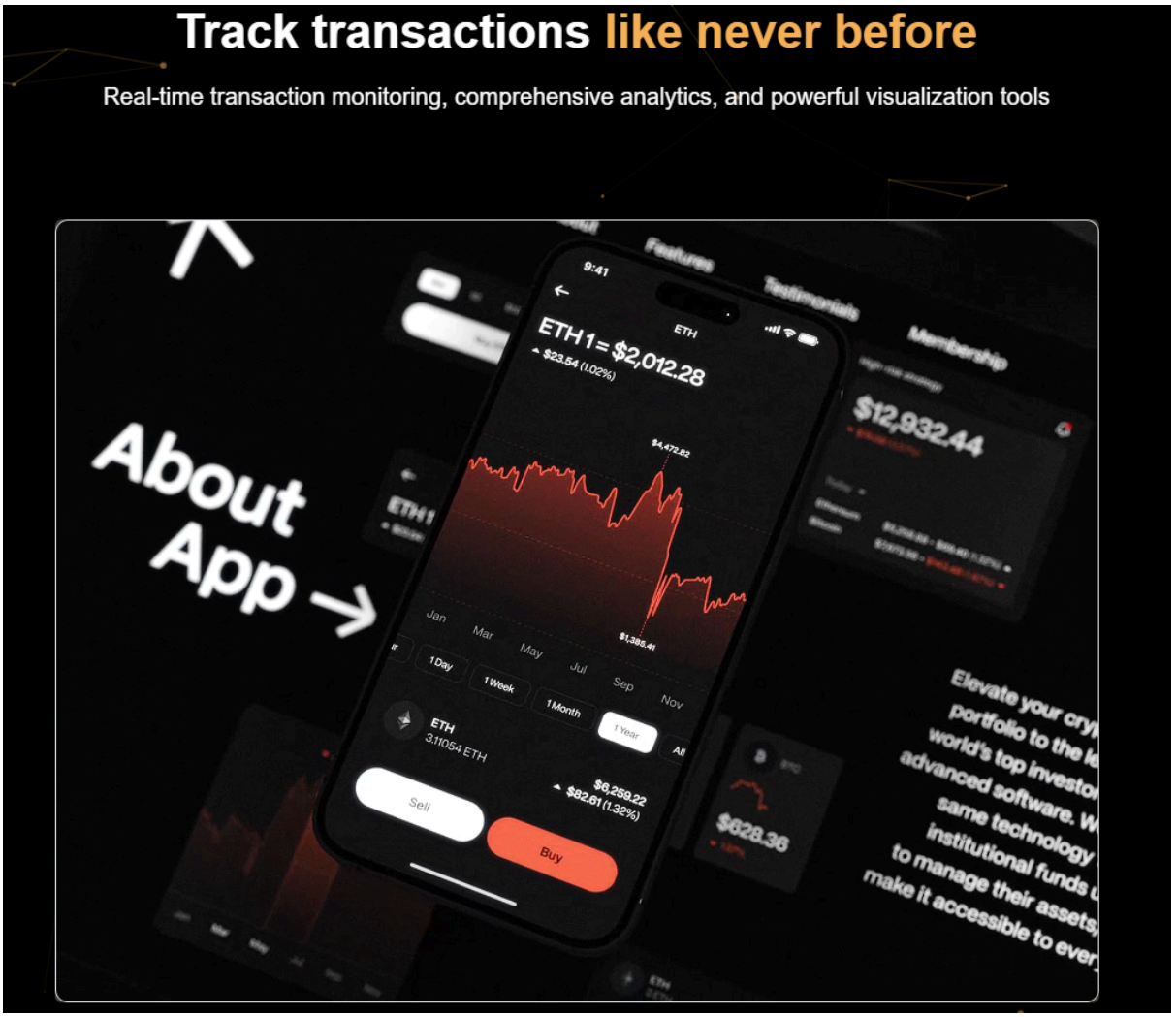
Below the search bar, key blockchain metrics are displayed in an organized format. Ethereum (ETH) and Optimism (OP) prices are shown with their respective percentage changes, offering real-time market insights. Additionally, blockchain activity indicators, such as the latest block number, total transactions, transactions per second (TPS), and Layer 1 (L1) transaction batch data, provide users with crucial network statistics. The OP mainnet transaction history is also visually represented, making it easier to track network activity trends over 14 days. This structured layout ensures that users can quickly access and interpret essential blockchain information.



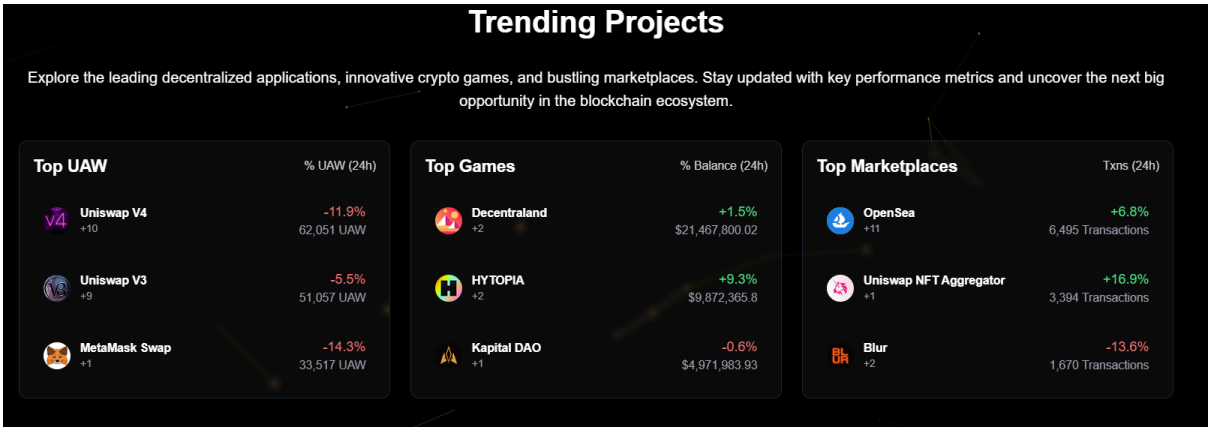
This screenshot showcases CryptoPath's homepage, highlighting a bold headline, "Your all-in-one crypto transaction explorer," and a concise subheader, "Vietnam's Premier Blockchain Explorer." A mobile preview on the right displays various crypto assets, while the call-to-action buttons ("Try CryptoPath," "Explore Markets," and "Discover NFTs") encourage immediate engagement. Logos at the bottom emphasize industry partnerships, reinforcing the platform's credibility.



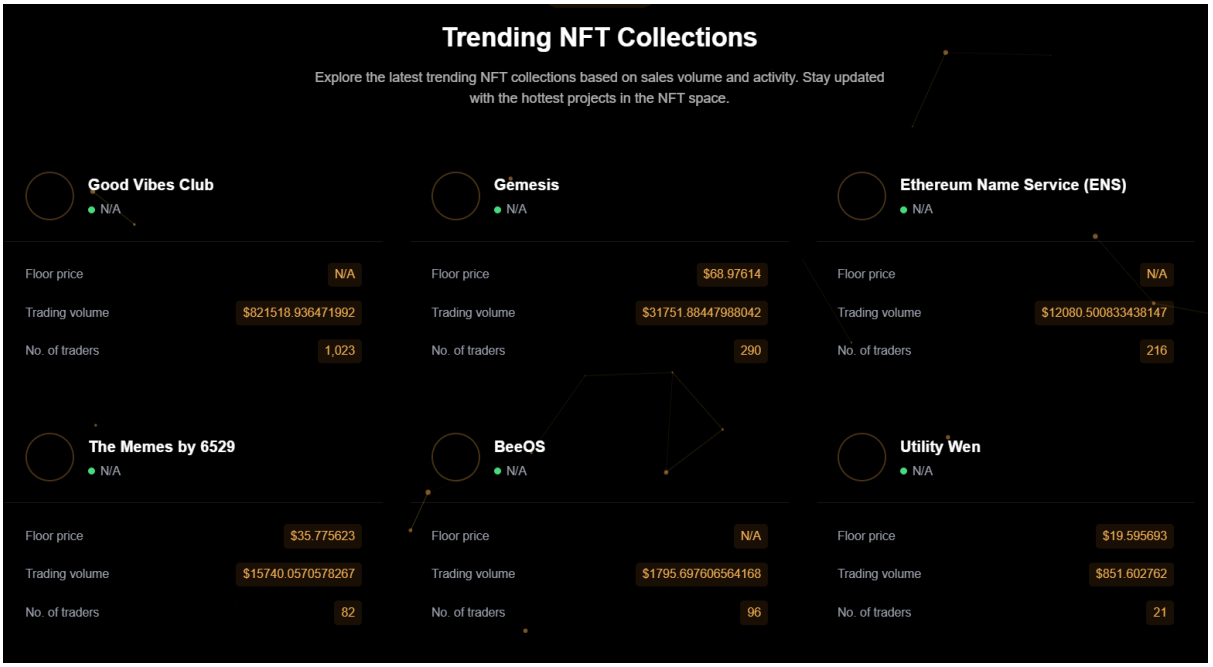
This section, titled “Explore Our Platform,” showcases the website's predominant features and introduces the “Crypto Price Dashboard.” A short tagline invites users to discover CryptoPath’s suite of blockchain tools, while “Explore” and “View Demo” buttons encourage deeper engagement.



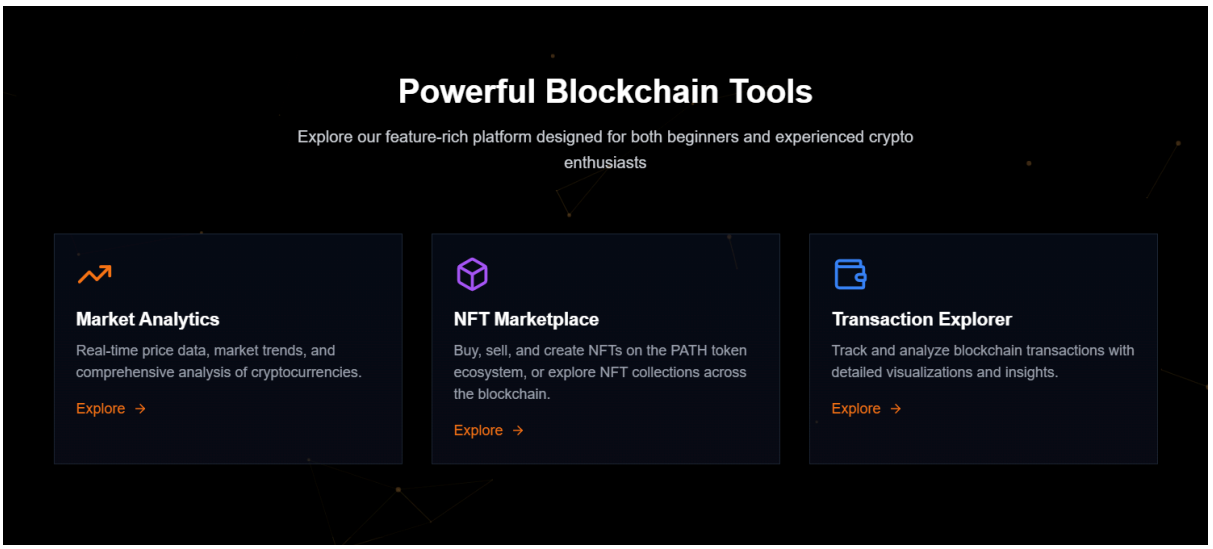
This section, titled “Track transactions like never before,” highlights real-time monitoring, analytics, and robust visualization tools



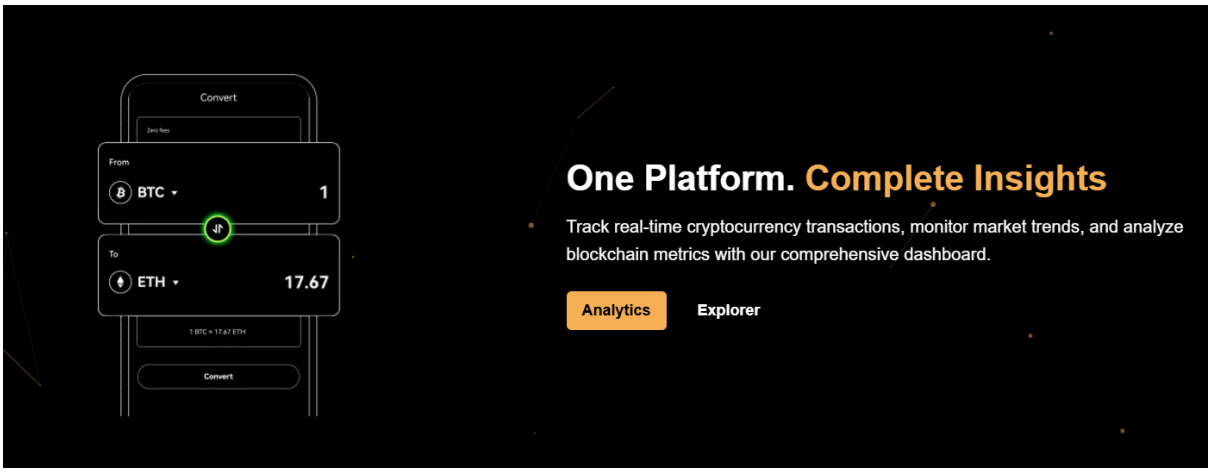
This Trending Projects section highlights decentralized applications, crypto games, and marketplaces. It tracks Top UAW (Unique Active Wallets), Top Games, and Top Marketplaces, showing 24h activity, user engagement, and transaction growth or decline, helping users identify emerging opportunities in the blockchain ecosystem.



This Trending NFT Collections section highlights top NFT projects based on sales volume and trader activity. Each collection displays floor price, trading volume, and number of traders, providing insights into the most active and valuable NFTs in the market..

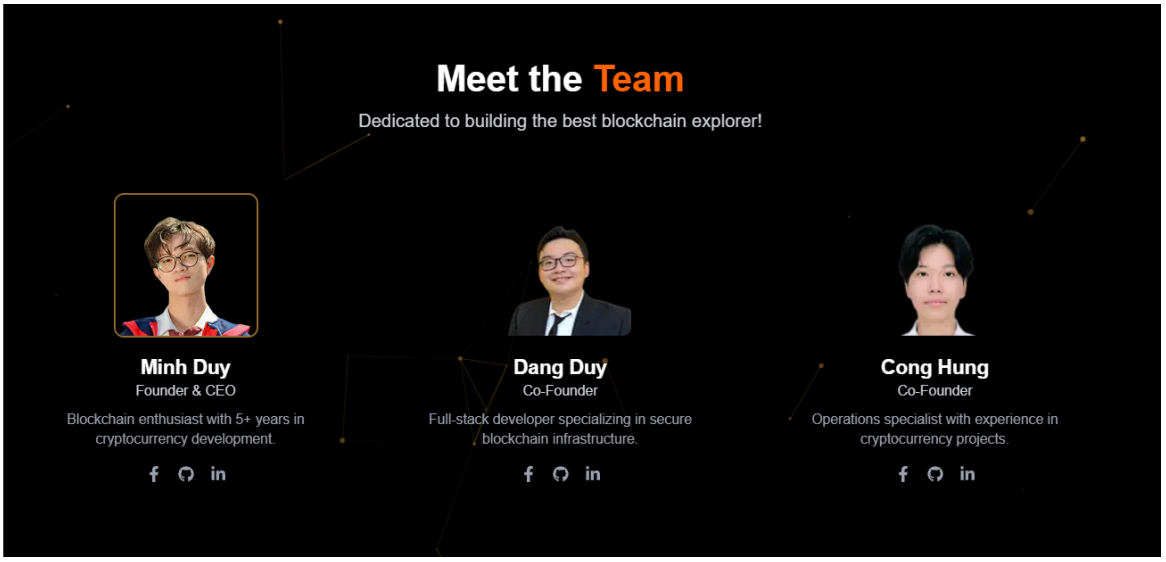


This “Powerful Blockchain Tools” section highlights three core offerings: Market Analytics, NFT Marketplace, and Transaction Explorer. Each block includes a brief description and an “Explore” button, inviting both new and experienced users to dive into real-time data, NFT trading, and blockchain transaction tracking.

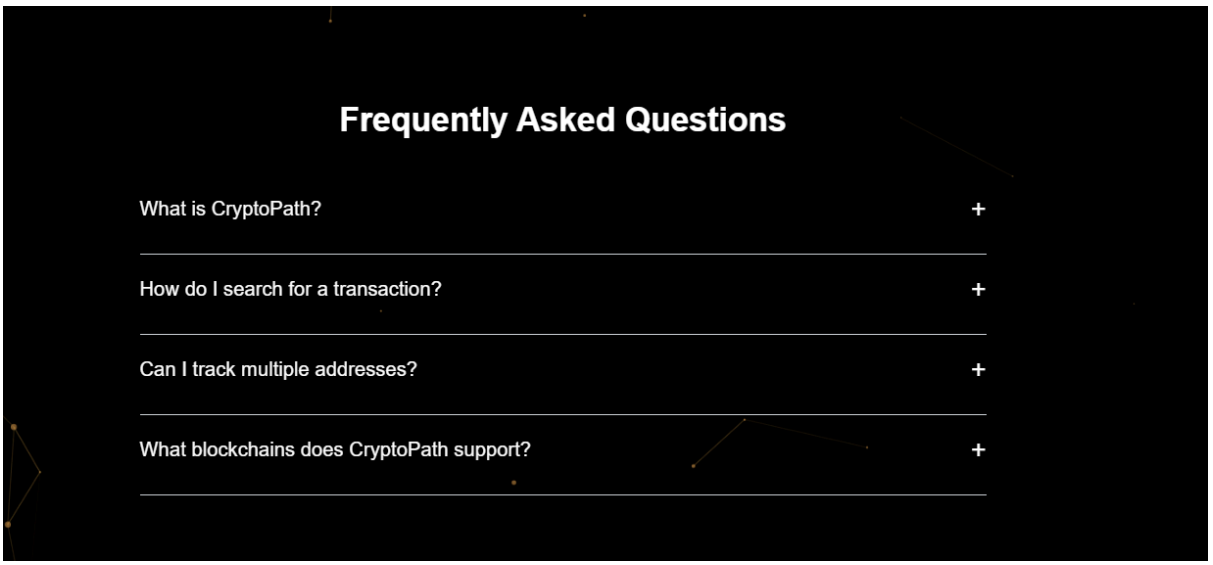


This section, titled “One Platform. Complete Insights,” features a BTC-to-ETH converter on a mobile interface. Two call-to-action buttons—“Analytics” and “Explorer”—prompt users to track transactions, monitor market trends, and analyze blockchain data in a single, comprehensive dashboard.

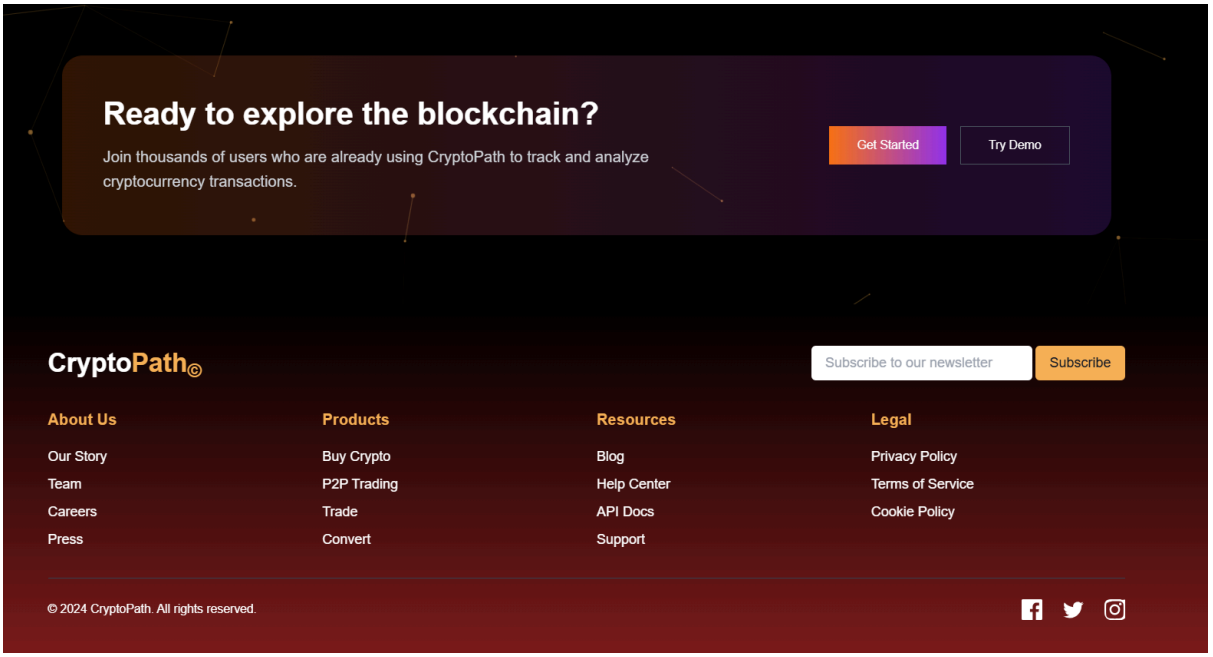
This section, titled “Your Gateway to Blockchain Data,” features an evolution-inspired illustration and highlights CryptoPath’s range of tools—from real-time transaction tracking to comprehensive market analysis. Users are encouraged to make informed, data-driven decisions about blockchain activity.



This “Meet the Team” section spotlights three core members—Minh Duy (Founder & CEO), Dang Duy (Co-Founder), and Cong Hung (Co-Founder)—each with specialized skills in blockchain, development, and operations. Their profiles emphasize the expertise driving CryptoPath’s mission to build a leading blockchain explorer.

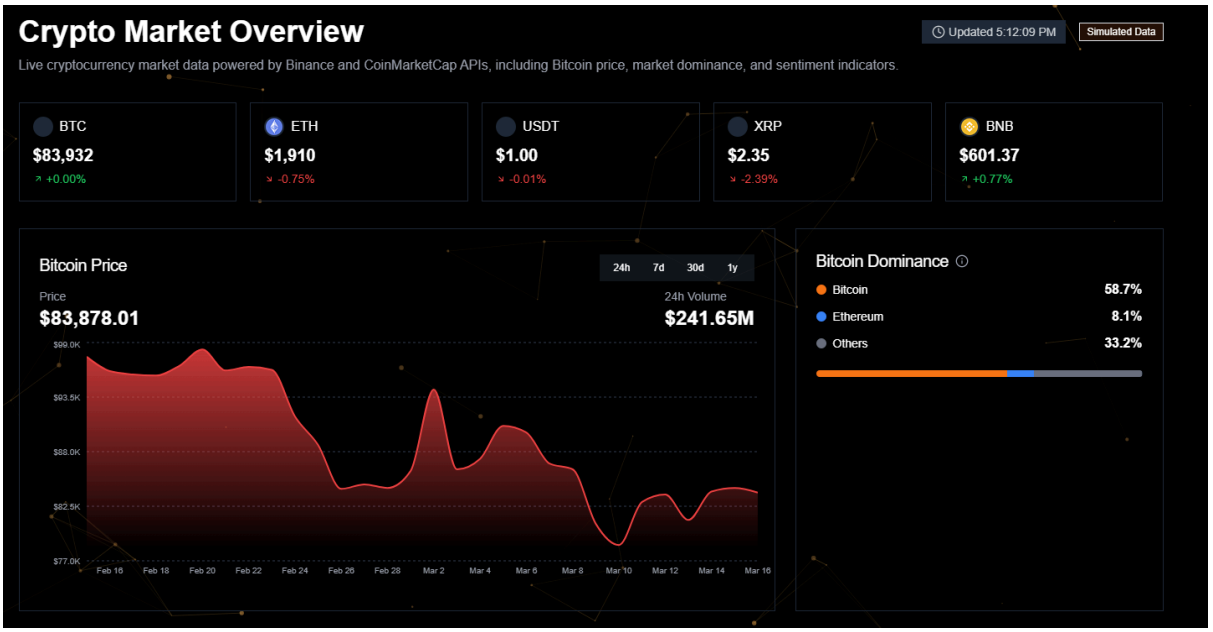


This “Frequently Asked Questions” section lists common queries about CryptoPath, covering its purpose, transaction search methods, multi-address tracking, and supported blockchains. Each question can be expanded for more detailed answers.

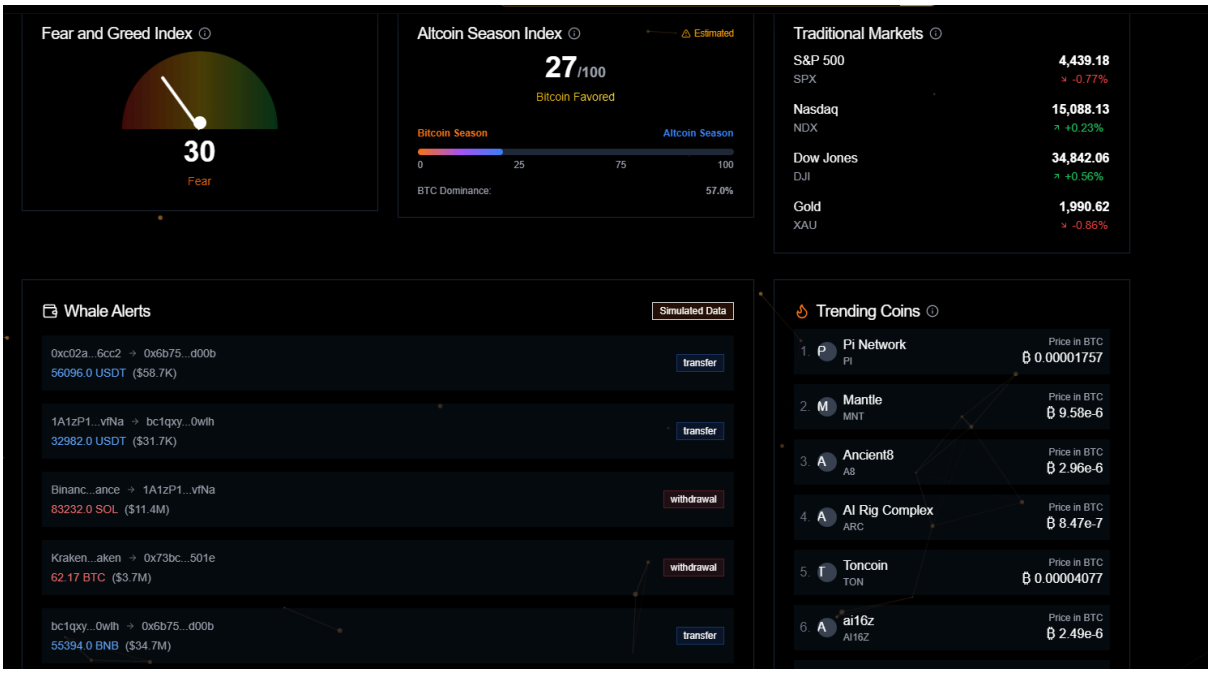


This final section invites users to explore the blockchain with a clear call-to-action—“Get Started” or “Try Demo.” A prominent banner encourages joining thousands of CryptoPath users. Below, the footer offers links to key pages (Our Story, Team, Careers, Press) and includes a newsletter signup, legal notices, and social media icons, rounding out the site’s comprehensive resources.

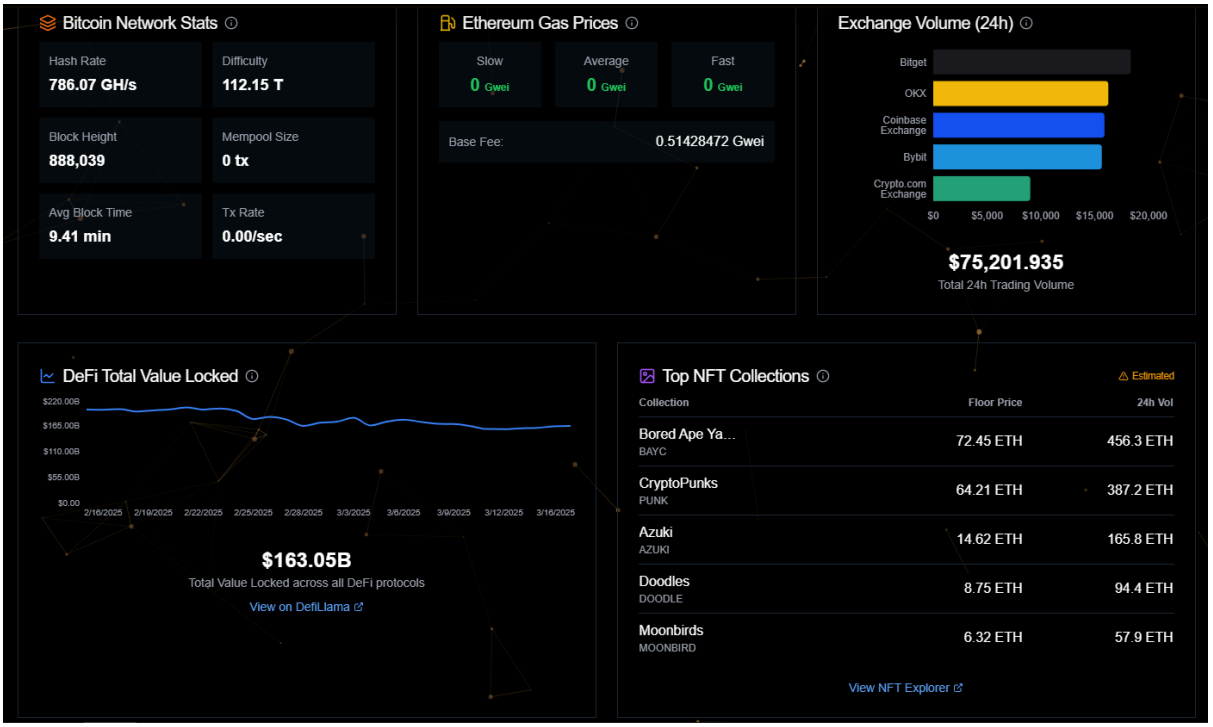
Market Overview Page:



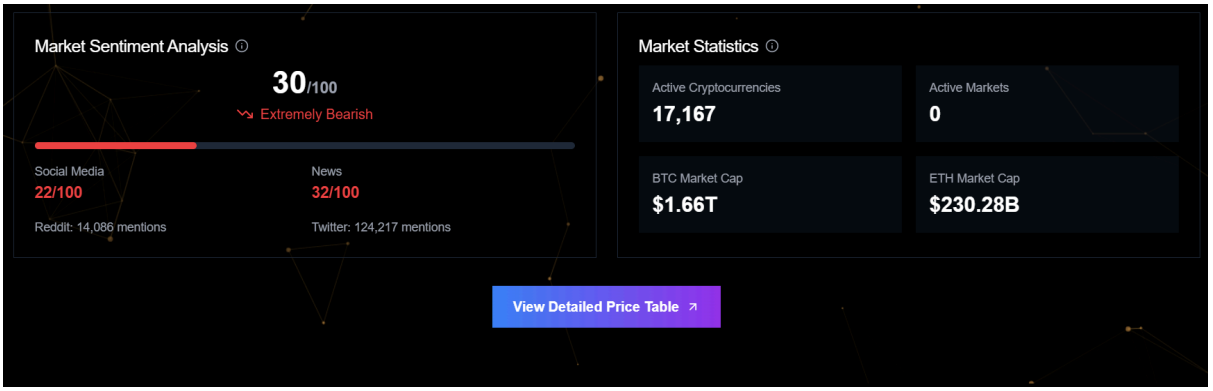
This “Crypto Market Overview” section displays real-time cryptocurrency data for major coins like BTC, ETH, USDT, XRP, and BNB, sourced from Binance and CoinMarketCap APIs. A Bitcoin price chart, 24-hour trading volume, and Bitcoin dominance metric provide a quick snapshot of overall market sentiment.



This section provides key market indicators, including the Fear and Greed Index and the Altcoin Season Index (Altcoin Favored). Traditional market performance (S&P 500, Nasdaq, Dow Jones, and Gold) is displayed alongside Whale Alerts, showing large Ethereum transfers. The Trending Coins panel lists top-performing cryptocurrencies based on recent price movements in BTC.



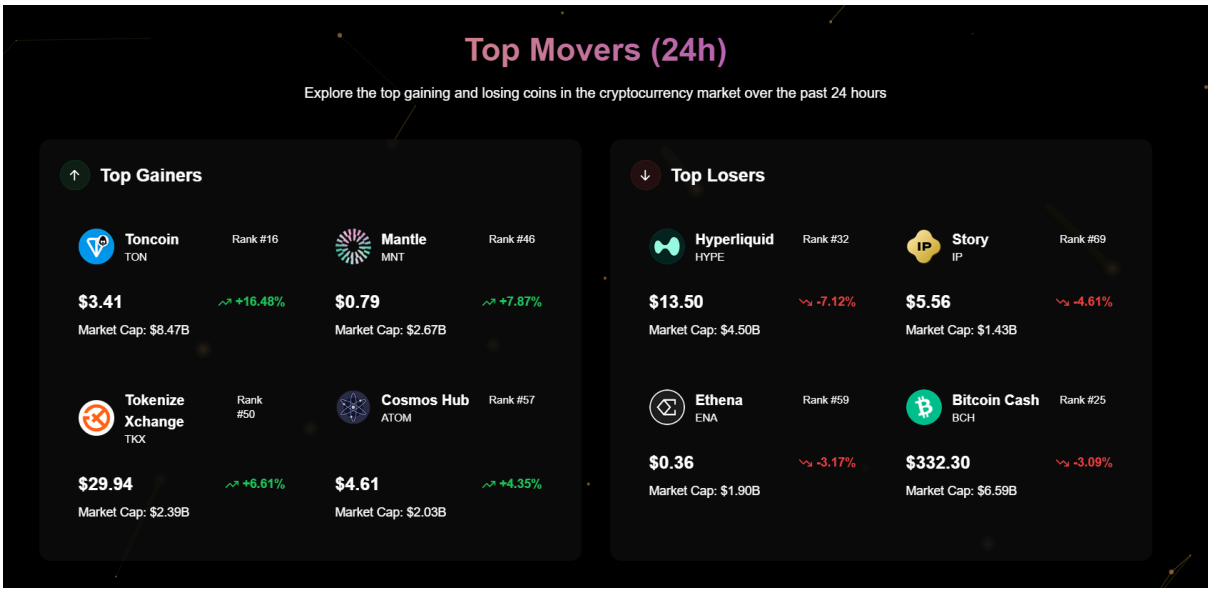
This dashboard presents key blockchain metrics, including Bitcoin Network Stats. while the Top NFT Collections section lists leading projects like Bored Ape Yacht Club.



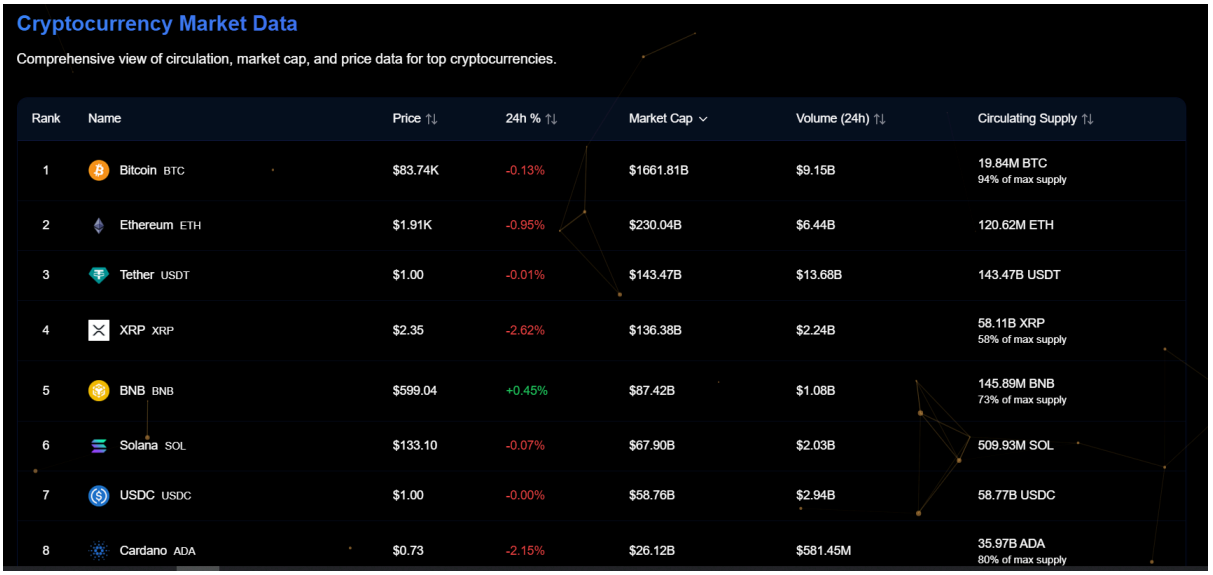
This section displays a Market Sentiment Analysis with sentiment from social media and news. The Market Statistics panel shows active cryptocurrencies, a BTC market cap, and an ETH market cap. A button links to a detailed price table for further insights.

PriceTable Page:

This section provides real-time cryptocurrency circulation data, displaying key market metrics: Total Market Cap (declining), 24h Trading Volume, and BTC Dominance. The last update time ensures users get the latest insights on market trends.

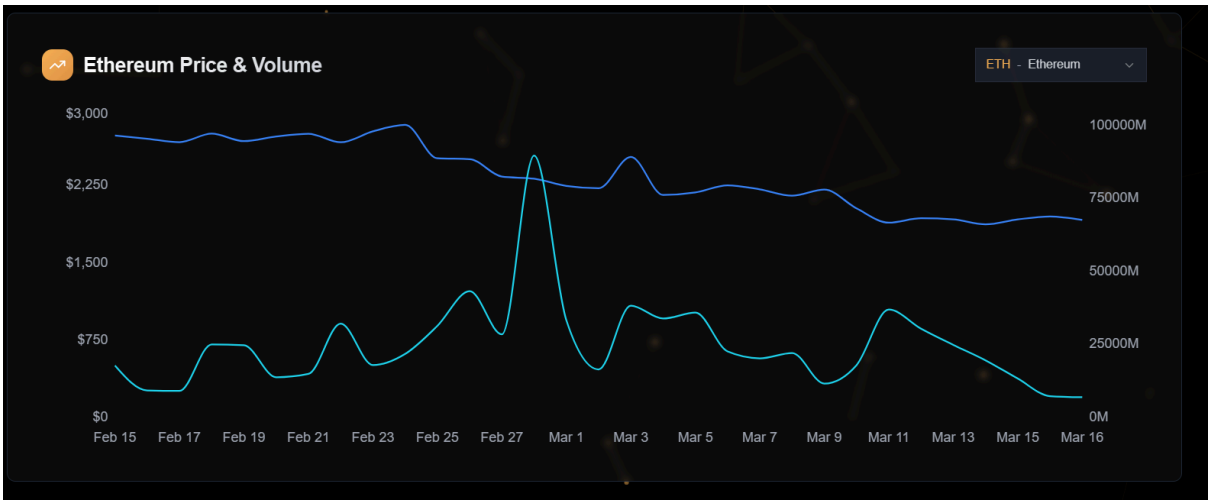


This section highlights the Top Movers (24h), showcasing the biggest gainers and losers in the cryptocurrency market. The Top Gainers list includes coins with strong upward momentum, while the Top Losers section tracks assets experiencing declines. Each entry displays price, market cap, and percentage change over the past 24 hours.

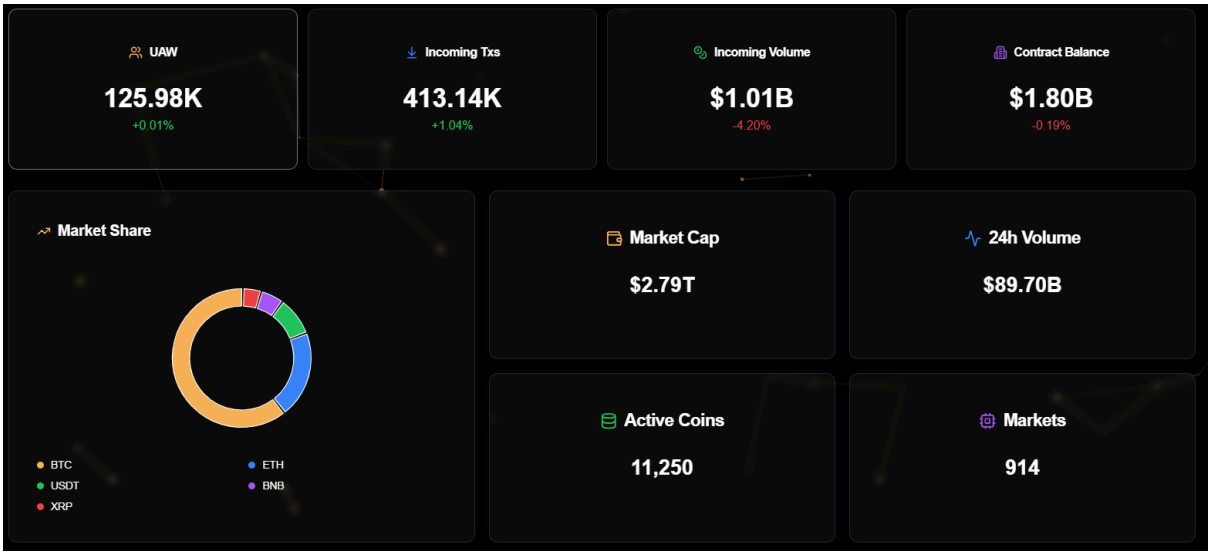


This section highlights the Top Movers (24h), showcasing the biggest gainers and losers in the cryptocurrency market. The Top Gainers list includes coins with strong upward momentum, while the Top Losers section tracks assets experiencing declines. Each entry displays price, market cap, and percentage change over the past 24 hours.

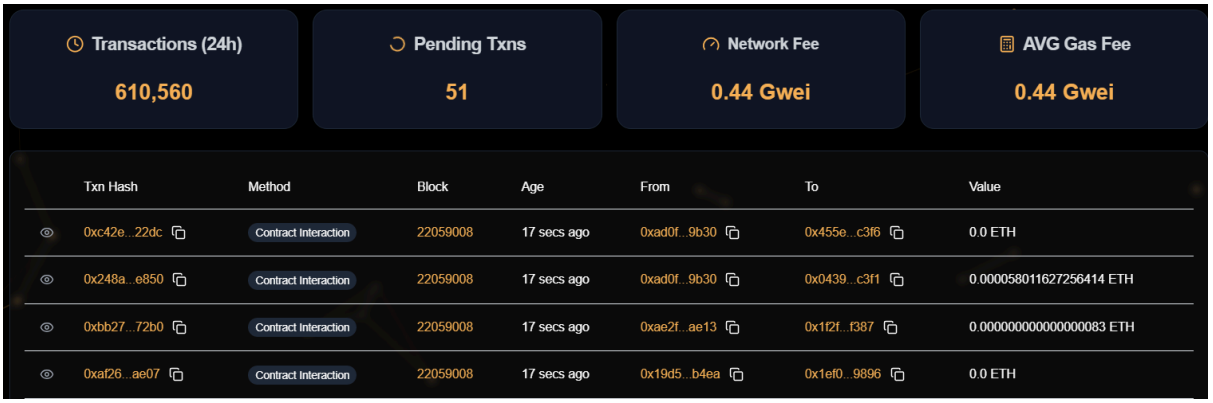
Transactions Page:



This chart displays Ethereum Price & Volume over time, showing fluctuations in ETH's market value and trading activity. The price trend remains relatively stable, while trading volume experiences notable spikes, indicating periods of increased market activity. This can change to other coins.

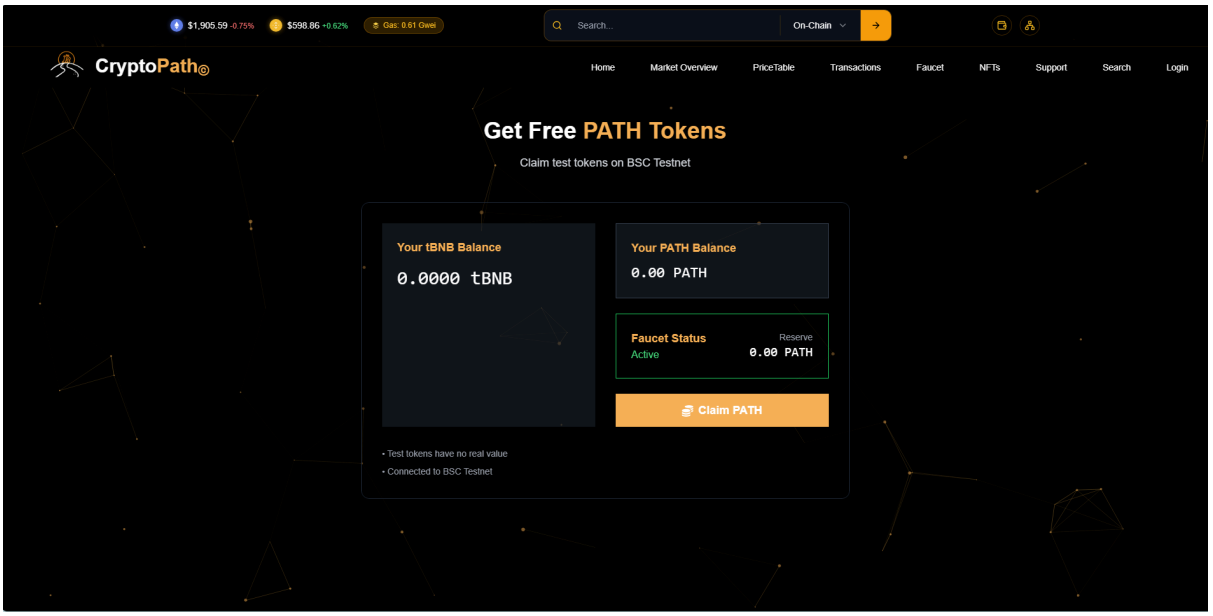


This dashboard provides an overview of blockchain activity, including active users, incoming transactions, and contract balances. It also displays market share distribution across major cryptocurrencies, total market cap, 24h trading volume, active coins, and markets, offering a snapshot of the current crypto landscape.



This section provides real-time blockchain transaction data, displaying 24h transaction count, pending transactions, network fees, and average gas fees. Below, a transaction table lists recent contract interactions, including transaction hashes, blocks, sender and recipient addresses, and transaction values.

Faucet Page:



This faucet page allows users to claim free PATH tokens on the BSC Testnet, ensuring they have sufficient tokens for testing transactions within the CryptoPath ecosystem.

The headline—"Get Free PATH Tokens"—immediately conveys the page's purpose, while the supporting text clarifies that these are test tokens with no real value..

A large, bright "Claim PATH" button acts as the primary call-to-action (CTA), encouraging users to claim their free tokens effortlessly.

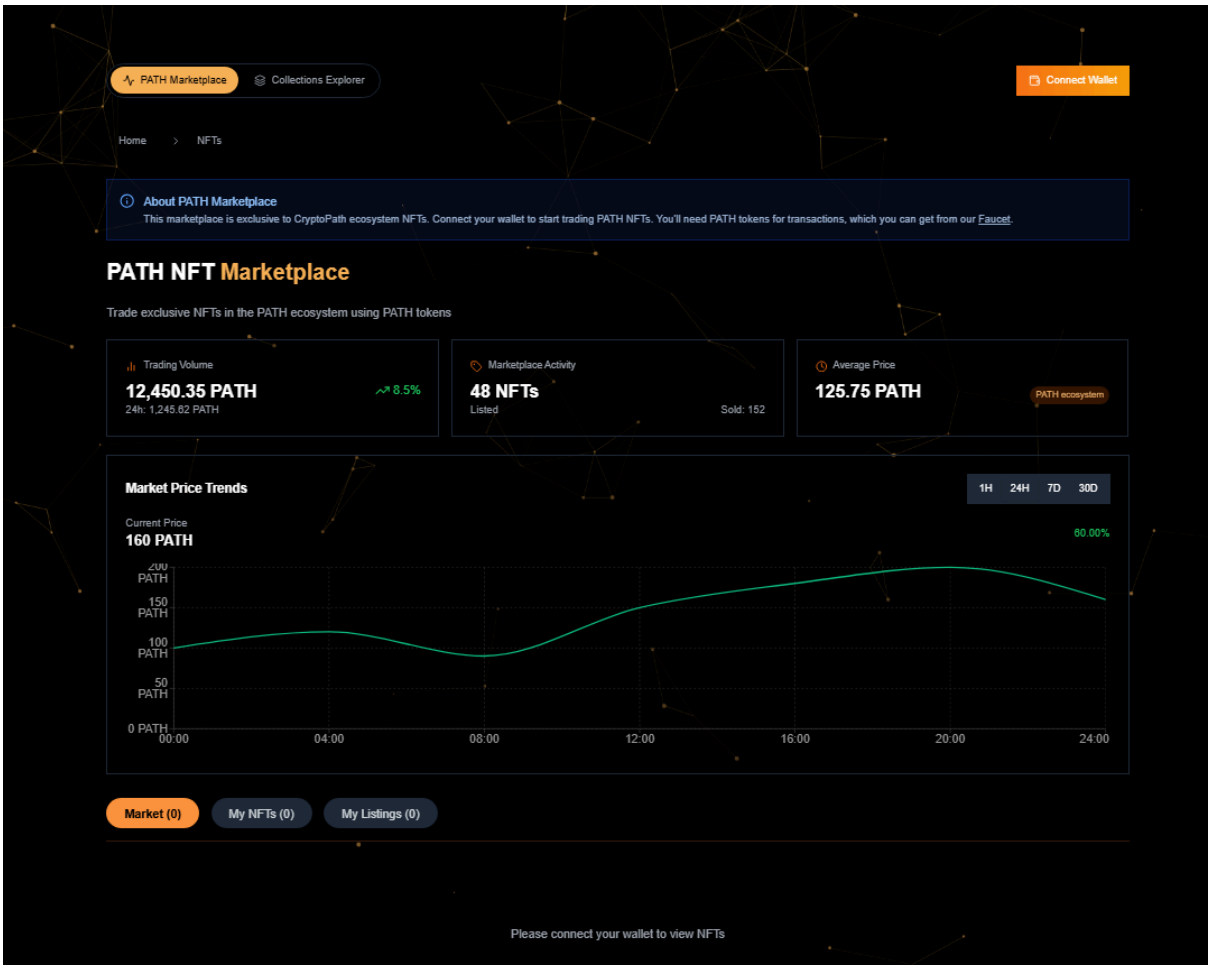
The footer provides additional notes:

- Test tokens have no real value, ensuring users understand their purpose.
- Connected to BSC Testnet, confirming the correct network for transactions.

This clear, user-friendly interface ensures that CryptoPath users can quickly and easily acquire the tokens they need for testing on the BSC Testnet.

NFT Page:

PATH Marketplace:



This NFT marketplace dashboard provides a streamlined interface for users to trade exclusive NFTs within the CryptoPath ecosystem. At the top, a notification banner labeled "About PATH Marketplace" briefly introduces the platform, emphasizing the requirement for PATH tokens for transactions and guiding users to the Faucet if needed. A "Connect Wallet" button in the top right corner serves as a clear entry point for users to interact with the marketplace.

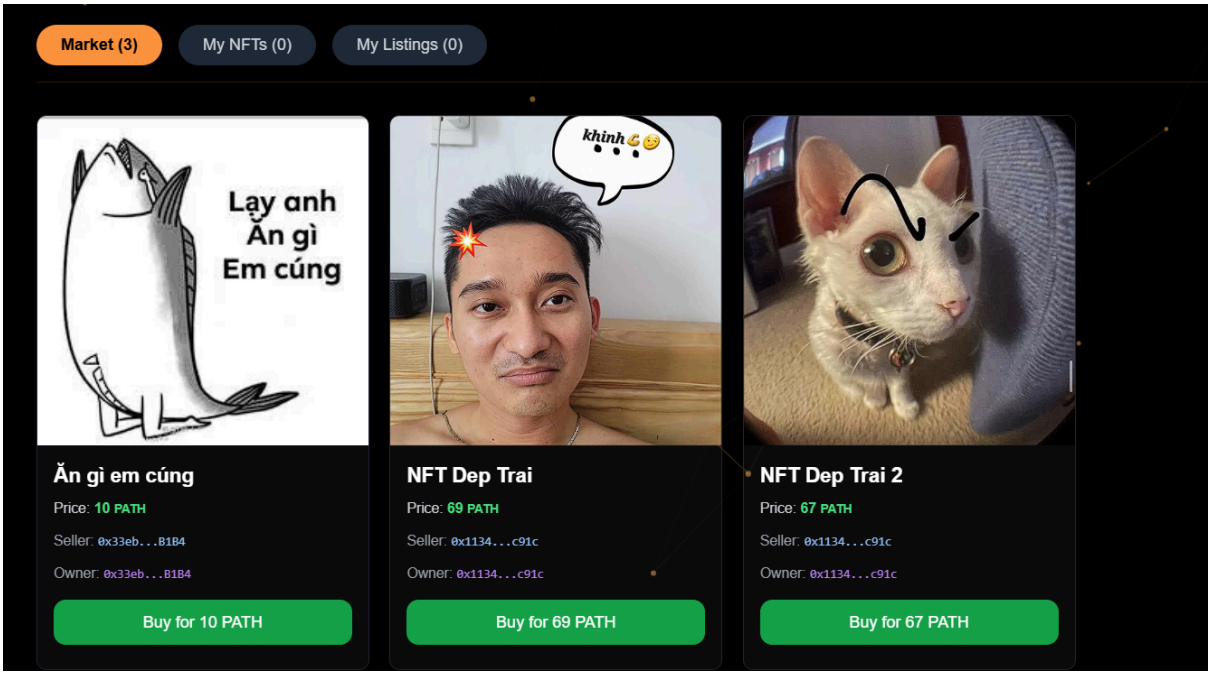
The dashboard displays key market insights in distinct sections:

- Trading Volume: 12,450.35 PATH, with a 24-hour volume of 1,245.82 PATH and an 8.5% increase.
- Marketplace Activity: 48 NFTs listed, with 152 sales completed.
- Average Price: 125.75 PATH, giving users an instant view of market valuation.

A Market Price Trends chart visualizes price fluctuations, with a current price of 160 PATH. Users can switch between timeframes (1H, 24H, 7D, 30D) to analyze market performance over different periods.

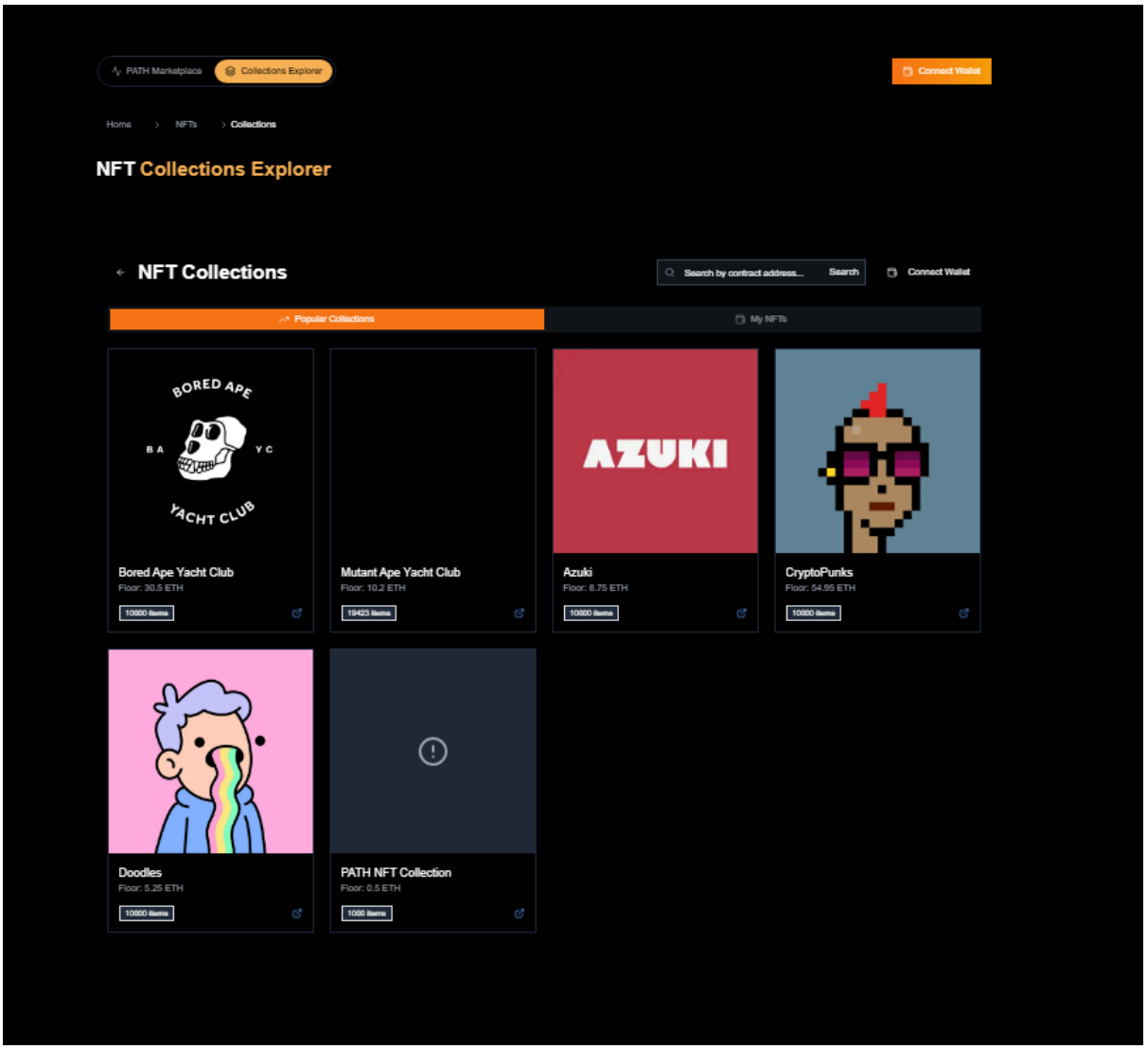
Below the chart, interactive navigation options allow users to explore:

- Market (for available NFTs)
- My NFTs (owned assets)
- My Listings (NFTs listed for sale)



This NFT Marketplace section displays available NFTs for purchase. Each listing includes an image, name, price in PATH tokens, seller, and owner details. Users can browse, buy NFTs, and manage their collections through the Market, My NFTs, and My Listings tabs.

Collection Explorer:



The NFT Collections Explorer page provides users with a structured interface to browse and discover popular NFT collections available on the CryptoPath marketplace. The dark-themed design, accented with gold and white highlights, maintains a sleek and modern aesthetic, ensuring a smooth browsing experience.

At the top, a navigation breadcrumb allows users to track their location within the marketplace (Home > NFTs > Collections), while a "Connect Wallet" button is prominently placed for seamless user engagement.

The page features a two-tab navigation system:

- Popular Collections (highlighted in orange) showcasing trending NFTs.
- My NFTs, allowing users to view their owned assets upon connecting a wallet.

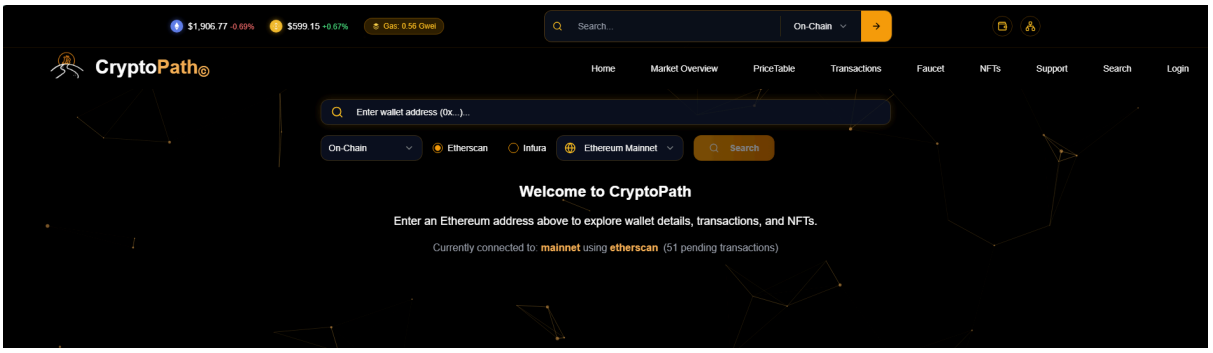
A search bar enables users to find collections via contract address, enhancing search efficiency.

The NFT collections grid displays six prominent collections, each with:

- Collection Name (e.g., Bored Ape Yacht Club, Azuki, CryptoPunks).
- Floor Price (e.g., 30.5 ETH for Bored Ape Yacht Club).
- Total Items (e.g., 10,000 items per collection).

- Collection Image for easy identification.
- External link icon, allowing users to explore collections further.

Search Page:



The CryptoPath Search Page allows users to explore wallet details, transactions, and NFTs by entering an Ethereum wallet address. The design features a dark-themed interface with gold and white accents, ensuring a sleek and professional blockchain analytics experience.

At the top, the navigation bar provides quick access to key sections, including Home, Market Overview, Price Table, Transactions, Faucet, NFTs, Support, Search, and Login.

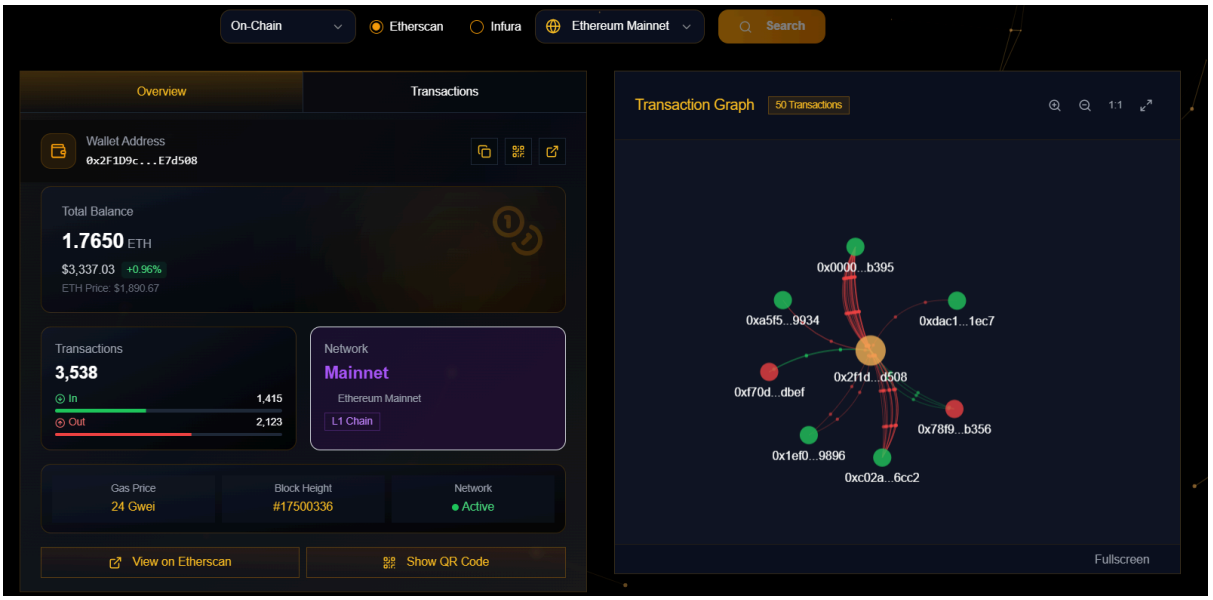
The central focus is the wallet search bar, where users can enter a wallet address (0x...) to retrieve relevant blockchain data. Accompanying this is a dropdown menu for selecting the network source, including:

- On-Chain (default)
- Off-chain (Neo4j)
- Etherscan
- Infura
- Ethereum Mainnet

The current network status is displayed, confirming the connection to Ethereum Mainnet via Etherscan, with 306 pending transactions, ensuring real-time transparency on blockchain activity.

This well-structured and intuitive search page provides a seamless experience for users looking to explore Ethereum wallet data efficiently.

Search Result:



The CryptoPath search results page provides an in-depth view of a wallet's activity, including balances, transactions, and token holdings. With a sleek dark interface and gold accents, the layout ensures clarity while presenting complex blockchain data in an accessible way.

A visual transaction graph on the right dynamically maps out the wallet's interactions, showing connections between addresses. This provides a clear representation of transaction history and fund movements, making it easier to track interactions and detect patterns.

Portfolio Section

Below, the Portfolio panel lists the wallet's assets, supporting multi-chain analysis. Users can:

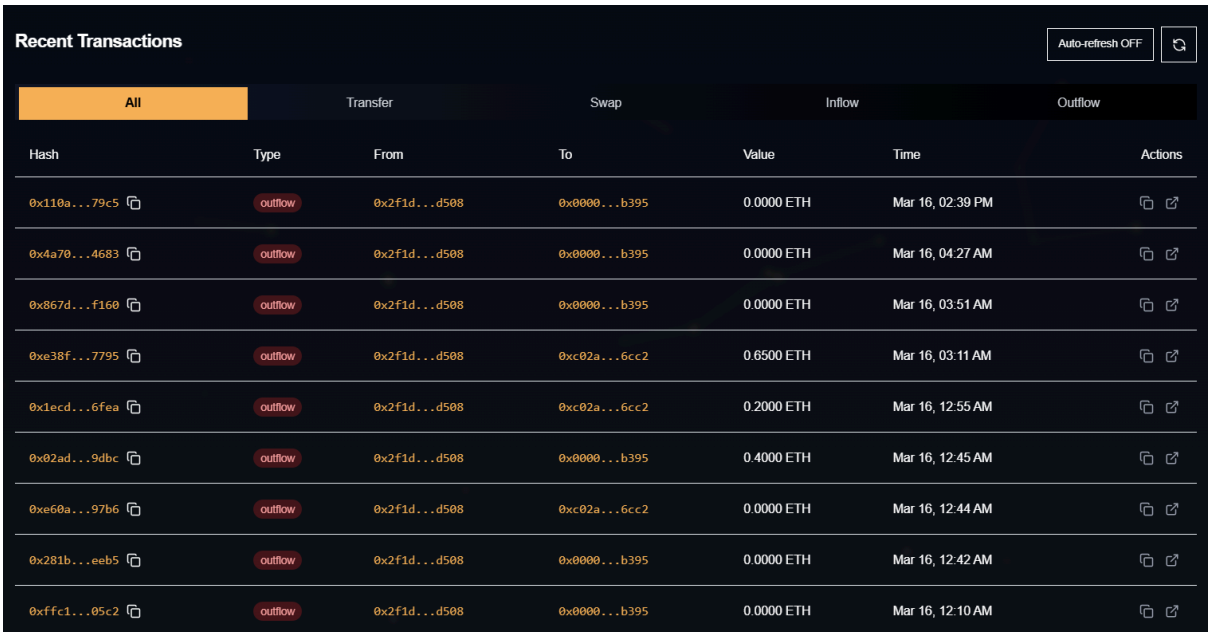
- Filter by provider (e.g., Alchemy).
- Toggle the display of zero-balance assets.
- Sort by chain for cross-network visibility.

Each token entry displays:

- The asset's balance.
- The blockchain it belongs to.
- An external link for verification if available.

The portfolio layout is designed for quick reference, ensuring users can track their assets and evaluate holdings at a glance.

To improve usability, the page includes pagination controls, allowing seamless browsing through multiple pages of token holdings. This is especially useful for wallets with a high number of assets spread across various chains.



The Recent Transactions section provides a detailed history of blockchain interactions for the searched wallet. Designed with a sleek dark interface, this table ensures that users can easily track activity across multiple transaction types.

At the top, users can filter transactions based on type, including:

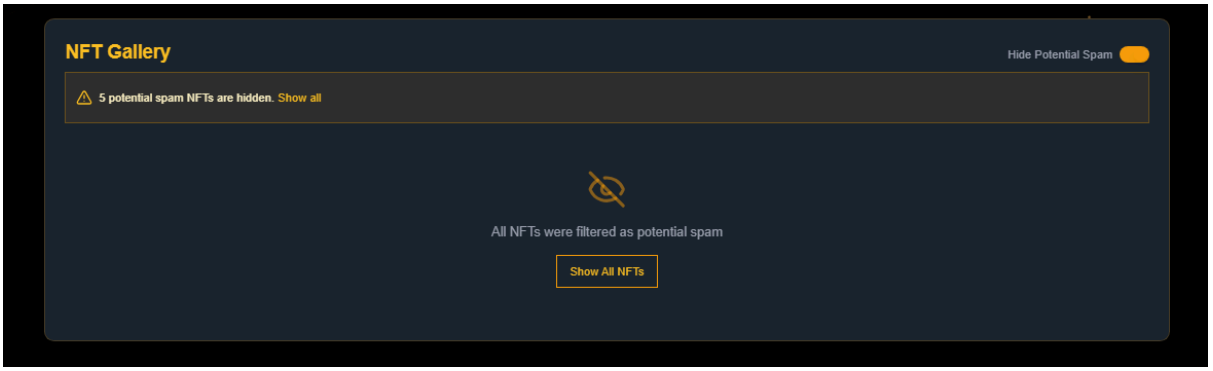
- All (default view displaying all transactions).
- Transfer (wallet-to-wallet transactions).
- Swap (token exchange activities).
- Inflow (incoming funds).
- Outflow (outgoing funds).

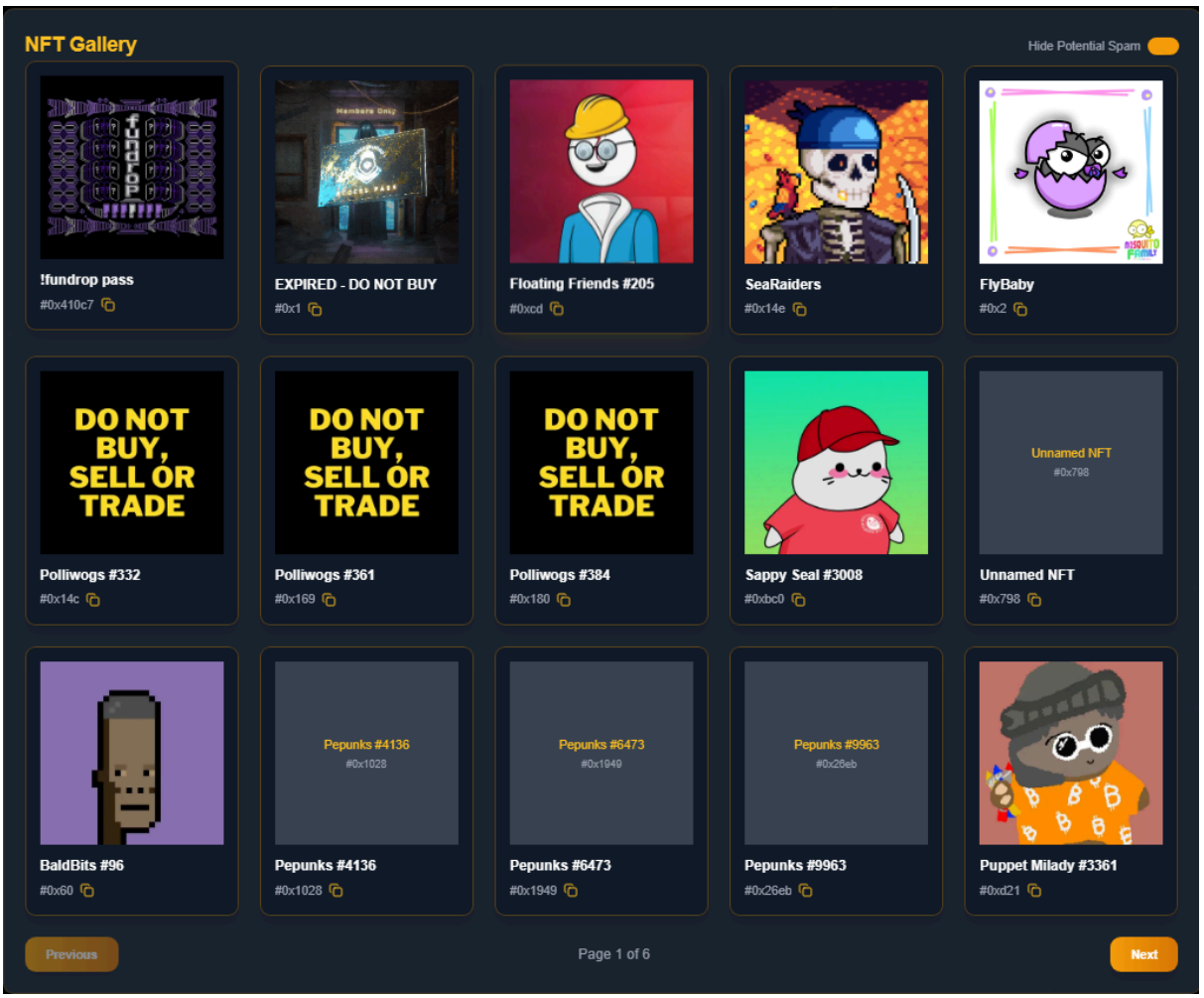
This categorization simplifies transaction tracking and allows users to focus on relevant activities.

Each row in the table presents key details of an individual transaction, including:

- Sender Address ("From")
- Receiver Address ("To")
- Transaction Value (in ETH)
- Timestamp (formatted for easy reading)

The structured layout ensures that users can quickly scan through activity logs, helping to analyze movement patterns and detect any irregularities.





The NFT Gallery provides a structured, secure, and user-friendly way to explore NFTs linked to a wallet.

Spam Filtering & Visibility Control

- Auto-filters potential spam NFTs with an option to show or hide flagged assets.
- If all NFTs are marked as spam, a "Show All NFTs" button allows manual review.

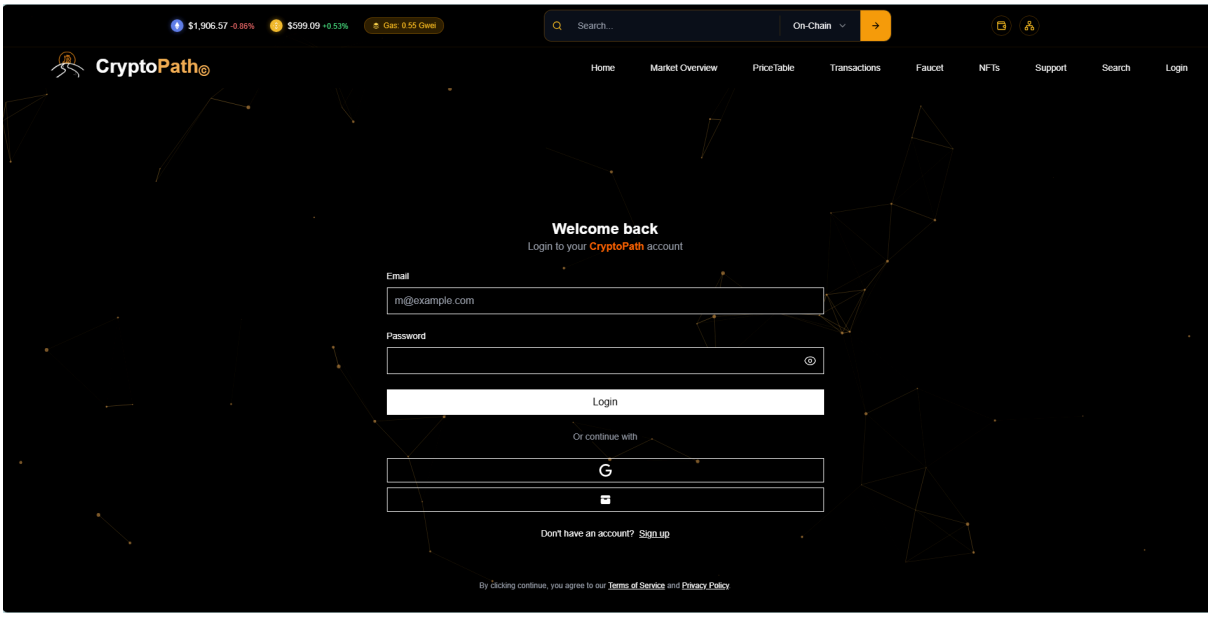
NFT Display & Collection Overview

- NFTs are shown with images, names, and contract addresses for easy identification.
- Contract address links enable verification on blockchain explorers.

Navigation & Accessibility

- Pagination controls allow smooth browsing through large collections.
- Clear layout and evenly spaced NFTs make finding assets effortless.

Login Page:



6. Core Functionalities

6.1 User Authentication and Session Management

- Secure login and logout using Supabase Auth.
- Persistent user sessions stored in local storage for seamless navigation.

6.2 Wallet and Portfolio Management

- View wallet balances, transaction history, and token holdings.
- Manage portfolios with real-time updates and detailed analytics.

6.3 Blockchain Data Visualization

- Interactive transaction graphs for tracing paths and relationships.

- Real-time market data visualization, including price trends and market caps.

6.4 Market Insights and Analytics

- Access detailed market overviews, price tables, and trending tokens.
- Analyze token performance, market dominance, and sentiment indicators.

6.5 NFT Marketplace

- Buy, sell, and mint NFTs within the PATH token ecosystem.
- Explore collections and manage owned NFTs with intuitive tools.

6.6 Faucet Integration

- Claim free PATH tokens on the BSC Testnet for testing and transactions.

6.7 Search and Explore

- Search for wallet addresses, tokens, and transactions with advanced filters.
- Explore blockchain data with detailed visualizations and insights.

6.8 Support and Accessibility

- Responsive UI support.
- Email support and comprehensive navigation for all user levels.

7. Backend Overview

The backend of the CryptoPath project includes several key components and functionalities:

7.1 RESTful API for Blockchain Data Retrieval

- Provides endpoints to fetch blockchain data, including Ethereum wallet balances, transaction histories, and token holdings.
- Integrates with multiple providers like Etherscan, Infura, and Alchemy to ensure reliable data access
- Utilizes data transformation and caching mechanisms to optimize performance and reduce API call overhead..

7.2 Data Transformation & Caching

- Transforms raw blockchain data into structured formats for visualization, such as transaction graphs and portfolio summaries.
- Implements caching strategies using in-memory and persistent storage to minimize API usage and improve response times.

7.3 Authentication & Rate Limiting

- Incorporates Supabase authentication to ensure secure user access and session management.
- Implements rate limiting and error handling to prevent abuse and manage API usage efficiently across multiple providers.

7.4 Graph Database Integration

- Uses Neo4j for mapping transaction relationships and tracing wallet interactions.
- Optimized queries enable efficient path discovery and transaction graph generation.
- Leverages caching and indexing to enhance performance for large-scale data lookups.

7.5 API Integration

- The system connects to Ethereum blockchain APIs, including **Etherscan API** (Etherscan, n.d.), **Infura API** (Infura, n.d.), **Alchemy API** (Alchemy, n.d.), and **Moralis API** (Moralis, n.d.), to fetch transaction data, smart contract interactions, and historical records.
- WebSocket technology is leveraged for real-time updates, ensuring efficient event-driven communication. APIs such as **CoinGecko API** (CoinGecko, n.d.), **Binance API** (Binance, n.d.), **DefiLlama API** (DefiLlama, n.d.), and **DappRadar API** (DappRadar, n.d.) are integrated to track live market data, DeFi trends, and token price fluctuations.
- The **Supabase Auth API** (Supabase, n.d.) is used for secure authentication, allowing seamless user management and access control.
- For user engagement and system alerts, **Nodemailer API** (Nodemailer, n.d.) is implemented to enable robust email communication.
- Stringent error handling mechanisms and data validation protocols ensure high data integrity and reliability. The integration of **CoinMarketCap API** (CoinMarketCap, n.d.) enhances accuracy in pricing data and market analytics.

7.6 Real-Time Notifications

- Implements WebSocket connections to provide real-time updates on wallet transactions and market trends.
- Allows users to subscribe to specific events, such as token price changes or wallet activity.

7.7 NFT Marketplace Integration

- Supports buying, selling, and minting NFTs using PATH tokens.
- Provides detailed NFT metadata and collection statistics for enhanced user experience.
- Integrates with IPFS for decentralized storage of NFT assets.

7.8 User Personalization

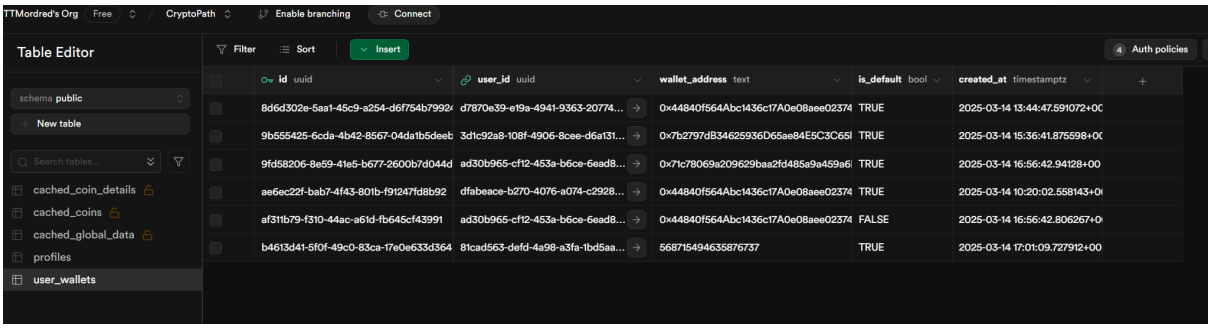
- Offers customizable settings for themes, notification preferences, and portfolio tracking.
- Synchronizes user profiles and wallet data across devices using Supabase.
- Provides a seamless experience with light/dark mode and responsive design.

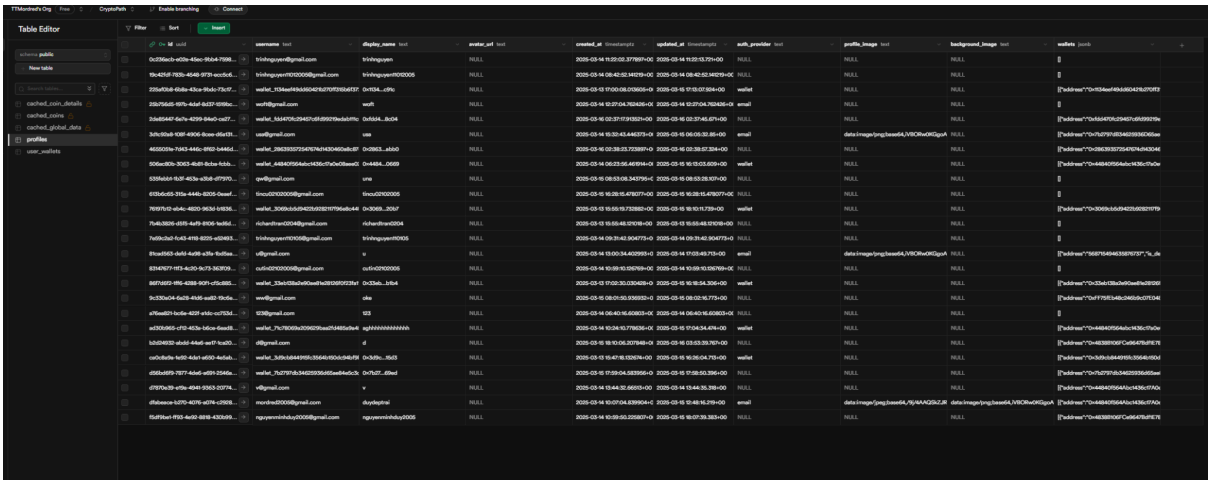
7.9 Supabase Details

This Supabase database schema includes multiple tables for cached cryptocurrency data, user profiles, and wallet management. Key tables:

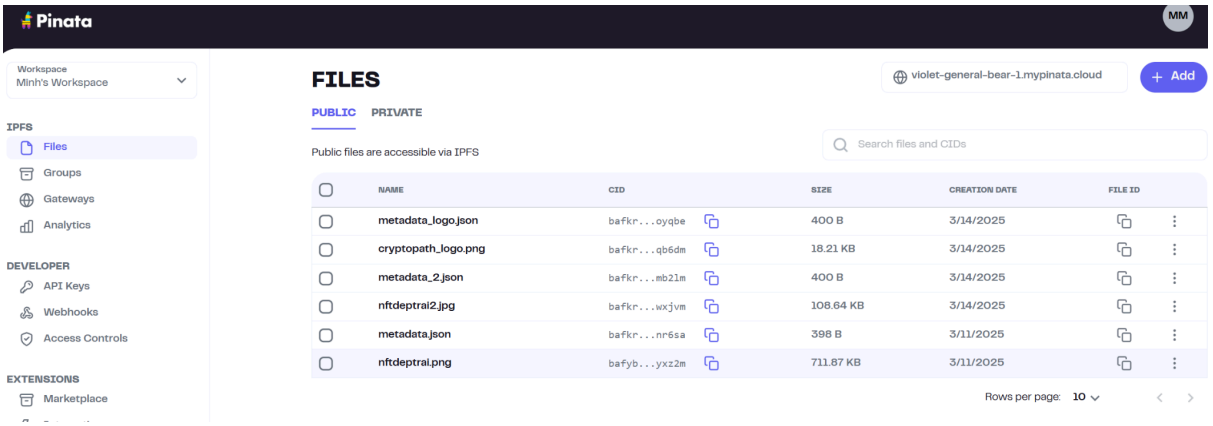
- cached_coins, cached_global_data, cached_coin_details: Store cryptocurrency-related data in JSON format with timestamps.
- profiles: Manages user information like username, avatar, profile image, and wallets, linked to auth.users.id.
- user_wallets: Stores wallet addresses, linking them to users via auth.users.id and indicating default wallets.

The schema enables user authentication, profile management, and cryptocurrency tracking within the system.





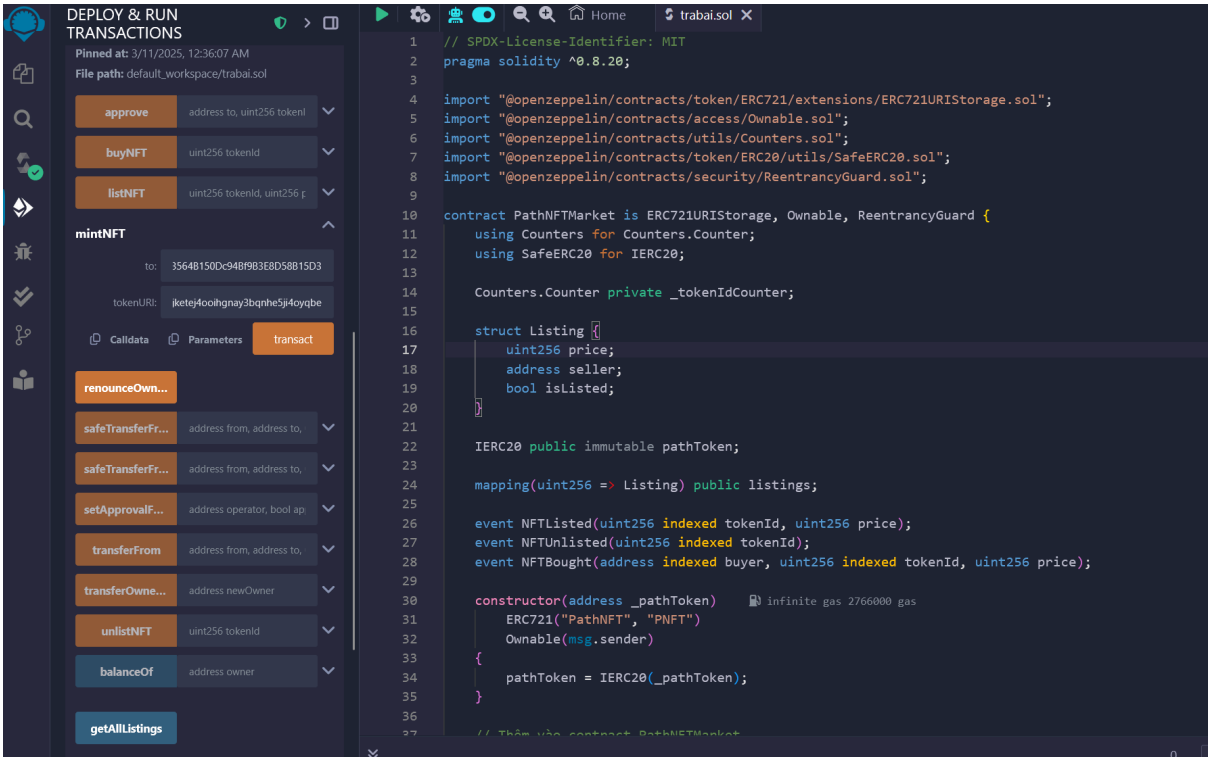
7.10 Data Storage



Our **Pinata IPFS** database serves as a decentralized storage system, securely hosting and managing publicly available files on the InterPlanetary File System (IPFS). It enables efficient organization, retrieval, and verification of digital assets while ensuring permanent and tamper-proof storage. Each stored file is assigned a unique Content Identifier (CID), along with metadata such as file size and creation date, allowing seamless access through IPFS gateways. This database architecture provides a scalable, resilient, and censorship-resistant solution for managing

distributed data. Users can upload, manage, and retrieve decentralized assets, ensuring permanent, tamper-proof storage.

7.11 Smart Contract



This **Remix IDE** shows a Solidity smart contract for an NFT marketplace using ERC721, ERC20, and OpenZeppelin libraries. The contract includes:

- Listing structure (price, seller, listing status).
- Mapping for token listings.
- Events for listing, unlisting, and buying NFTs.
- Functions for minting, buying, and managing NFTs.

The left panel displays Remix's Deploy & Run Transactions module, where users can:

1. Deploy the contract with the provided constructor.
2. Mint an NFT by assigning a token URI.
3. List, buy, and manage NFTs through available contract functions.

This setup allows deploying and testing NFT contracts on Ethereum or compatible blockchains.

8. Backend Code

```
// Contract addresses
const NFT_CONTRACT_ADDRESS = "0x279Bd9304152E0349427c4B7F35FffFD439edcFa";
const PATH_TOKEN_ADDRESS = "0xc3e9Cf26237c9002c0C04305D637AEa3d9A4A1DE";
const ITEMS_PER_PAGE = 8;

// ABI definitions
const NFT_ABI = [
  "function totalSupply() view returns (uint256)",
  "function ownerOf(uint256 tokenId) view returns (address)",
  "function tokenURI(uint256 tokenId) public view returns (string memory)",
  "function getAllListings() external view returns (uint256[] memory)",
  "function listings(uint256) view returns (uint256 price, address seller, bool isListed)",
  "function buyNFT(uint256 tokenId) external",
  "function listNFT(uint256 tokenId, uint256 price) external",
  "function unlistNFT(uint256 tokenId) external"
];

const TOKEN_ABI = [
  "function approve(address spender, uint256 amount) returns (bool)",
  "function allowance(address owner, address spender) view returns (uint256)",
  "function balanceOf(address account) external view returns (uint256)"
];
```

This NextJS snippet defines contract addresses and ABI (Application Binary Interface) for an NFT marketplace. Key components:

1. Contract Addresses:
    - NFT_CONTRACT_ADDRESS: Specifies the deployed NFT contract.
    - PATH_TOKEN_ADDRESS: Defines the ERC20 token contract used for transactions.
2. NFT ABI (Application Binary Interface):
    - Includes view functions for total supply, ownership, and token metadata (tokenURI).
    - Implements external functions for listing, buying, and unlisting NFTs.
3. Token ABI:
    - Contains standard ERC20 functions like approve, allowance, and balanceOf for token management.

This setup allows frontend applications to interact with smart contracts, enabling NFT transactions and token-based operations on a blockchain network.

```
// Handle List NFT
const handleListNFT = async (tokenId: string, price: string) => {
  if (!account) {
    alert("Please connect wallet first!");
    return;
  }

  setProcessing(true);
  try {
    const provider = getProvider();
    const signer = provider.getSigner();
    const contract = new ethers.Contract(NFT_CONTRACT_ADDRESS, NFT_ABI, signer);

    // Validate ownership
    const owner = await contract.ownerOf(tokenId);
    if (owner.toLowerCase() !== account.toLowerCase()) {
      throw new Error("You are not the owner");
    }

    const tx = await contract.listNFT(
      tokenId,
      ethers.utils.parseUnits(price, 18)
    );
    await tx.wait();
    await refreshData();
  } catch (error) {
    console.error("Listing failed:", error);
    alert(error instanceof Error ? error.message : "Unknown error");
  } finally {
    setProcessing(false);
  }
};
```

This NextJS function allows users to list their NFTs for sale on a blockchain marketplace using Ethers.js.

How it works:

1. Wallet Connection Check:
   - If the user hasn't connected their wallet, an alert prompts them to do so.
2. Set Processing State:
   - setProcessing(true); indicates that the function is in progress.
3. Blockchain Interaction:
   - Retrieves a provider and signer from getProvider().
   - Connects to the NFT contract using ethers.Contract.
4. Ownership Validation:
   - Calls contract.ownerOf(tokenId) to check if the connected user owns the NFT.
   - If the user is not the owner, an error is thrown.
5. List NFT on Marketplace:
   - Calls contract.listNFT(tokenId, ethers.utils.parseUnits(price, 18)) to list the NFT for sale at the specified price.
   - Waits for the transaction to be confirmed.
6. Refresh Data & Error Handling:
   - Calls refreshData() after a successful listing.

- Catches and logs any errors, alerting the user if something goes wrong.

```
// Handle buy NFT
const handleBuyNFT = async (tokenId: string, price: string) => {
  if (!account) {
    alert("Please connect wallet!");
    return;
  }

  setProcessing(true);
  try {
    const provider = getProvider();
    const signer = provider.getSigner();

    // Approve token
    const tokenContract = new ethers.Contract(PATH_TOKEN_ADDRESS, TOKEN_ABI, signer);
    const requiredAllowance = ethers.utils.parseUnits(price, 18);
    const currentAllowance = await tokenContract.allowance(account, NFT_CONTRACT_ADDRESS);

    if (currentAllowance.lt(requiredAllowance)) {
      const tx = await tokenContract.approve(NFT_CONTRACT_ADDRESS, ethers.constants.MaxUint256);
      await tx.wait();
    }

    // Execute buy
    const nftContract = new ethers.Contract(NFT_CONTRACT_ADDRESS, NFT_ABI, signer);
    const tx = await nftContract.buyNFT(tokenId, { gasLimit: 500000 });
    await tx.wait();

    await refreshData();
  } catch (error) {
    console.error("Purchase failed:", error);
    alert("Transaction failed! Check console for details.");
  } finally {
    setProcessing(false);
  }
};
```

This NextJS function enables users to purchase an NFT from the marketplace using Ethers.js.

How it Works:

1. Check if Wallet is Connected
   - If the user hasn't connected their wallet, an alert prompts them.
2. Set Processing State
   - setProcessing(true); indicates the transaction is in progress.
3. Get Provider & Signer
   - Retrieves a blockchain provider and signer (wallet) to authorize transactions.
4. Approve Token Spending
   - Creates a contract instance for the ERC-20 token used for payments.
   - Converts the NFT price into the correct decimal format using ethers.utils.parseUnits(price, 18).
   - Checks the user's current allowance for spending tokens.
   - If the allowance is insufficient, the function approves the NFT marketplace contract to spend tokens on behalf of the user.
5. Execute NFT Purchase
   - Creates a contract instance for the NFT marketplace.
   - Calls buyNFT(tokenId) on the contract, specifying a gas limit.
   - Waits for transaction confirmation.
6. Refresh Data & Handle Errors

- Calls refreshData() after purchase completion.
- Catches errors and alerts the user if the purchase fails.
- Finally, resets setProcessing(false);.

```
// Handle update whitelist
const handleUpdateWhitelist = async (address: string, status: boolean) => {
  if (!account || !isOwner) return;

  try {
    setProcessing(true);
    const provider = getProvider();
    const signer = provider.getSigner();
    const contract = new ethers.Contract(NFT_CONTRACT_ADDRESS, NFT_ABI, signer);

    const tx = await contract.updateWhitelist(address, status);
    await tx.wait();
    alert('Whitelist updated successfully!');
  } catch (error) {
    console.error("Whitelist update failed:", error);
    alert(error instanceof Error ? error.message : "Update failed");
  } finally {
    setProcessing(false);
  }
};
```

This NextJS function allows the contract owner to update the whitelist status of a given address using Ethers.js.

How it Works:

1. Check If the Wallet is Connected & If the User is the Owner
   - If the user hasn't connected their wallet (!account) or isn't the contract owner (!isOwner), the function exits immediately.
2. Set Processing State
   - setProcessing(true); ensures UI feedback during execution.
3. Get Provider & Signer
   - Retrieves the blockchain provider (getProvider()) and the user's wallet (getSigner()).
4. Interact with the Smart Contract
   - Initializes a contract instance using:
     - NFT_CONTRACT_ADDRESS
     - NFT_ABI
     - signer
   - Calls the updateWhitelist(address, status) function from the smart contract.
5. Wait for Transaction Confirmation
   - Waits for the blockchain to process the transaction (tx.wait()).
6. Success or Error Handling

- Displays an alert message confirming the update.
- Catches and logs errors, showing an alert if the update fails.

7. Reset Processing State
   - setProcessing(false); ensures UI updates after execution.

```
// Handle mint NFT
const handleMintNFT = async (recipient: string, tokenURI: string) => {
  if (!account || !isOwner) return;

  setProcessing(true);

  try {
    const provider = getProvider();
    const signer = provider.getSigner();
    const contract = new ethers.Contract(NFT_CONTRACT_ADDRESS, NFT_ABI, signer);

    const tx = await contract.mintNFT(recipient, tokenURI);
    await tx.wait();
    await refreshData();
  } catch (error) {
    console.error("Minting failed:", error);
    alert(error instanceof Error ? error.message : "Mint failed");
  } finally {
    setProcessing(false);
  }
};
```

This NextJS function allows the contract owner to mint an NFT and send it to a specific recipient wallet using Ethers.js.

How It Works:

1. Check If the Wallet is Connected & Owner
   - If the user hasn't connected their wallet (!account) or isn't the owner (!isOwner), the function exits immediately.
2. Set Processing State
   - setProcessing(true); ensures UI feedback during execution.
3. Get Provider & Signer
   - Retrieves the blockchain provider (getProvider()) and the signer (wallet of the connected user).
4. Initialize Contract Instance
   Connects to the NFT contract using:
   - NFT_CONTRACT_ADDRESS (contract address)
   - NFT_ABI (contract ABI)
   - signer (connected user)
5. Call mintNFT Function in the Contract
   - Mints an NFT to the recipient address with the provided tokenURI (metadata link).
6. Wait for Transaction Confirmation

- Waits for the blockchain to confirm the transaction (tx.wait()).

7. Refresh Data
   - Calls refreshData(); to update UI after minting.
8. Error Handling
   - Logs errors (console.error()) and shows an alert if minting fails.
9. Reset Processing State
   - setProcessing(false); ensures UI updates after execution.

```
const fetchNFTs = useCallback(async () => {
  if (!account) return;

  try {
    const provider = getProvider();
    const contract = new ethers.Contract(NFT_CONTRACT_ADDRESS, NFT_ABI, provider);

    const listedIds = await contract.getAllListings().catch(() => []);

    // Market NFTs
    const marketNFTs = await Promise.all(
      listedIds.map(async (id: ethers.BigNumber) => {
        try {
          const [uri, listing, owner] = await Promise.all([
            contract.tokenURI(id),
            contract.listings(id),
            contract.ownerOf(id).catch(() => '0x0')
          ]);
          const metadata = await fetch(uri.toString()).then(res => res.json());
          return {
            id: id.toString(),
            ...metadata,
            price: ethers.utils.formatUnits(listing.price, 18),
            seller: listing.seller,
            owner: owner,
            isListed: listing.isListed
          };
        } catch (error) {
          console.error(`Error processing NFT ${id}:`, error);
          return null;
        }
      })
    );

    // Owned NFTs
    const totalSupply = await contract.totalSupply().catch(() => ethers.BigNumber.from(0));
    const allIds = Array.from({ length: totalSupply.toNumber() }, (_, i) => i);
```

This function retrieves all listed NFTs and owned NFTs from a deployed NFT smart contract using Ethers.js.

How It Works:

1. Check if Wallet is Connected
   - If account is not connected, it exits immediately (return).
2. Get Provider & Contract Instance
   - getProvider() fetches the blockchain provider.
   - Creates a new contract instance using:

- NFT_CONTRACT_ADDRESS (contract address)
- NFT_ABI (contract ABI)
- provider (wallet connection)

3. Fetch Listed NFTs
   - Calls getAllListings() from the NFT contract to retrieve all listed NFTs.
   - Uses .catch(() => []) to handle errors gracefully.
4. Fetch Metadata for Each Listed NFT
   - Loops through listedIds (IDs of listed NFTs).
   - Fetches on-chain NFT data (tokenURI, listings, ownerOf).
   - Fetches off-chain metadata (stored on IPFS or external storage).
   - Returns NFT details:
     - id: Token ID
     - metadata: NFT metadata (image, name, description, etc.)
     - price: Price (formatted to readable units)
     - seller: Address of seller
     - owner: Current NFT owner
     - isListed: Boolean flag for sale status
   - If fetching fails, logs an error and returns null.
5. Fetch Owned NFTs
   - Calls totalSupply() to get the total number of minted NFTs.
   - Generates an array of all token IDs.

```
// Contract address
export const CONTRACT_ADDRESS = "0xc3e9Cf26237c9002c0C04305D637AEa3d9A4A1DE"; // PATH Token V3 address
export const CAKE_TOKEN_ADDRESS = "0xFa60D973F7642B748046464e165A65B7323b0DEE"; // CAKE token on BSC Testnet
export const PANCAKE_ROUTER = "0xD99D1c33F9fC3444f8101754aBC46c52416550D1"; // PancakeSwap Router on BSC Testnet

// ABI for PATH Token V3 - Updated with all functions from the new contract
export const CONTRACT_ABI = [
  // Read functions
  "function balanceOf(address account) view returns (uint256)",
  "function symbol() view returns (string)",
  "function name() view returns (string)",
  "function decimals() view returns (uint8)",
  "function FAUCET_AMOUNT() view returns (uint256)",
  "function DAILY_LIMIT() view returns (uint256)",
  "function COOLDOWN() view returns (uint256)",
  "function swapFee() view returns (uint256)",
  "function feeWallet() view returns (address)",
  "function pairedToken() view returns (address)",
  "function owner() view returns (address)",
  "function faucetBalance() view returns (uint256)",
  "function antiBotEnabled() view returns (bool)",
  "function blacklist(address account) view returns (bool)",
  "function faucetRecords(address account) view returns (uint256 lastRequest, uint256 dailyCount, uint256 lastBlock)",
  "function allowance(address owner, address spender) view returns (uint256)",

  // Write functions
  "function requestTokens() external",
  "function swapTokens(uint256 amountIn, uint256 amountOutMin, uint24 fee) external",
  "function addLiquidity(uint256 amount0Desired, uint256 amount1Desired, uint24 fee, int24 tickLower, int24 tickUpper) external returns (uint256 tokenId)",
  "function approve(address spender, uint256 amount) external returns (bool)",
  "function refillFaucet(uint256 amount) external",
  "function setFeeParameters(uint256 _newFee, address _newWallet) external",
  "function manageBlacklist(address[] calldata addresses, bool status) external",
  "function toggleAntiBot() external",
  "function transfer(address to, uint256 amount) external returns (bool)",

  // Events
  "event Transfer(address indexed from, address indexed to, uint256 value)",
  "event Approval(address indexed owner, address indexed spender, uint256 value)",
  "event FaucetUsed(address indexed user, uint256 amount)",
  "event LiquidityAdded(address indexed user, uint256 tokenId)",
```

1 Contract Addresses

- PATH Token V3: 0xc3e9cF26237c9002cC04305D637AEa3d94AA1DE
- CAKE Token (BSC Testnet): 0xFa6D0973F7642B74804646e16A56B7323bD8DEE

- PancakeSwap Router (BSC Testnet): 0xD99D1c33F9fC3444f8101754aBC46c52416550D1

2. Key ABI Functions

Read Functions (No Gas)

- Check balance, symbol, name, decimals.
- Get swap fee, owner, paired token, faucet balance.
- Check if anti-bot protection or blacklist is enabled.

Write Functions (Requires Gas)

- Token Swap & Liquidity: swapTokens(), addLiquidity()
- Approve Spending: approve()
- Faucet & Security: requestTokens(), refillFaucet(), toggleAntiBot(), blacklist()
- Transfer Tokens: transfer()

Events

- Transfer & Approval logs.
- Faucet Claiming & Liquidity Added notifications.

```
  const initContract = async () => {
    if (window.ethereum) {
      await checkNetwork();
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = provider.getSigner();
      const contract = new ethers.Contract(CONTRACT_ADDRESS, CONTRACT_ABI, signer);

      // Lấy số dư tBNB
      const balance = await provider.getBalance(await signer.getAddress());
      setTbnbBalance(ethers.utils.formatEther(balance));

      // Các thông tin khác
      const tokenBal = await contract.balanceOf(await signer.getAddress());
      const faucetBal = await contract.faucetBalance();

      setContract(contract);
      setTokenBalance(ethers.utils.formatEther(tokenBal));
      setFaucetBalance(ethers.utils.formatEther(faucetBal));
    }
  };

  if (window.ethereum) {
    window.ethereum.on('chainChanged', initContract);
    initContract();
  }
}, []);
```

Function: initContract

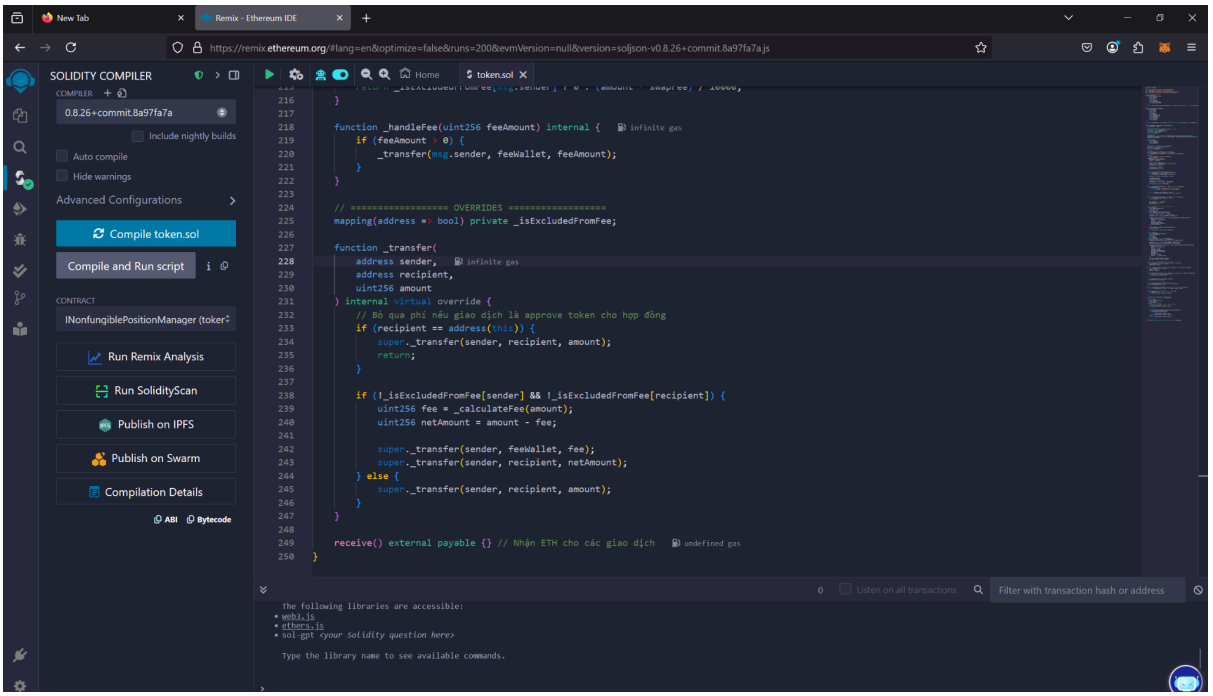
This function initializes the connection to a smart contract and retrieves relevant blockchain data.

Steps in the Function

1. Check for Ethereum Provider
    - Ensures the user has MetaMask or another web3 provider.
2. Connect to Ethereum Network
    - Uses ethers.js to create a Web3Provider instance.
    - Retrieves the signer (user's wallet).

- Connects to the smart contract using CONTRACT_ADDRESS and CONTRACT_ABI.

3. Retrieve Balances
   - BNB Balance: Calls provider.getBalance(userAddress), then formats it.
   - Token Balance: Calls contract.balanceOf(userAddress).
   - Faucet Balance: Calls contract.faucetBalance().
4. Store Data
   - Updates the state variables:
     - setTbnbBalance() stores BNB balance.
     - setTokenBalance() stores token balance.
     - setFaucetBalance() stores faucet balance.
5. Listen for Network Changes
   - Re-initializes the contract when the user switches networks.



Setup Instructions

- **Clone the repository**: git clone https://github.com/TTMordred/CryptoPath.git
- **Navigate to project directory**: cd cryptopath
- **Install dependencies**: npm install next --legacy-peer-deps
- **Set up environment variables**: touch .env.local
- **Populate .env.local with:**

```
ETHERSCAN_API_KEY=YOUR_API_KEY
ETHERSCAN_API_URL=https://api.etherscan.io/api
SMTP_HOST=smtp.example.com
SMTP_PORT=587
SMTP_SECURE=false
SMTP_USER=your-email@example.com
SMTP_PASSWORD=your-password
NEO4J_URI=neo4j+s://your-database-uri
NEO4J_USERNAME=your-username
NEO4J_PASSWORD=your-password
NEXTAUTH_URL=https://cryptopath.vercel.app/
NEXTAUTH_SECRET=your-secret-key
NEXT_PUBLIC_INFURA_KEY=your-infura-key
NEXT_PUBLIC_WALLETCONNECT_PROJECT_ID=your-walletconnect-projectid
NEXT_PUBLIC_URL=http://localhost:3000
ALCHEMY_API_KEY=your-alchemy-key
REACT_APP_DAPPRADAR_API_KEY=your-DAPPRADAR-key
COINMARKETCAP_API_KEY=your-COINMARKETCAP-key
```

- **Start the development server**: npm run dev

API Design

1. Wallet API

File: app/api/wallet/route.ts

Endpoint: /api/wallet
Method: GET

Query Parameters:

- **address** *(string, required)*: Ethereum wallet address.
- **network** *(string, optional)*: Blockchain network *(default: mainnet)*.
- **provider** *(string, optional)*: Data provider *(default: etherscan)*.

Response:

```
{
  "balance": "0.1234 ETH",
  "transactionCount": 42,
  "network": "mainnet"
}
```

2. NFT Stats API

File: app/api/analytics/nft-stats/route.ts

Endpoint: /api/analytics/nft-stats
Method: GET

Response:

```
{
  "collections": [
    {
      "name": "Bored Ape Yacht Club",
      "symbol": "BAYC",
      "floorPrice": 72.45,
      "volume24h": 456.32,
      "totalVolume": 890745.23,
      "owners": 6213
    },
    ...
  ]
}
```

3. DeFi TVL API

File: app/api/analytics/defi-tvl/route.ts

Endpoint: /api/analytics/defi-tvl
Method: GET

Query Parameters:

- **protocol** *(string, optional)*: Specific DeFi protocol *(default: all)*.

Response:

```
{
  "data": [
    {
      "date": "2023-10-01",
      "tvl": 123456789.12
    },
    ...
  ]
}
```

4. CoinMarketCap Listings API

File: app/api/coinmarketcap/listings/route.ts

Endpoint: /api/coinmarketcap/listings
Method: GET

Query Parameters:

- **limit** *(number, optional)*: Number of cryptocurrencies to fetch *(default: 10)*.
- **sort** *(string, optional)*: Sorting criteria *(default: market_cap)*.
- **sort_dir** *(string, optional)*: Sorting direction *(default: desc)*.

Response:

```
{
  "data": [
    {
      "id": 1,
      "name": "Bitcoin",
      "symbol": "BTC",
      "price": 65000,
      "market_cap": 1300000000000,
      "volume_24h": 28000000000
    },
    ...
  ]
}
```

5. Whale Alerts API

File: app/api/analytics/whale-alerts/route.ts

Endpoint: /api/analytics/whale-alerts
Method: GET

Query Parameters:

- **limit** *(number, optional)*: Number of transactions to fetch *(default: 10)*.

Response:

```
{
  "transactions": [
    {
      "id": "tx_12345",
      "symbol": "ETH",
      "amount": 500,
      "value": 1500000,
      "from": "Binance",
```

```
      "to": "Unknown",
      "type": "transfer",
      "timestamp": 1696200000
    },
    ...
  ],
  "totalValue": 5000000,
  "timestamp": 1696200000
}
```


6. Gas Prices API

File: app/api/analytics/gas-prices/route.ts

Endpoint: /api/analytics/gas-prices
Method: GET

Response:


```
{
  "slow": 10,
  "average": 25,
  "fast": 50,
  "baseFee": 5.5,
  "timestamp": 1696200000
}
```


7. Portfolio API

File: app/api/portfolio/route.ts

Endpoint: /api/portfolio
Method: GET

Query Parameters:

- **address** *(string, required)*: Ethereum wallet address.
- **provider** *(string, optional)*: Data provider *(moralis, alchemy, or combined)*.

Response:


```
{
  "address": "0x123...abc",
  "tokens": [
    {
```

```
      "contractAddress": "0xabc...123",
      "balanceFormatted": "100.0",
      "usdPrice": 1.5,
      "usdValue": 150.0
    },
    ...
  ],
  "totalValue": 1500.0,
  "totalBalance": 10,
  "provider": "moralis"
}
```


8. Trending Coins API

File: app/api/analytics/trending-coins/route.ts

Endpoint: /api/analytics/trending-coins
Method: GET

Response:


```
{
  "coins": [
    {
      "id": "bitcoin",
      "name": "Bitcoin",
      "symbol": "BTC",
      "market_cap_rank": 1,
      "price_btc": 1.0,
      "thumb": "/icons/btc.svg",
      "score": 100
    },
    ...
  ]
}
```


9. Fear & Greed Index API

File: app/api/analytics/fear-greed-index/route.ts

Endpoint: /api/analytics/fear-greed-index
Method: GET

Response:

```
{
  "value": 45,
  "valueText": "Fear",
  "timestamp": 1696200000
}
```


References

Binance (2024) *Assets*, Binance. Available at: https://www.binance.com/en/my/wallet/account/earn

Etherscan (2024) *The Ethereum Blockchain Explorer*, Etherscan. Available at: https://etherscan.io/

Ethereum Smart Contract Best Practices (2024) *Visualization*, Material for MkDocs. Available at: https://consensys.github.io/smart-contract-best-practices/security-tools/visualization/

De.Fi (2024) *Free smart contract audit: Scan any smart contract, token or NFT*, De.Fi. Available at: https://de.fi/scanner

MISTTRACK BY SLOWMIST (2024) *A Crypto Tracking and Compliance Platform for Everyone*, MISTTRACK. Available at: https://misttrack.io

OKX (2017) *(Read-only) Guest mode - Demo*, OKX. Available at: https://www.okx.com/web3

OWASP Top Ten (2024) *Top 10 Web Application Security Risks*, OWASP. Available at: https://owasp.org/www-project-top-ten/

Aceternity UI (n.d.) *Make your websites look 10x better*. Available at: https://ui.aceternity.com

COBE (n.d.) *5kB WebGL Globe*. Available at: https://cobe.vercel.app

Federal Bureau of Investigation (FBI) (2023) *2023 Cryptocurrency Fraud Report*. Available at: https://www.ic3.gov/Media/PDF/AnnualReport/2023_IC3CryptocurrencyReport.pdf

Next.js (n.d.) *The React Framework for the Web*. Available at: https://nextjs.org

React Flow (n.d.) *Wire Your Ideas with React Flow*. Available at: https://reactflow.dev

React Icons (n.d.) *React Icons*. Available at: https://react-icons.github.io/react-icons/

Recharts (n.d.) *A composable charting library built on React components*. Available at: https://recharts.org

shadcn/ui (n.d.) *Build your component library*. Available at: https://ui.shadcn.com

Tailwind CSS (n.d.) *Rapidly build modern websites without ever leaving your HTML*. Available at: https://tailwindcss.com