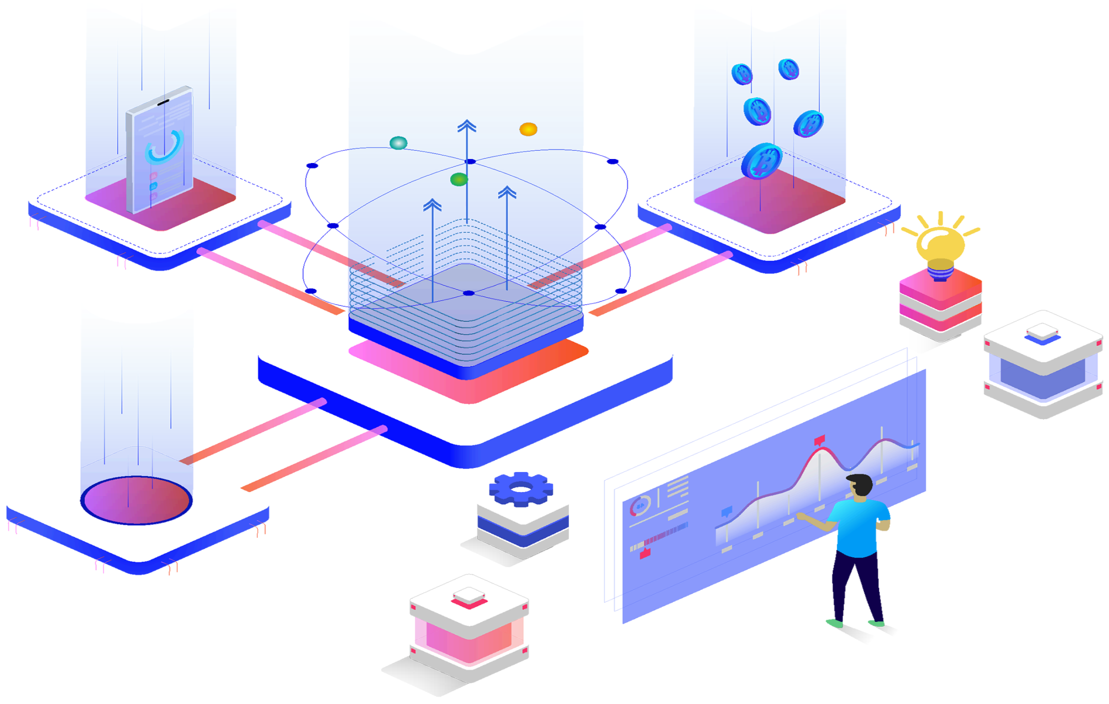
Assignment 2 Report



Unit: COS30049 - Computing Technology Innovation Project
Tutor: Dr.Tran Trung Tin
Group: 3
Group Set 3
Location: Swinburne Da Nang

Table Of Content

I. Project background and introduction

1.1 Overview

The Blockchain Data Visualisation System aims to provide a comprehensive platform for data visualization and analysis within blockchain networks. The solution will provide users with clear and simple visual representations of transactional data, network activity, and overall blockchain performance.

1.2 Objectives

- Provide a solid framework for blockchain data visualization.
- Improve data accessibility and understanding for stakeholders.
- Improve decision-making by utilizing intelligent data visualization.

1.3 Importance

With the growing popularity of blockchain technology, it is critical to have efficient tools for processing and interpreting data. The Blockchain Data Visualisation System meets this demand by providing an intuitive user interface for interpreting sophisticated data relationships and structures.

II. Team introduction

2.1 Nguyen Duc Le Nguyen - 104224493

❖ Database Design and Implementation:
  ➢ Take the lead in designing and implementing the database structure.
  ➢ Create tables and define relationships according to project requirements.
  ➢ Ensure efficient data storage and retrieval mechanisms.

❖ Integration with Front-End:
  ➢ Work on integrating the backend functionalities with the front-end interface.
  ➢ Test and debug interactions to ensure smooth data flow and real-time updates.
  ➢ Implement mechanisms for dynamically generating content based on user actions or data changes.

❖ Testing and Debugging:
  ➢ Take responsibility for thoroughly testing the backend functionalities.
  ➢ Identify and troubleshoot any issues that arise during testing.
  ➢ Ensure data accuracy, proper error handling, and responsiveness of the backend.

2.2. Le Minh Long - 104180139

❖ Server-Side Logic:
  ➢ Implement the backend logic necessary for handling user interactions and data processing.
  ➢ Create APIs for data submission and retrieval as per project requirements.

- ➢ Ensure proper handling of user requests and responses from the server.

- ❖ User-Friendly Design:
  - ➢ Implement mechanisms for providing well-crafted messages to users in various scenarios.
  - ➢ Craft messages for scenarios such as upload success, unsupported file types, item not found, server connection errors, etc.
  - ➢ Ensure that user interactions with the application are smooth and intuitive.
- ❖ Testing and Debugging:
  - ➢ Assist in testing the backend functionalities and identifying any issues.
  - ➢ Collaborate with Student 1 on troubleshooting and resolving backend-related bugs.
  - ➢ Ensure thorough testing to maintain data accuracy and system responsiveness.

2.3 Tran Minh Quan - 104167682

- ❖ Database Design and Implementation:
  - ➢ Assist Student 1 in designing and implementing the database structure.
  - ➢ Contribute to creating tables and defining relationships in the database.
  - ➢ Ensure data integrity and efficiency in data storage and retrieval.
- ❖ Server-Side Logic:
  - ➢ Collaborate with Minh Long in implementing backend logic for user interactions and data processing.
  - ➢ Work on creating APIs for data submission and retrieval in collaboration with Minh Long.
  - ➢ Ensure seamless communication between the frontend and backend components.
- ❖ Testing and Debugging:
  - ➢ Collaborate with Nguyen and Long in testing the backend functionalities.
  - ➢ Assist in identifying and troubleshooting issues during testing.
  - ➢ Ensure that all backend functionalities meet quality standards and perform as expected.

III. Project requirement list and description

3.1 User Interface (UI)

A user-centric web interface will be developed to ensure a seamless and enjoyable interaction with the system. Key features include:

3.1.1 Search Bar for Information Retrieval

Users will be able to quickly and easily retrieve relevant information by typing wallet addresses into an obvious search field. Enhancing the user experience overall is the aim of the search feature's user-friendly design.

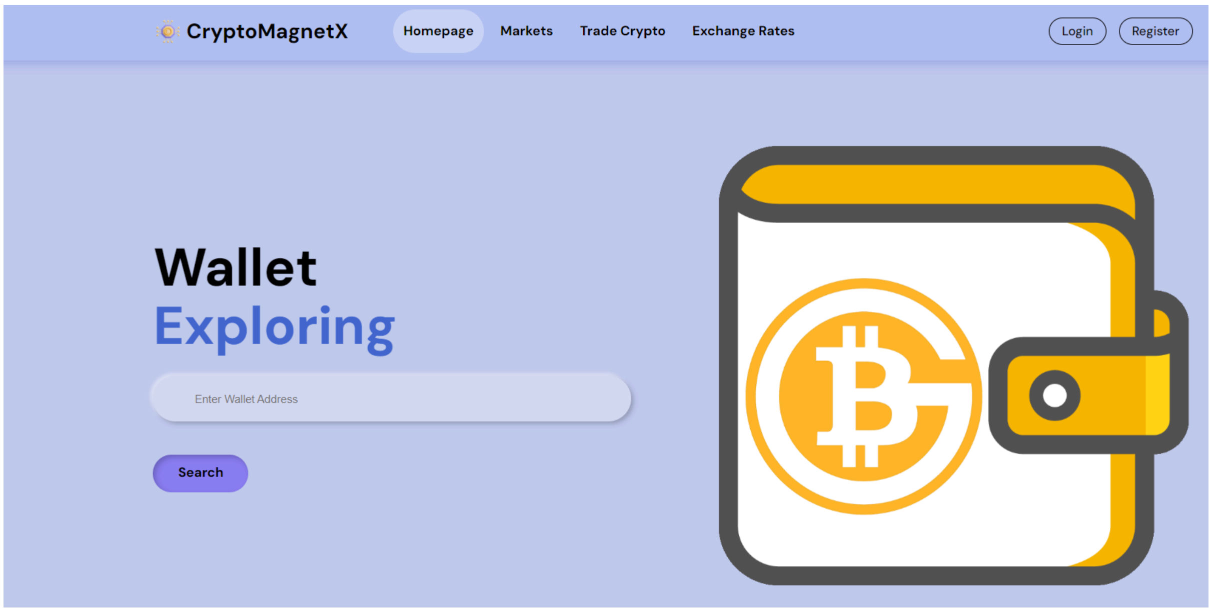
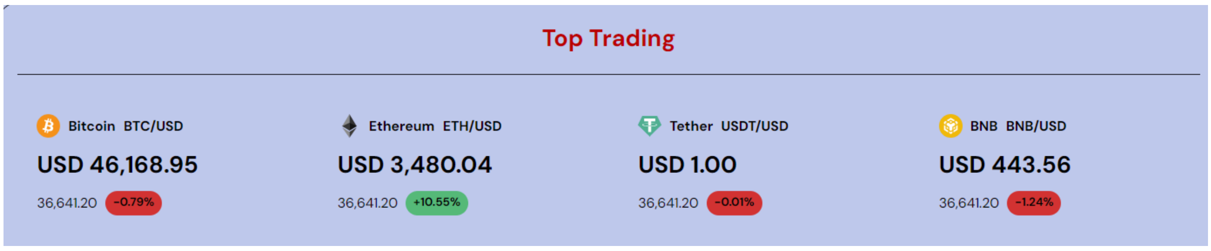


Figure 1. User Interface

3.1.2. Graph for Transaction Exploration

The implementation of an interactive graph will furnish users with a graphical depiction of transactions among wallet addresses. This dynamic feature makes it possible to investigate and examine transactional linkages, which adds to a thorough comprehension of the data.



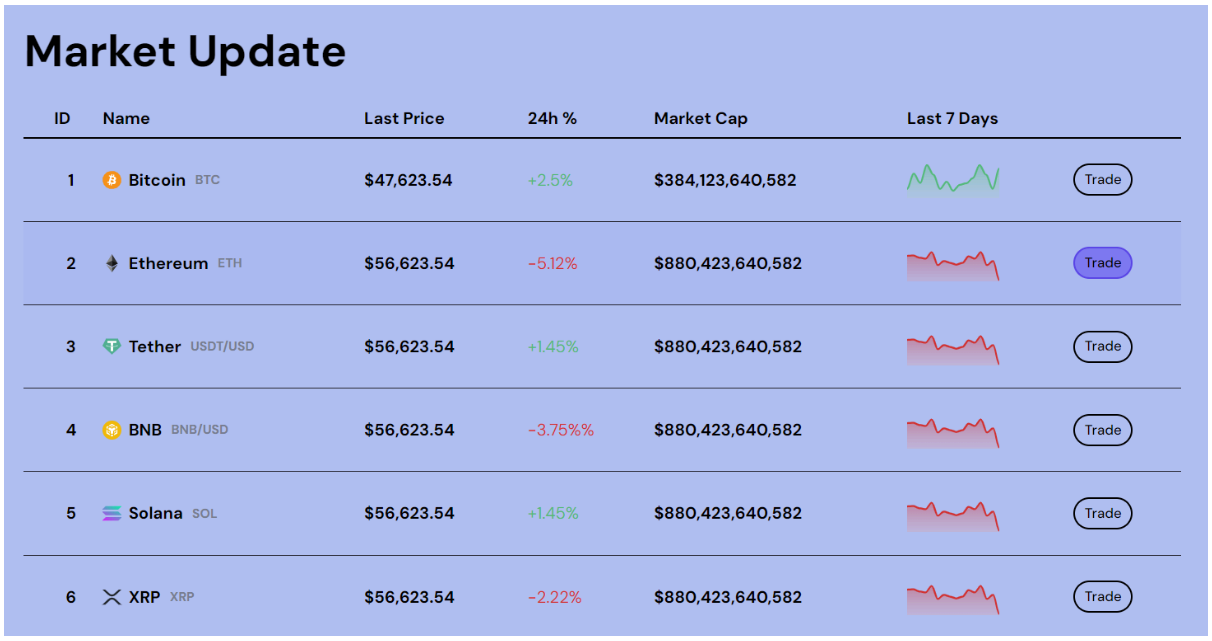
Top Trading

Bitcoin BTC/USD	Ethereum ETH/USD	Tether USDT/USD	BNB BNB/USD
USD 46,168.95	USD 3,480.04	USD 1.00	USD 443.56
36,641.20 -0.79%	36,641.20 +10.55%	36,641.20 -0.01%	36,641.20 -1.24%

Figure 2. Graph for Transaction Exploration

3.1.3 Table for Coin Status Updates

To show real-time changes in currency status and related data, a dynamic table will be included. This tabular structure makes sure users can easily access and understand the most recent information, resulting in a productive and educational experience.



Market Update

ID	Name	Last Price	24h %	Market Cap	Last 7 Days	
1	Bitcoin BTC	$47,623.54	+2.5%	$384,123,640,582		Trade
2	Ethereum ETH	$56,623.54	-5.12%	$880,423,640,582		Trade
3	Tether USDT/USD	$56,623.54	+1.45%	$880,423,640,582		Trade
4	BNB BNB/USD	$56,623.54	-3.75%%	$880,423,640,582		Trade
5	Solana SOL	$56,623.54	+1.45%	$880,423,640,582		Trade
6	XRP XRP	$56,623.54	-2.22%	$880,423,640,582		Trade

Figure 3. Table for Coin Status Updates

3.1.4 Login/Register Page for Credentials

To securely handle user credentials, a special login/register page will be created. By giving users access to extra features and enabling them to keep a customized profile within the system, this area guarantees a personalized experience for users.

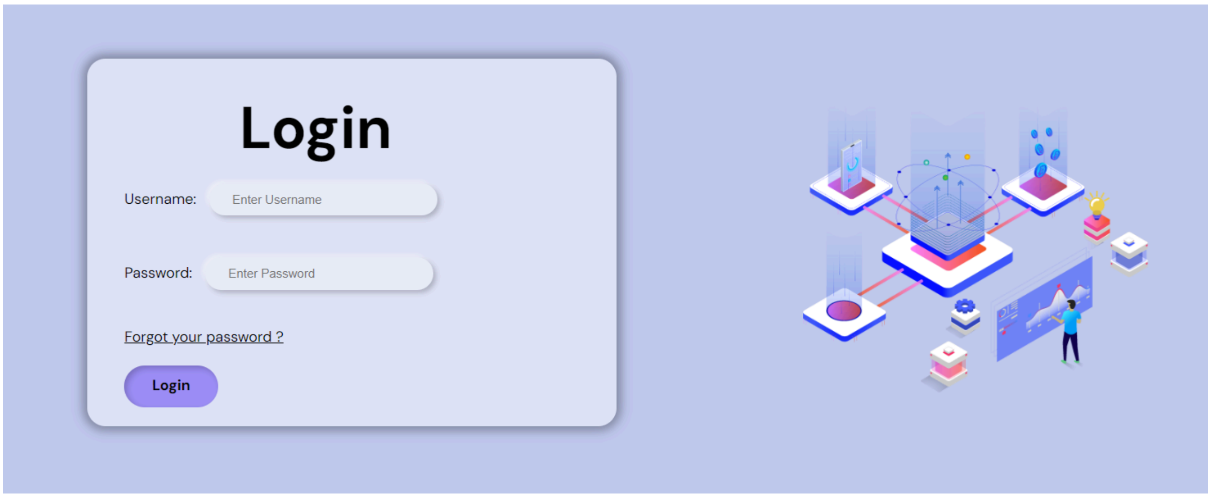


Figure 4 . Login/Register Page for Credentials

3.1.5. Intuitive Navigation

The user interface will have a simple navigation system that lets users move between parts with ease. The overall usability of the web interface will be improved by labels that are clear and succinct and by icons that are instantly recognisable.

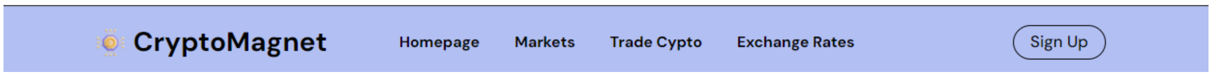
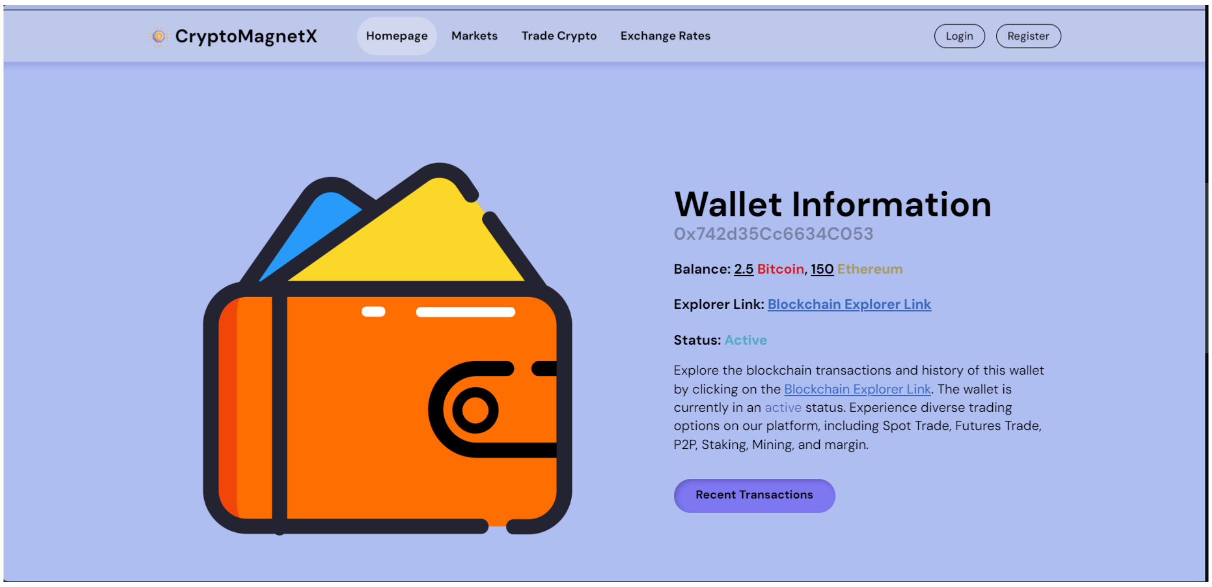


Figure 5 . Intuitive Navigation

3.2 Address Information Display

Entering a wallet address causes the system to quickly retrieve and show important data, such as the amount of the wallet and other relevant information related to the address. This makes sure users can quickly and easily access important information.



Wallet Information

0x742d35Cc6634C053

Balance: 2.5 Bitcoin, 150 Ethereum

Explorer Link: Blockchain Explorer Link

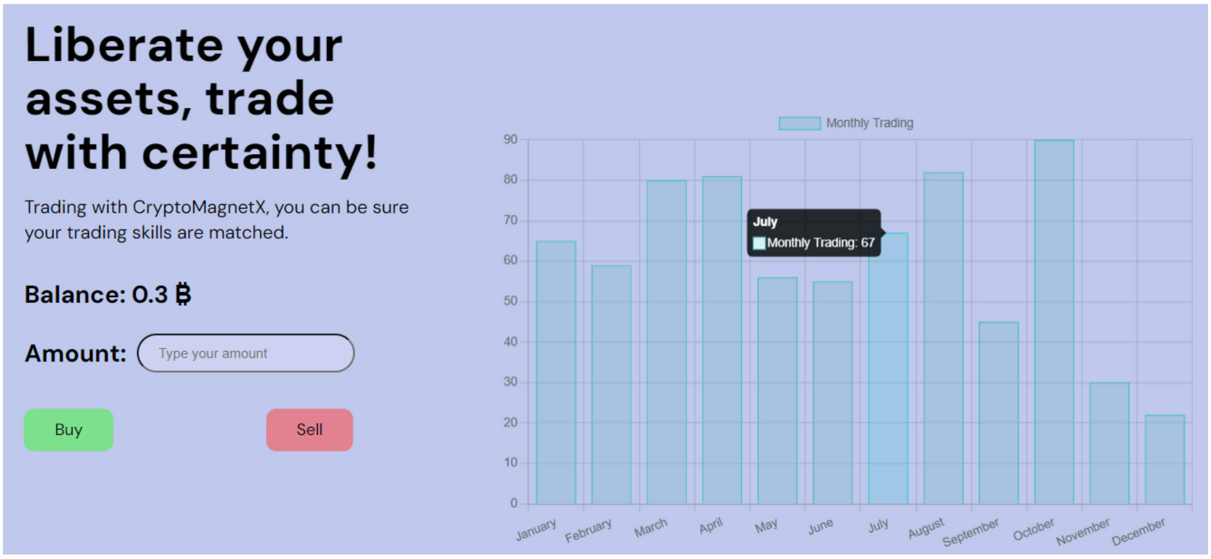
Status: Active

Explore the blockchain transactions and history of this wallet by clicking on the Blockchain Explorer Link. The wallet is currently in an active status. Experience diverse trading options on our platform, including Spot Trade, Futures Trade, P2P, Staking, Mining, and margin.

Figure 6. Address Information Display

3.3 Transaction Information Chain

In addition to the graphical representation, detailed transaction information will be presented in tabular format on the website.



Liberate your assets, trade with certainty!

Trading with CryptoMagnetX, you can be sure your trading skills are matched.

Balance: 0.3 ฿

Amount:

Figure 7. Trading Graph

The table will include relevant details such as transaction timestamps, amounts, and any additional information necessary for a comprehensive overview. The edges connecting these nodes symbolize the transactional chain, visually portraying the flow of funds and interactions between connected addresses.

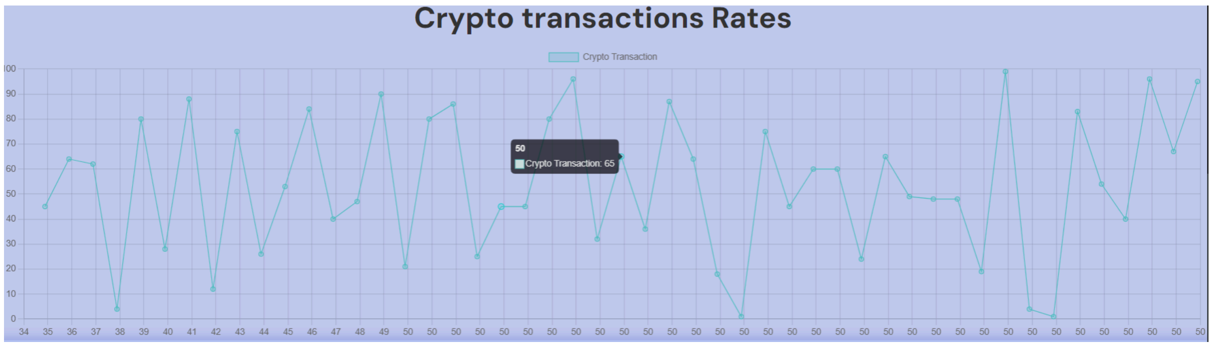
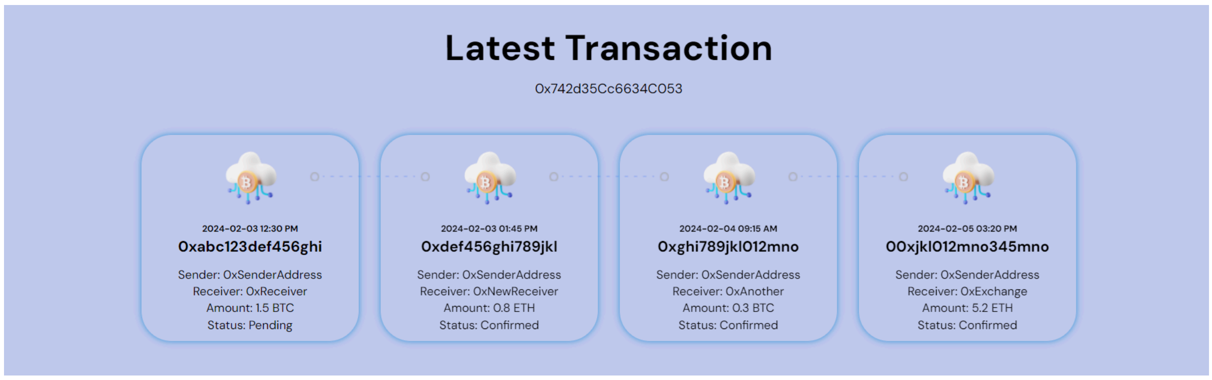


Figure 8. Transaction Information Graph

Users will have the capability to explore transaction paths intuitively. Interaction with nodes, such as clicking, will allow them to follow the chain, navigate through connected addresses, and gain insights into the transactional history.


Latest Transaction

0x742d35Cc6634C053

2024-02-03 12:30 PM
Oxabc123def456ghi
Sender: OxSenderAddress
Receiver: OxReceiver
Amount: 1.5 BTC
Status: Pending

2024-02-03 01:45 PM
Oxdef456ghi789jkl
Sender: OxSenderAddress
Receiver: OxNewReceiver
Amount: 0.8 ETH
Status: Confirmed

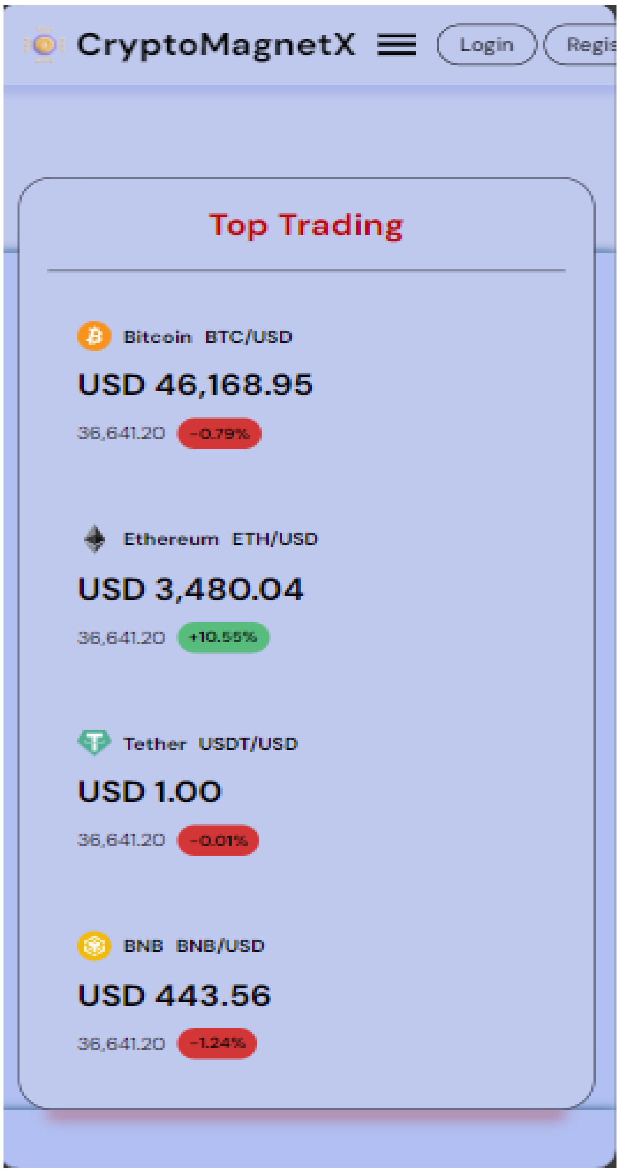
2024-02-04 09:15 AM
Oxghi789jkl012mno
Sender: OxSenderAddress
Receiver: OxAnother
Amount: 0.3 BTC
Status: Confirmed

2024-02-05 03:20 PM
OOxjkl012mno345mno
Sender: OxSenderAddress
Receiver: OxExchange
Amount: 5.2 ETH
Status: Confirmed

Figure 9. Transaction Information Chain

3.4 Responsive Design

Prioritizing a responsive design, our system ensures seamless adaptation to diverse screen sizes and devices. With a focus on mobile responsiveness, users can effortlessly access the system, guaranteeing a consistent and user-friendly experience across various platforms.


CryptoMagnetX Login Regis

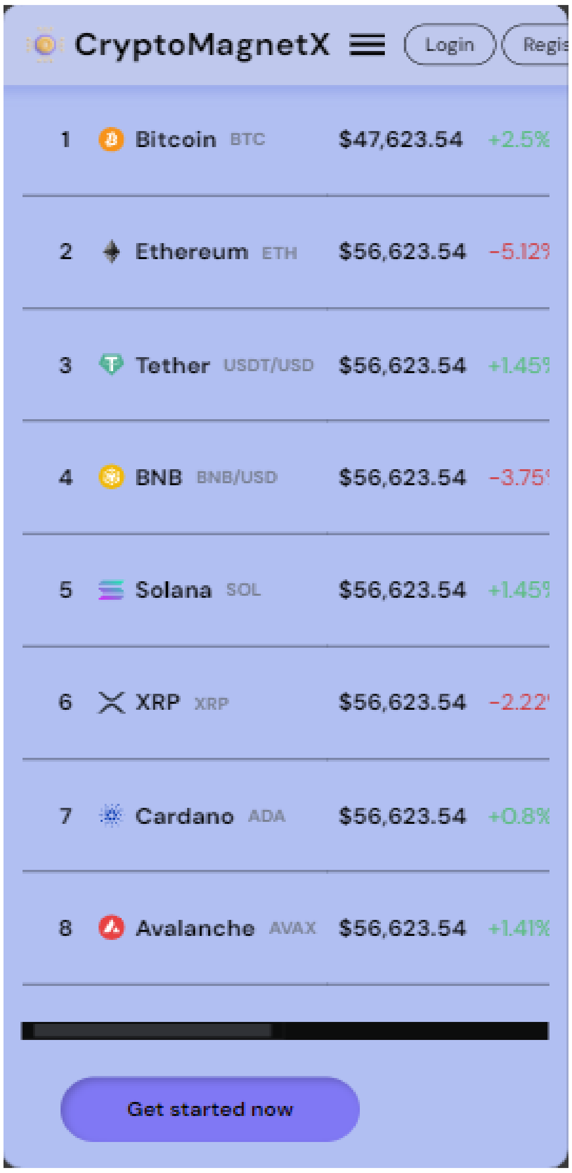
Top Trading

Bitcoin BTC/USD
USD 46,168.95
36,641.20 -0.79%

Ethereum ETH/USD
USD 3,480.04
36,641.20 +10.55%

Tether USDT/USD
USD 1.00
36,641.20 -0.01%

BNB BNB/USD
USD 443.56
36,641.20 -1.24%


CryptoMagnetX Login Regis

1	Bitcoin BTC	$47,623.54	+2.5%
2	Ethereum ETH	$56,623.54	-5.12
3	Tether USDT/USD	$56,623.54	+1.45
4	BNB BNB/USD	$56,623.54	-3.75
5	Solana SOL	$56,623.54	+1.45
6	XRP XRP	$56,623.54	-2.22
7	Cardano ADA	$56,623.54	+0.8%
8	Avalanche AVAX	$56,623.54	+1.41%

Get started now

Figure 10&11. Responsive website

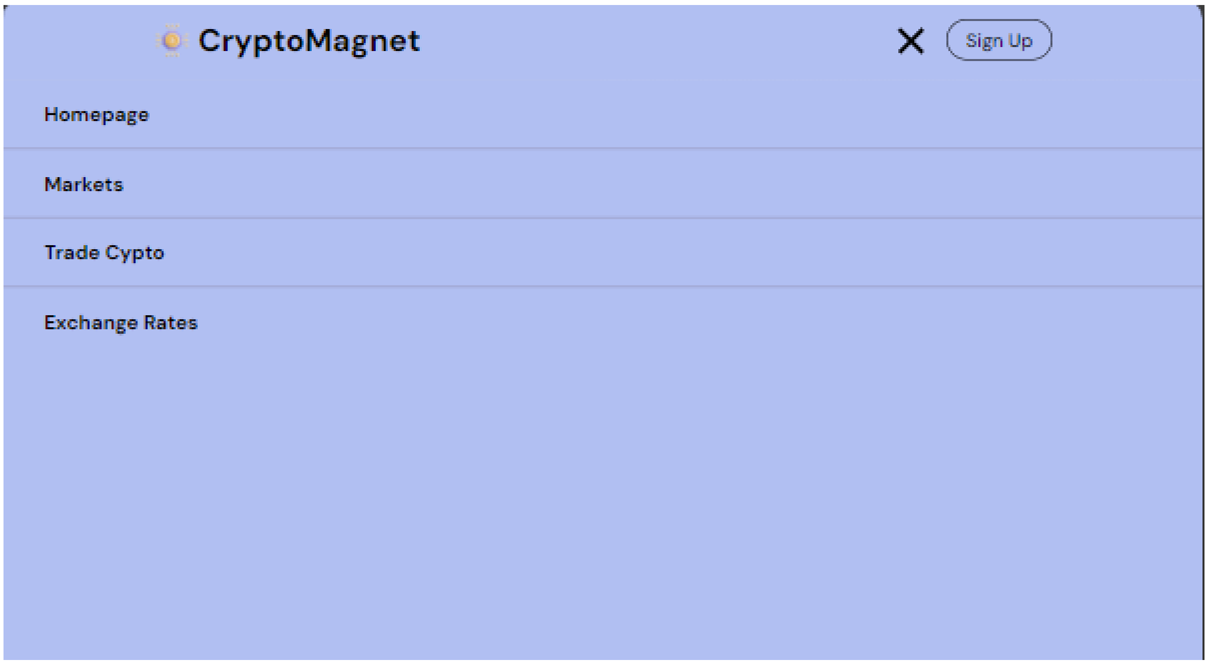


Figure 12. Responsive website

IV. Project design

4.1 Overall System Architecture Design

4.1.1 System Requirements

We implement wallet searching to retrieve data and to allow searching the address chains of transactions. Furthermore, display trading data in a tabular format that is simple to use to Aimprove accessibility and guarantee an accurate depiction of pertinent financial facts for a more seamless and effective user experience.

4.1.2 Data Model

We created a relational database with Neo4J to ensure effective administration and archiving of user and transaction data. This method makes it possible to create a scalable and organized data model, making retrieving and manipulating data easier. The system's usage of Neo4J by the system contributes to improved data integrity and strong relationship support, which in turn leads to optimal efficiency and dependability while handling complex transactions and user-related data.

4.1.3 Choose Technologies

Neo4J was chosen as the database for smooth data management. For the front end, we employed HTML, CSS, and JavaScript to create a responsive and interactive user interface, ensuring a dynamic and engaging user experience with smooth data flow and presentation.

4.1.4 System Architecture Diagram

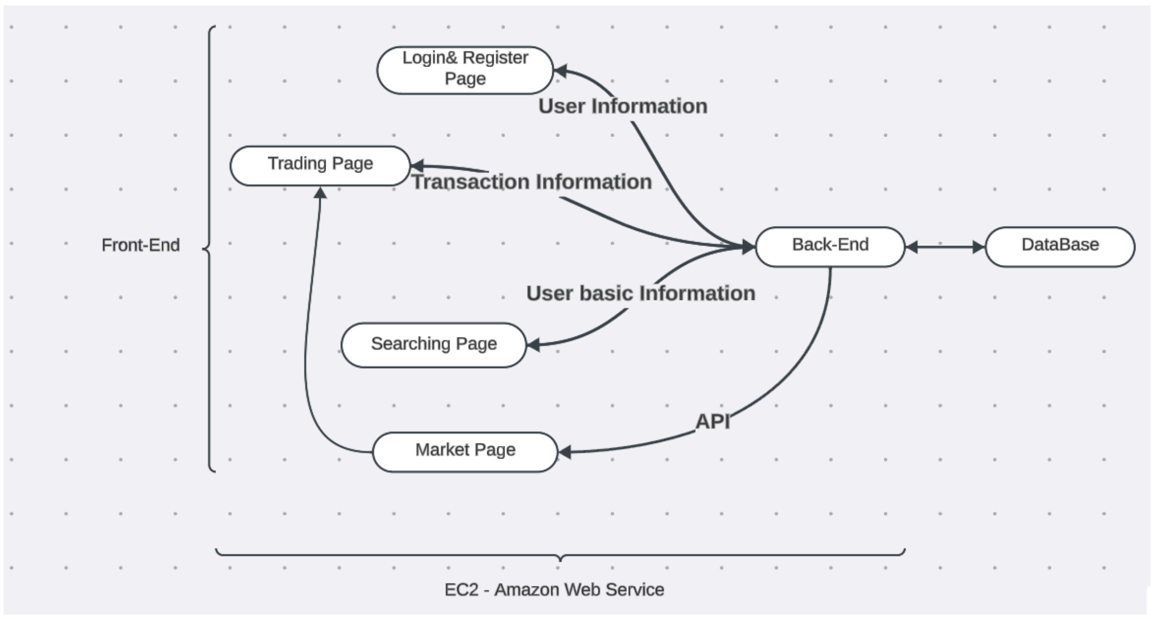


Figure 19. System Architecture Diagram

4.1.5 Deployment Strategy

We will host virtual servers using AWS EC2's capabilities, which ensures scalability and dependable performance. A smooth link between the database and the backend can be established by integrating Neo4J with AWS via APIs. This thoughtful AWS setup guarantees a strong architecture that enables effective data sharing, peak system performance, and scalability to meet evolving requirements without sacrificing data accessibility and integrity.

4.2 Front-end Prototype

We've created a blueprint for the website by using Micro.com and carefully allocating space for each feature. This ensures a comprehensive and logical layout with areas for navigation, search, login, and registration.

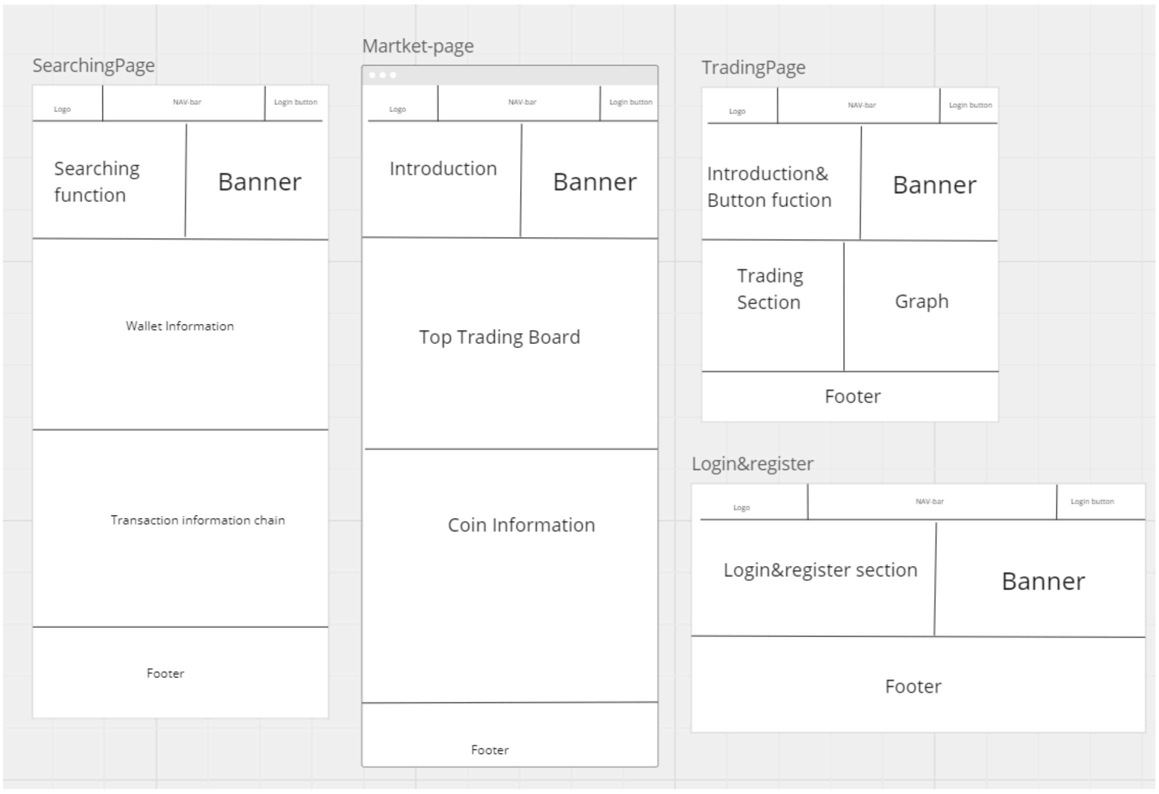


Figure 13. Wireframes and Sketches

Using Figma.com, we created high-fidelity HTML and CSS prototypes to guarantee a visually appealing and simple-to-use user experience. This step is to improve the user experience by fine-tuning the website's interactive and visual components.

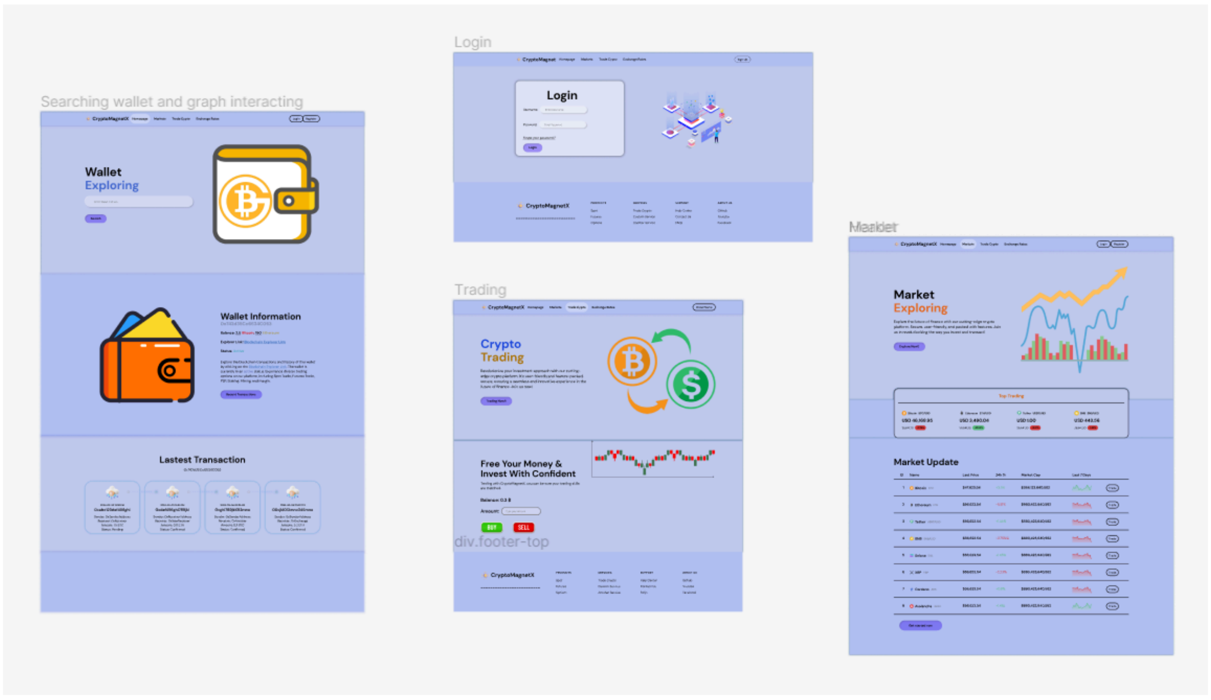


Figure 14. High-fidelity Prototypes

4.3 User Flow Diagrams

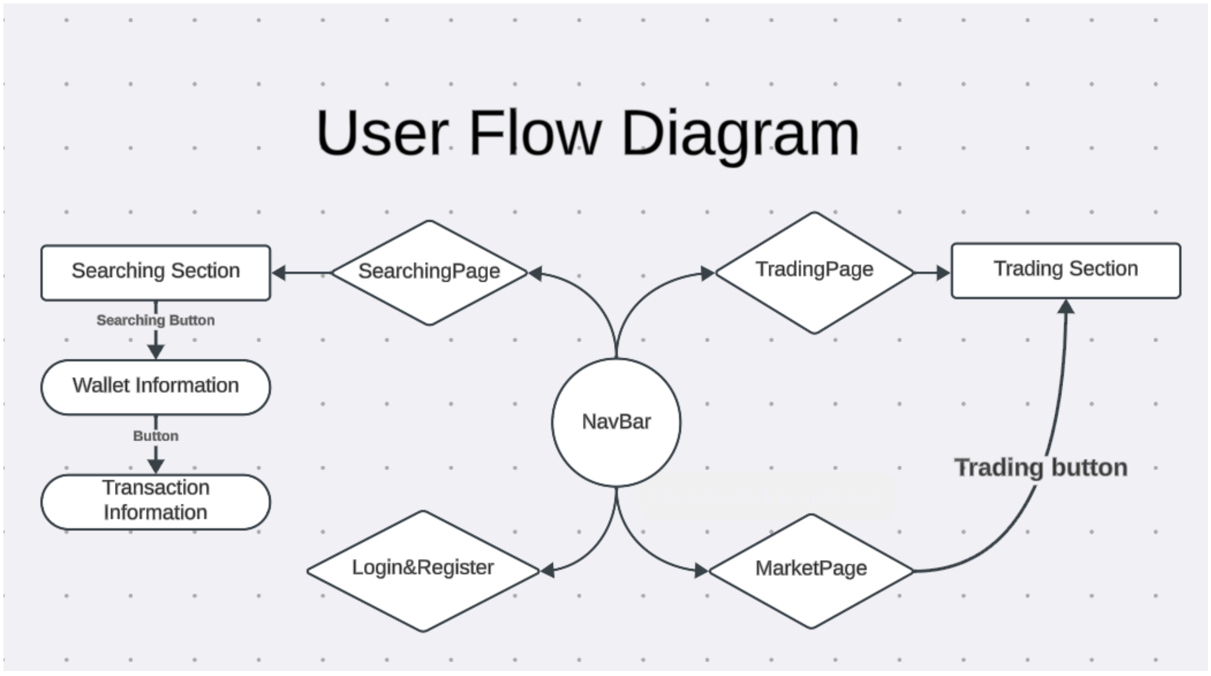


Figure 15. User Flow Diagram

The user flow diagrams depict a logical progression through the website:

- Users navigate through different pages using the intuitive **Navigation bar**.
- Accessing the **Searching page**, users can explore wallet information via the searching button and delve into transaction graphs using a dedicated function button.

- Moving to the **Market page**, users gain insights into coin status and engage in trading activities by clicking the trade button.
- Users proceed to the **Trading page** for coin transactions, facilitated by clearly labeled buy and sell buttons, ensuring a smooth and understandable trading experience.

4.4 Backend Database Design

Our project utilizes Neo4j, a graph database, to effectively visualize relationships between nodes. Neo4j's graph-based approach is ideal for representing complex relationships in our dataset, allowing for efficient querying and visualization of data.

4.4.1 Nodes Import:

To populate the database with node data, we import information from a CSV file named 'nodes.csv'. This CSV contains information about different addresses, including their unique address identifier (addressId), type (type), Bitcoin balance (btc), and Ethereum balance (eth).

4.4.2 Relationships Import:

The database also imports relationship data from another CSV file named 'relationships.csv'. This CSV contains transactional data between different addresses, including transaction hash (hash), value transferred (value), input details (input), transaction index (transaction_index), gas information (gas, gas_used, gas_price), transaction fee (transaction_fee), block number (block_number), block hash (block_hash), and block timestamp (block_timestamp).

These queries create the nodes and relationships within the Neo4j graph database, enabling efficient representation and querying of data for our project's backend functionality.

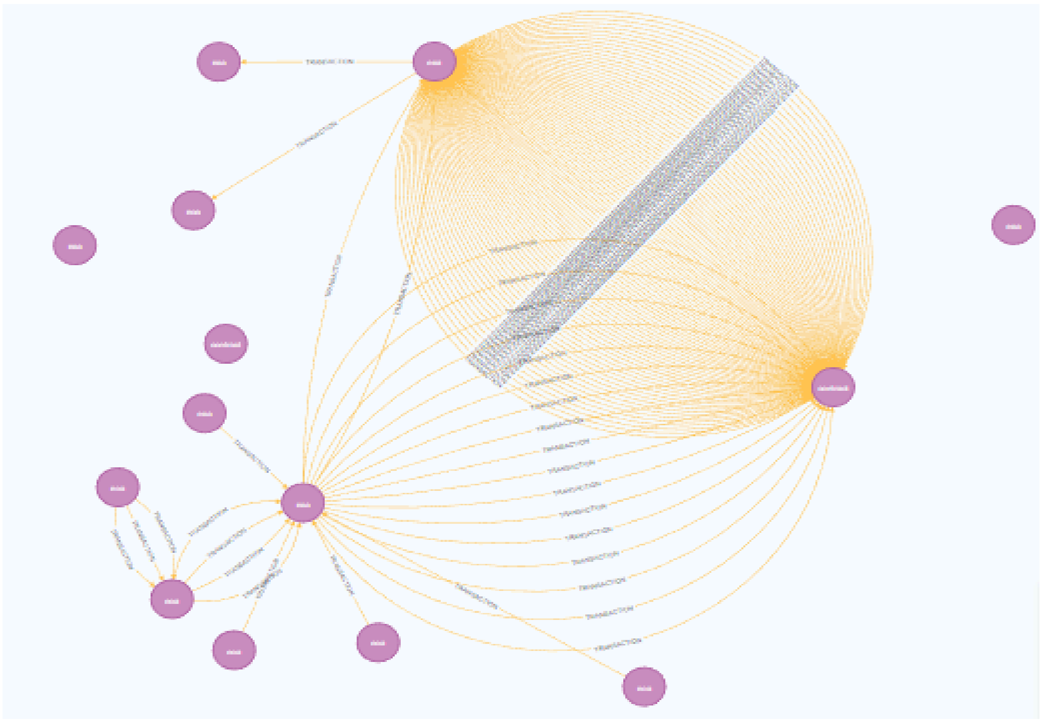


Figure 16. Neo4J graph database

4.5 API Design

4.5.1 Cryptocurrency Wallet Management API:

This API plays a role in managing cryptocurrency wallets and transactions within a Neo4j database. It allows us for operations such as retrieving wallet details, managing transactions, user authentication (login and signup), and calculating and visualizing average transaction values.

4.5.1.1 Endpoints

Wallet Information Retrieval

- URL: `/api/wallet/:address`
- Method: `GET`
- Description: Retrieves detailed information about a specific wallet by its address.

Wallet Transactions Retrieval

1. URL: `/api/wallet/:address/transactions`
2. Method: `GET`
3. Description: Fetches transactions related to a specific wallet address.

User Login

- URL: `/login`
- Method: `POST`
- Description: Authenticates a user based on username and password.

User Signup

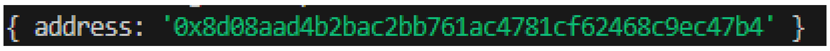
- URL: `/api/signup`
- Method: `POST`
- Description: Registers a new user and wallet in the system.

Average Transaction Value Calculation

- URL: `/api/transactions/average/:address`
- Method: `GET`
- Description: Calculates the average value of transactions where the specified address has sold or bought.

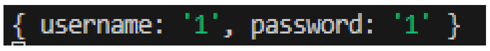
4.5.1.2 Request Parameters

Wallet Address: Required for endpoints that operate on specific wallets or transactions. It is specified in the URL path as :address.



```
{ address: '0x8d08aad4b2bac2bb761ac4781cf62468c9ec47b4' }
```

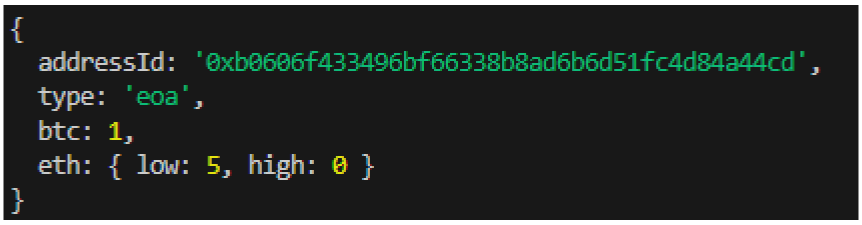
User Credentials: Required for the login and signup endpoints. These are provided in the request body as username and password.



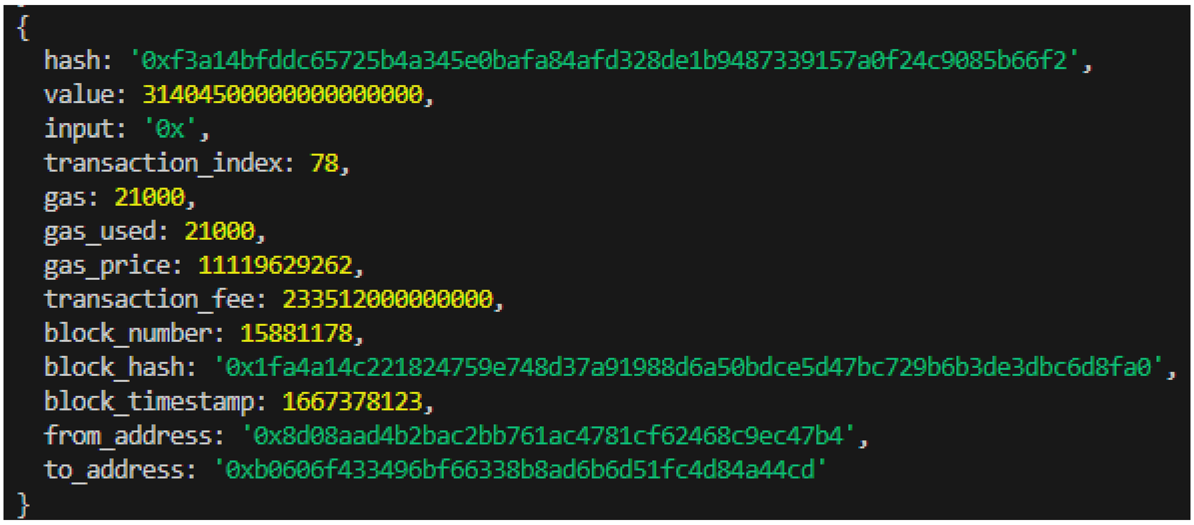
```
{ username: '1', password: '1' }
```

4.5.1.3 Response Format

The response format for these APIs is **JSON**. Each endpoint returns JSON objects that contain information related to the request, such as wallet details, transactions, or user authentication results.



```
{
  addressId: '0xb0606f433496bf66338b8ad6b6d51fc4d84a44cd',
  type: 'eoa',
  btc: 1,
  eth: { low: 5, high: 0 }
}
```

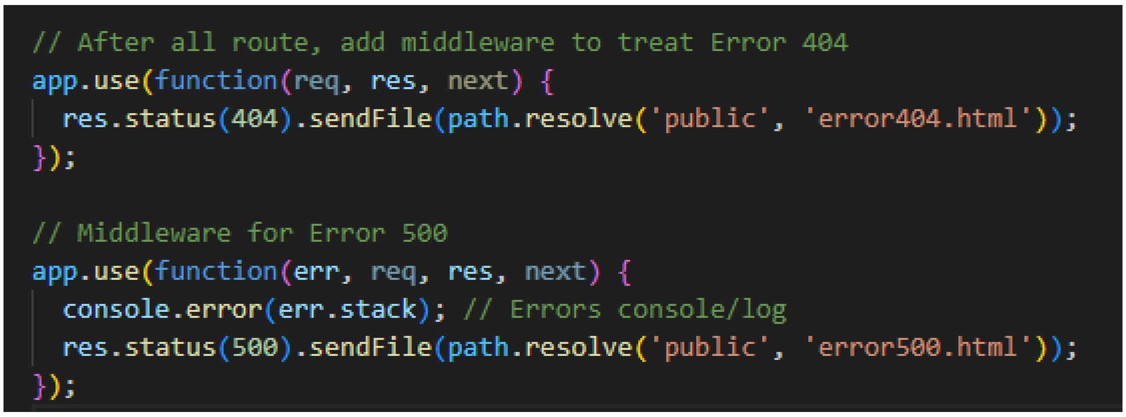


```
{
  hash: '0xf3a14bfddc65725b4a345e0bafa84afd328de1b9487339157a0f24c9085b66f2',
  value: 314045000000000000000,
  input: '0x',
  transaction_index: 78,
  gas: 21000,
  gas_used: 21000,
  gas_price: 11119629262,
  transaction_fee: 233512000000000,
  block_number: 15881178,
  block_hash: '0x1fa4a14c221824759e748d37a91988d6a50bdce5d47bc729b6b3de3dbc6d8fa0',
  block_timestamp: 1667378123,
  from_address: '0x8d08aad4b2bac2bb761ac4781cf62468c9ec47b4',
  to_address: '0xb0606f433496bf66338b8ad6b6d51fc4d84a44cd'
}
```

Figure 17. Terminal displaying fetch data from webpage to database

For successful requests, a 200 OK status code is typically returned along with the relevant data in the response body.

For requests that fail due to client errors, such as invalid parameters or authentication failures, appropriate codes are returned (e.g., 404 Not Found for invalid URLs). Server errors result in a 500 Internal Server Error status code.



```
// After all route, add middleware to treat Error 404
app.use(function(req, res, next) {
  res.status(404).sendFile(path.resolve('public', 'error404.html'));
});

// Middleware for Error 500
app.use(function(err, req, res, next) {
  console.error(err.stack); // Errors console/log
  res.status(500).sendFile(path.resolve('public', 'error500.html'));
});
```

Figure 17. Running middleware for the error of the web

4.5.2 Binance WebSocket API for Real-Time Price Updates

4.5.2.1 Endpoints

Real-time ticker information:

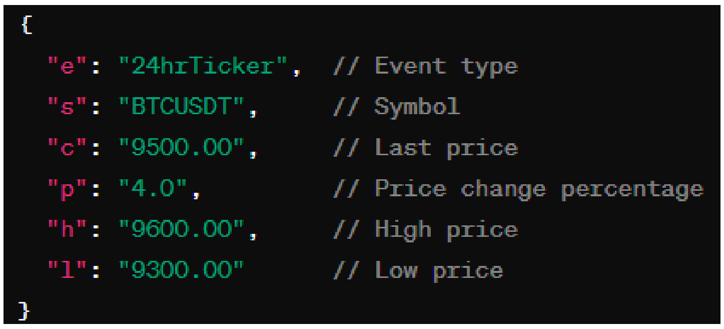
`wss://stream.binance.com:9443/ws/<symbol>@ticker`
All market tickers:
`wss://stream.binance.com:9443/ws/!ticker@arr`

4.5.2.2 Request Parameters

`<symbol>@ticker`: The specific cryptocurrency symbol and ticker information to subscribe to. This is part of the WebSocket URL.

4.5.2.3 Response Format

The WebSocket API sends messages in JSON format containing live market data. Each message includes fields such as:



```
{
  "e": "24hrTicker",   // Event type
  "s": "BTCUSDT",      // Symbol
  "c": "9500.00",      // Last price
  "p": "4.0",          // Price change percentage
  "h": "9600.00",      // High price
  "l": "9300.00"       // Low price
}
```

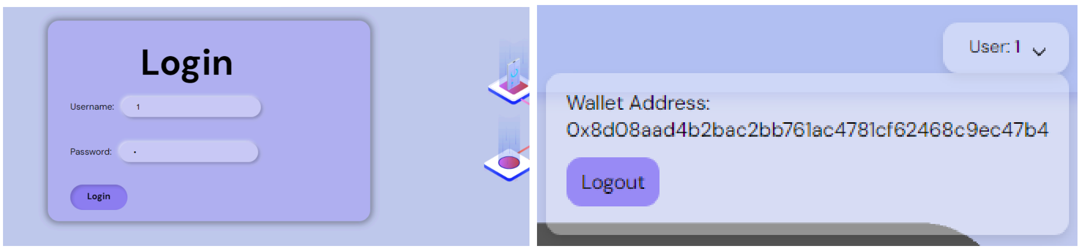
4.6 Function Description

4.6.1 Login

Purpose: Allows users to securely access their accounts on the CryptoSwin platform by entering their username and password. The username and password also be restored and avoid being the same as the other account.

Use Cases: Users must log in to perform trading activities, check their wallet balances, and access personalized features.

- **Step 1:** Open the CryptoSwin website and navigate to the login page.
- **Step 2:** Enter username and password in the respective fields.
- **Step 3:** Click the "Login" button to access the account.



Login
Username: 1
Password: ·
Login

User: 1
Wallet Address:
0x8d08aad4b2bac2bb761ac4781cf62468c9ec47b4
Logout

Figure 18. Login function and user info display

4.6.2 Sign Up

Purpose: Enables new users to register and create an account on CryptoSwin, providing them with access to cryptocurrency trading and wallet management. Each account that has been created will automatically get its new wallet address and it is the only one, different from the others.

Use Cases: New users sign up to start trading, manage their cryptocurrency portfolio, and utilize platform services.

- **Step 1:** Navigate to the Sign Up page from the homepage.
- **Step 2:** Fill in the registration form with username, email, and password.

- **Step 3:** Submit the form to create a new account.

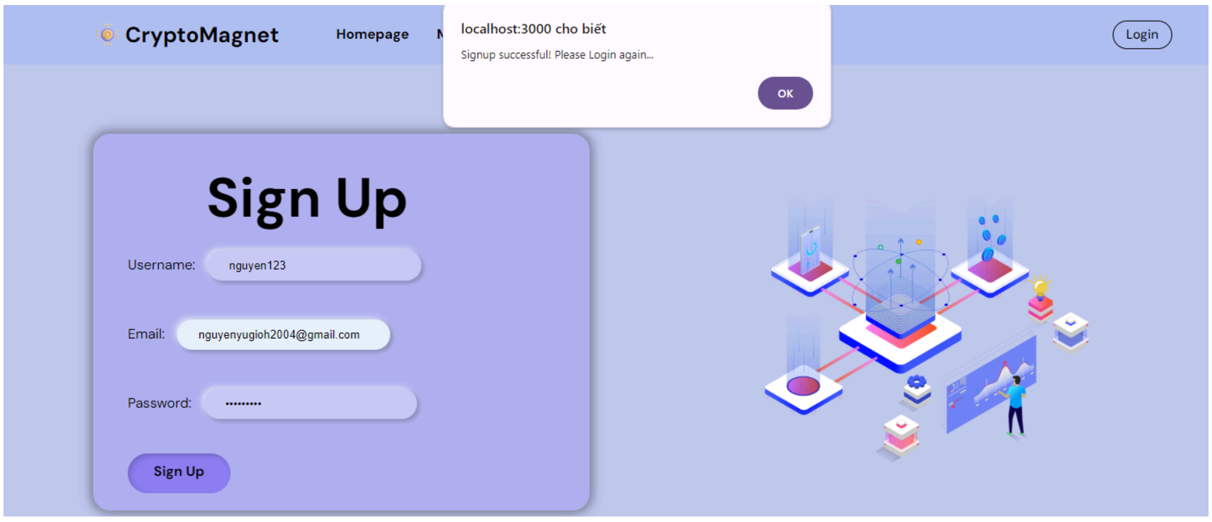


Figure 19. Sign up function to create a new account

4.6.3 Search Wallet

Purpose: To enable users to search for and view detailed information about cryptocurrency wallets. This includes viewing the wallet's balance for different cryptocurrencies and accessing a comprehensive history of transactions associated with the wallet.

Use Cases:

- **Step 1:** The user navigates to the Search Wallet functionality within the CryptoSwin platform.

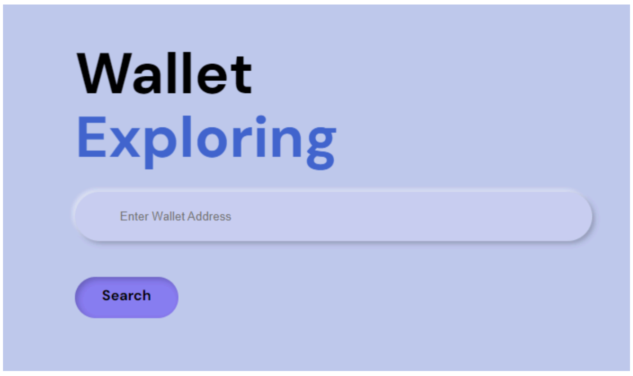
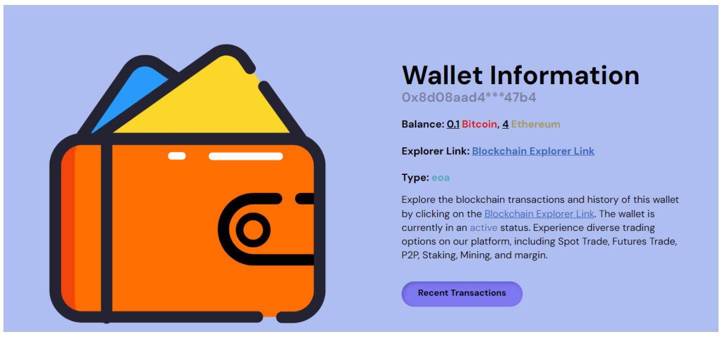


Figure 20. Wallet Address searching information

- **Step 2:** The user enters the unique address of the cryptocurrency wallet they want to search for.
- **Step 3:** Upon submission, the platform gets and displays detailed information about the wallet. This includes the wallet's balance in various cryptocurrencies, and a history of transactions (incoming and outgoing).



Wallet Information
0x8d08aad4***47b4

Balance: 0.1 Bitcoin, 4 Ethereum

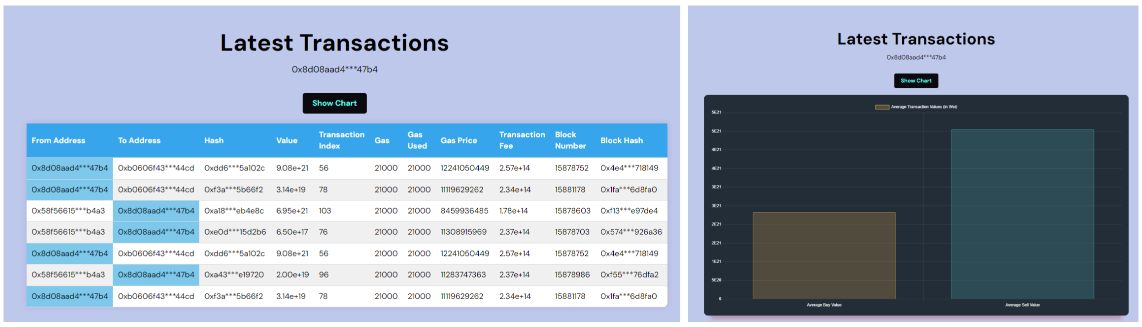
Explorer Link: Blockchain Explorer Link

Type: eoa

Explore the blockchain transactions and history of this wallet by clicking on the Blockchain Explorer Link. The wallet is currently in an active status. Experience diverse trading options on our platform, including Spot Trade, Futures Trade, P2P, Staking, Mining, and margin.

Figure 21. Wallet information and balance

- **Step 4:** Users can inspect specific transaction details, including transaction hashes, amounts transferred, and timestamps, to gain a deeper understanding of the wallet's activity patterns.



Latest Transactions
0x8d08aad4***47b4

From Address	To Address	Hash	Value	Transaction Index	Gas	Gas Used	Gas Price	Transaction Fee	Block Number	Block Hash
0x8d08aad4***47b4	0xb0606f43***44cd	0xdd6***5a102c	9.08e+21	56	21000	21000	12241050449	2.57e+14	15878752	0x4e4***718149
0x8d08aad4***47b4	0xb0606f43***44cd	0xf3a***5b66f2	3.14e+19	78	21000	21000	11119629262	2.34e+14	15881178	0x1fa***6d8fa0
0x58f56615***b4a3	0x8d08aad4***47b4	0xa18***eb4e8c	6.95e+21	103	21000	21000	8459936485	1.78e+14	15878603	0xf13***e97de4
0x58f56615***b4a3	0x8d08aad4***47b4	0xe0d***15d2b6	6.50e+17	76	21000	21000	11308915969	2.37e+14	15878703	0x574***926a36
0x8d08aad4***47b4	0xb0606f43***44cd	0xdd6***5a102c	9.08e+21	56	21000	21000	12241050449	2.57e+14	15878752	0x4e4***718149
0x58f56615***b4a3	0x8d08aad4***47b4	0xa43***e19720	2.00e+19	96	21000	21000	11283747363	2.37e+14	15878986	0xf55***76dfa2
0x8d08aad4***47b4	0xb0606f43***44cd	0xf3a***5b66f2	3.14e+19	78	21000	21000	11119629262	2.34e+14	15881178	0x1fa***6d8fa0

Latest Transactions
0x8d08aad4***47b4

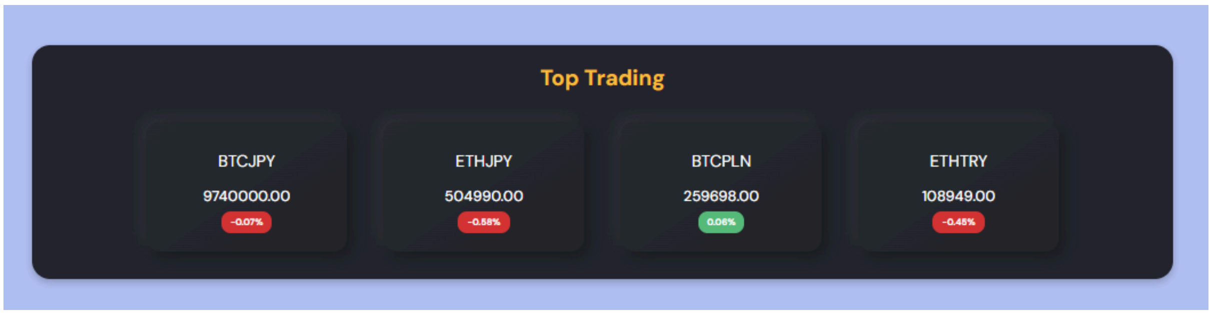
Figure 22. Transaction information and Average value transaction

4.6.4 Market Update

Purpose: This function is within the CryptoSwin application and provides users with real-time information on cryptocurrency prices and market movements. This feature is designed to help users make informed trading decisions by offering up-to-date data on various cryptocurrencies such as Bitcoin (BTC), Ethereum (ETH), BNB, XRP, and others.

Use Cases:

- **Step 1:** Users access the Market Update section on the CryptoSwin platform. This area is dedicated to displaying the latest market data for various cryptocurrencies.

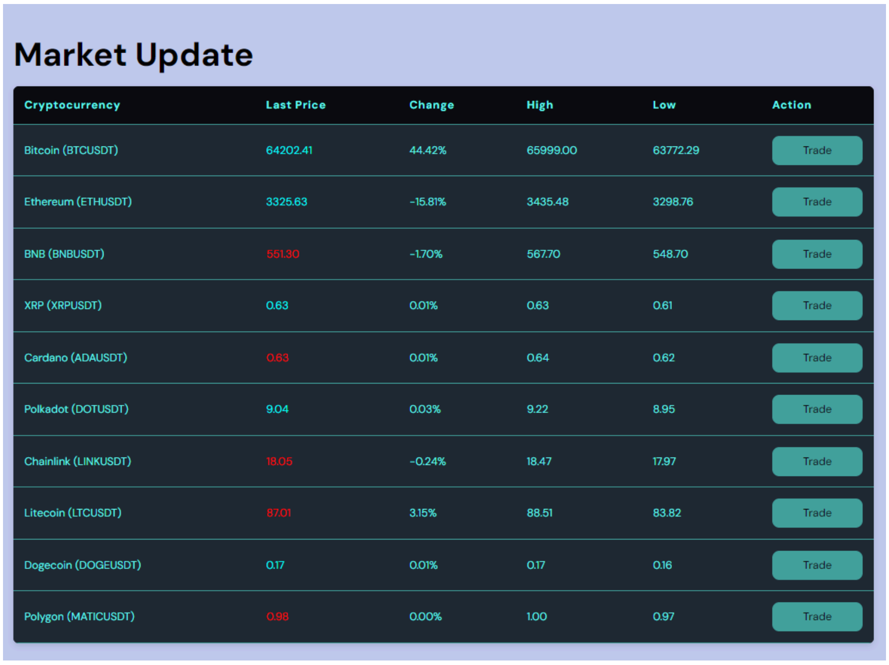


Top Trading

BTCJPY	ETHJPY	BTCPLN	ETHTRY
9740000.00	504990.00	259698.00	108949.00
-0.07%	-0.58%	0.06%	-0.45%

Figure 23. Top trading board

- **Step 2:** Users can view real-time data for multiple cryptocurrencies, including the last price, percentage change, high, low, and a direct link to trade each cryptocurrency. This data is crucial for users looking to buy or sell cryptocurrencies based on current market conditions and gives them the fastest way to get into the crypto market only by clicking trade button and waiting for the Auction of another wallet.

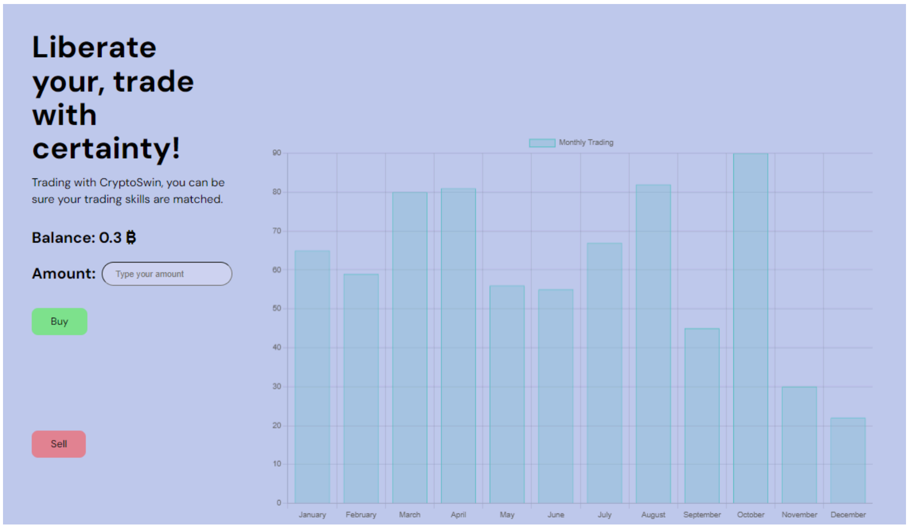


Market Update

Cryptocurrency	Last Price	Change	High	Low	Action
Bitcoin (BTCUSDT)	64202.41	44.42%	65999.00	63772.29	Trade
Ethereum (ETHUSDT)	3325.63	-15.81%	3435.48	3298.76	Trade
BNB (BNBUSDT)	551.30	-1.70%	567.70	548.70	Trade
XRP (XRPUSDT)	0.63	0.01%	0.63	0.61	Trade
Cardano (ADAUSDT)	0.63	0.01%	0.64	0.62	Trade
Polkadot (DOTUSDT)	9.04	0.03%	9.22	8.95	Trade
Chainlink (LINKUSDT)	18.05	-0.24%	18.47	17.97	Trade
Litecoin (LTCUSDT)	87.01	3.15%	88.51	83.82	Trade
Dogecoin (DOGEUSDT)	0.17	0.01%	0.17	0.16	Trade
Polygon (MATICUSDT)	0.98	0.00%	1.00	0.97	Trade

Figure 24. Real-time market price

- **Step 3:** For trading, users can click on the "Trade" button corresponding to the cryptocurrency of interest. This action navigates the user directly to the trading page, where they can execute their trade.



Liberate your, trade with certainty!

Trading with CryptoSwin, you can be sure your trading skills are matched.

Balance: 0.3 ฿

Amount: Type your amount

Figure 25. Trading crypto section

4.6.4 Error Pages (404 and 500)

Purpose: Provides users with informative error messages when a page is not found (404) or an internal server error occurs (500), improving the overall user experience during navigation failures or server issues.

Use Cases: Displayed automatically when a user tries to access a non-existent page or when the server encounters an unexpected error.

- **For a 404 error**: users are informed the page is not found and are provided a link to return to the homepage.
- **For a 500 error**: users are told something went wrong on the server side, with a suggestion to try again later and a link back to the homepage.

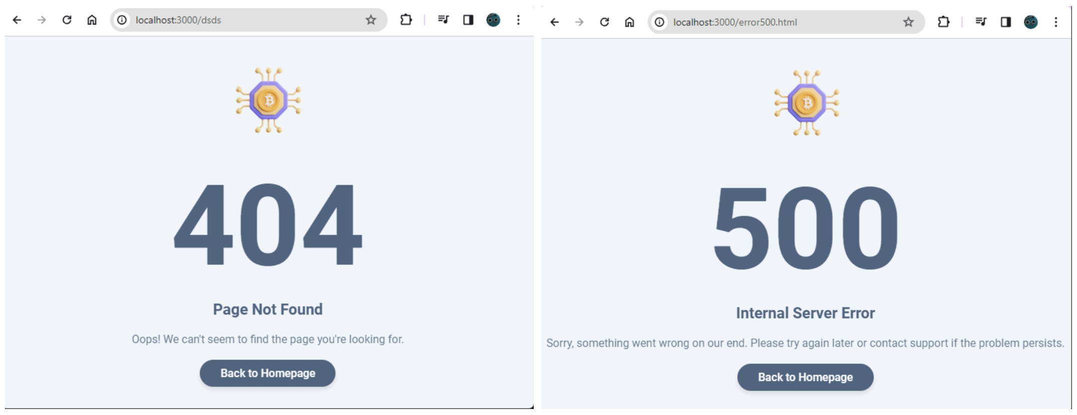

localhost:3000/dsds
404
Page Not Found
Oops! We can't seem to find the page you're looking for.
Back to Homepage
localhost:3000/error500.html
500
Internal Server Error
Sorry, something went wrong on our end. Please try again later or contact support if the problem persists.
Back to Homepage

Figure 26. Error page for some specific situation

4.7 Project Deployment Instruction

Step 1: Install Node.js and npm

Ensure that Node.js and npm (Node Package Manager) are installed on your system. If not, download and install them from Node.js official website.

Step 2: Clone the Project (Extract the zip file)

- Locate the zip file (e.g., CryptoSwin.zip) in your file system.
- Right-click on the zip file and choose "Extract All..." or a similar option depending on your operating system. This will decompress the files into a folder.
- Navigate to the extracted folder. This folder contains all the project files necessary to set up and run the CryptoSwin application.

Step 3: Install npm Modules

Navigate to the project directory and run the following command to install all required npm modules specified in the package.json file:

```
npm install
npm install express
npm install cors
npm install body-parser
npm install neo4j-driver
```

Step 4: Database Setup

- Install Neo4j:

Download and install Neo4j Desktop from the Neo4j website.

- Create a New Database:

Open Neo4j Desktop and create a new project.
Within the new project, click on "Add" and then "Local DBMS" to create a new database.

Configure your database by setting a name, password, and choosing the version. Remember the credentials as you'll need them for the application configuration.

- Import Data:
    - **Option 1: Import data from csv file**

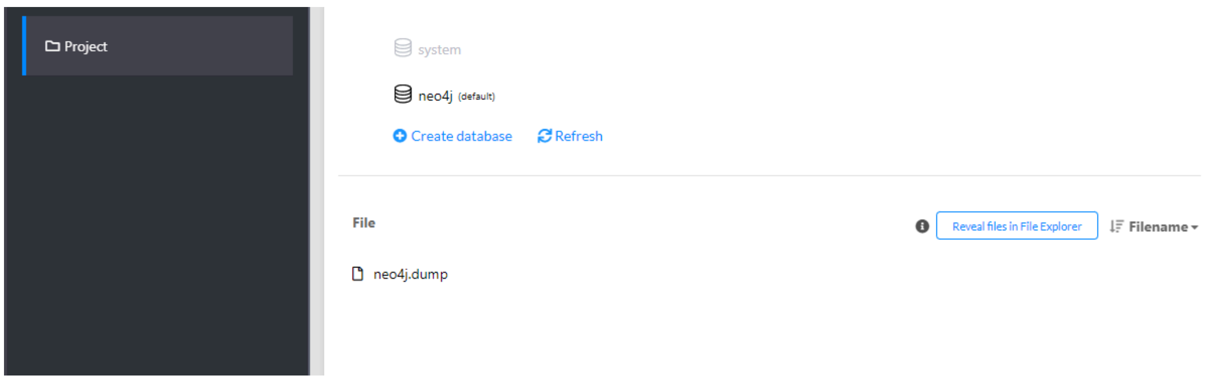
Navigate to C:\Users\.Neo4jDesktop\relate-data\dbmss\database_name\import and paste 2 CSV files (nodes.csv and relationship.csv)

In the Neo4J browser paste this import code to the query input bar:

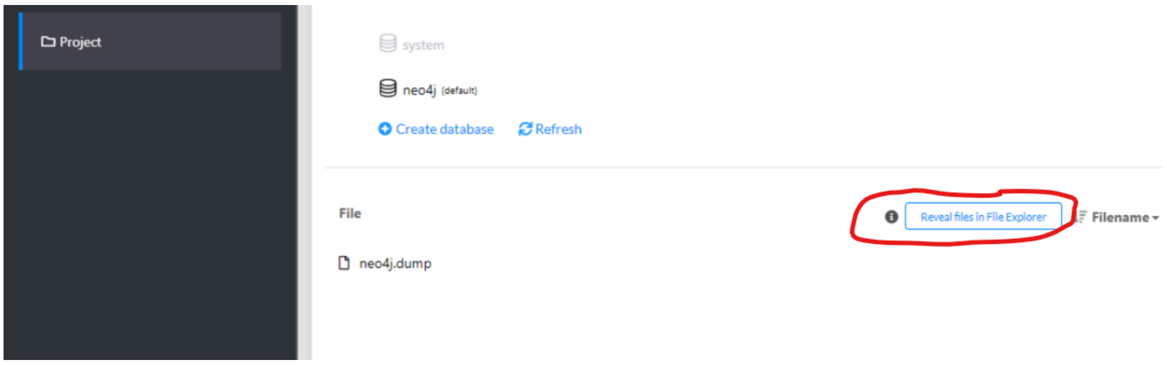
```
//Import the 'nodes.csv' and create or update Address nodes with their type.
LOAD CSV WITH HEADERS FROM 'file:///nodes.csv' AS row
MERGE (address:Address {addressId: row.addressId})
ON CREATE SET address.type = row.type
ON MATCH SET address.type = row.type;

//LOAD btc and eth
LOAD CSV WITH HEADERS FROM 'file:///nodes.csv' AS row
MERGE (address:Address {addressId: row.addressId})
ON CREATE SET address.type = row.type, address.btc = toFloat(row.btc), address.eth =
toInteger(row.eth)
ON MATCH SET address.type = row.type, address.btc = toFloat(row.btc), address.eth =
toInteger(row.eth);

LOAD CSV WITH HEADERS FROM 'file:///nodes.csv' AS row
MERGE (a:Address {addressId: row.addressId})
ON CREATE SET a.username = row.username, a.password = row.password
ON MATCH SET a.username = row.username, a.password = row.password;

//Import the 'relationships.csv' and create the transactions between Address nodes.
LOAD CSV WITH HEADERS FROM 'file:///relationships.csv' AS row
MATCH (from:Address {addressId: row.from_address}), (to:Address {addressId:
row.to_address})
CREATE (from)-[r:TRANSACTION]->(to)
SET r.hash = row.hash,
    r.value = toFloat(row.value),
    r.input = row.input,
    r.transaction_index = toInteger(row.transaction_index),
    r.gas = toInteger(row.gas),
    r.gas_used = toInteger(row.gas_used),
    r.gas_price = toInteger(row.gas_price),
    r.transaction_fee = toFloat(row.transaction_fee),
    r.block_number = toInteger(row.block_number),
    r.block_hash = row.block_hash,
    r.block_timestamp = toInteger(row.block_timestamp);
```

-
    - **Option 2: Import data from dump File**

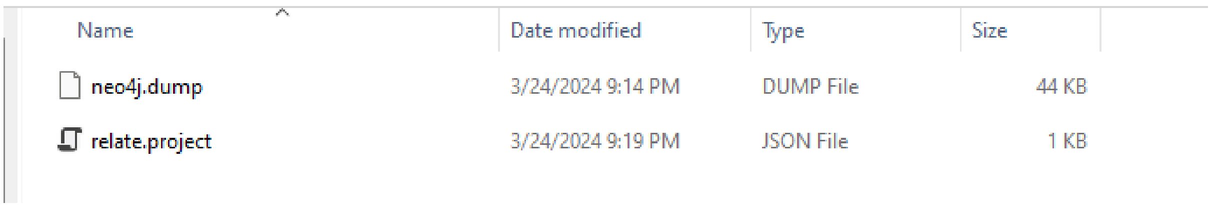
From Neo4j Desktop, create new project



Navigate to the project folder by click “Reveal files in Exlporer”

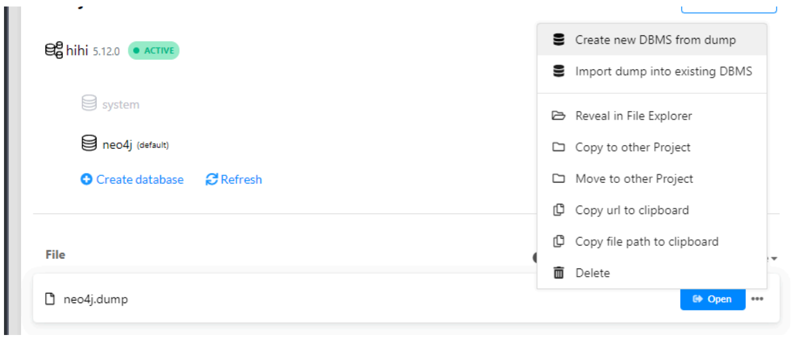


Pastes the neo4j.dump (which is attached in the Assignment2_code.zip file) file to the project folder

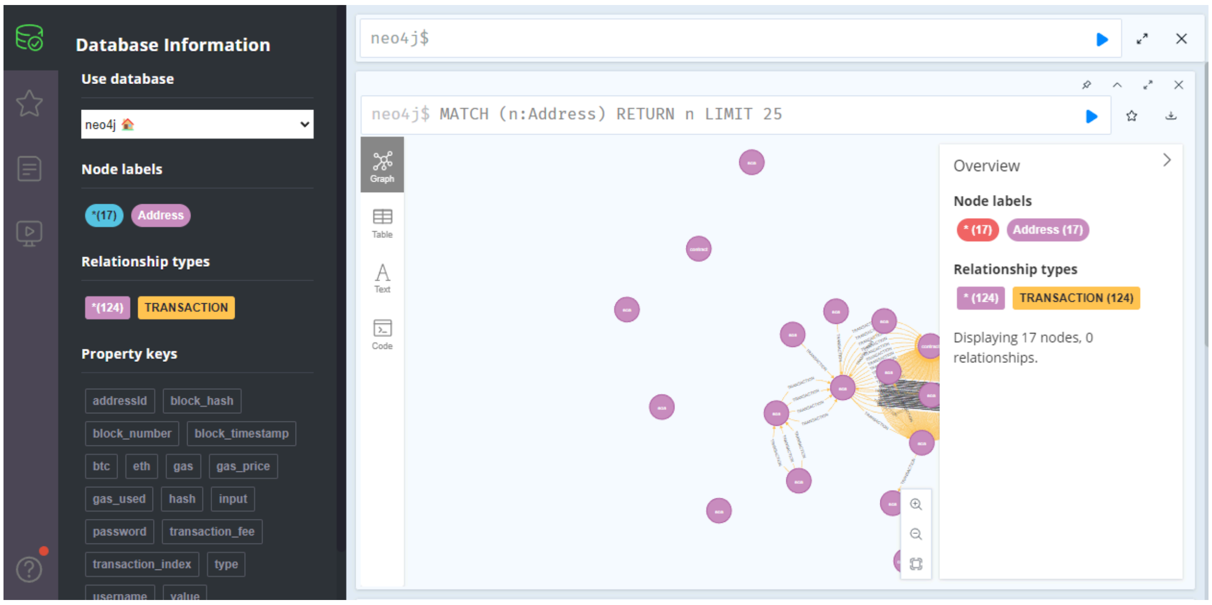


Name	Date modified	Type	Size
neo4j.dump	3/24/2024 9:14 PM	DUMP File	44 KB
relate.project	3/24/2024 9:19 PM	JSON File	1 KB

Move back again to Neo4j Desktop, you will see a dump file in "File" section. Click to "..." next to the neo4j.dump and choose create a new DBMS.



Follow the Create new DBSM flown by setting a DB name and password. Then start and open the created DMSM and navigate to Neo4J Browser. The Graph Database has been created.



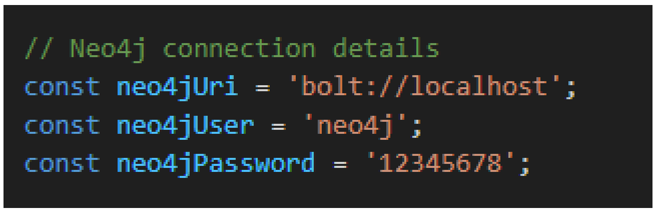
```
neo4j$ MATCH (n:Address) RETURN n LIMIT 25
```

Step 5: Configure the Application

- Update the `server.js` or any relevant configuration files with your Neo4j database credentials and any other environment-specific settings.
- For Neo4j, ensure the following parameters are set correctly:

```
const neo4jUri = 'bolt://localhost';
const neo4jUser = 'neo4j';
const neo4jPassword = 'your_password';
```

- For example:



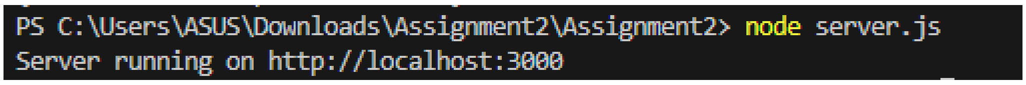
```
// Neo4j connection details
const neo4jUri = 'bolt://localhost';
const neo4jUser = 'neo4j';
const neo4jPassword = '12345678';
```

Figure 27. Connect key to our neo4j database

Step 6: Start the Application

Run the following command in the **Visual Code Studio** (for example) terminal from the root directory of your project to start the server: `node server.js`

This command will launch the CryptoSwin application on the default port or the one you've configured. Access the application through the browser by navigating to http://localhost:<port>, replacing <port> with your configured port number.



```
PS C:\Users\ASUS\Downloads\Assignment2\Assignment2> node server.js
Server running on http://localhost:3000
```

Figure 28. Instruction to run the web

Step 7: Verify Deployment
Once the application is running, navigate through its features to ensure everything is working correctly. Test the functionalities such as logging in, signing up, trading crypto, and viewing the market updates to confirm successful deployment.

Additional Notes:

- Ensure all the paths to static files (CSS, JS, images) in the HTML files are correct.
- If you encounter any npm module-related issues, consult the module's documentation for compatibility information or additional setup instructions.
- The database must be in "Open" status during the testing section.

V. Reference

- Miro.com (2024) Low Fidelity Wireframes. Available at: https://miro.com/app/board/uXjVNx8jn1k=/ (Accessed: 4/2/2024).
- Figma.com (2024) Crypto_figma. Available at: https://www.figma.com/file/dAj5dSYKj5egQkX3eQwhRG/Crypto_figma?type=design&node-id=0-1&mode=design&t=2w1SfGoSrFNY59oN-0 (Accessed: 4/2/2024).
- Lucid.app, (2024) User Flow Diagram. Available at: https://lucid.app/lucidspark/e05ee300-517e-45bb-8152-68f58d7dc0cd/edit?viewport_loc=-3066%2C-1701%2C12953%2C6156%2C0_0&invitationId=inv_cf1e8730-644a-4040-957b-b24b9fdc89b1 (Accessed: 4/2/2024).
- *Node.js V21.7.1 documentation* (no date) *Index | Node.js v21.7.1 Documentation*. Available at: https://nodejs.org/docs/latest/api/ (Accessed: 24 March 2024).
- *Change log* (no date) *Binance API Documentation*. Available at: https://binance-docs.github.io/apidocs/spot/en/#change-log (Accessed: 24 March 2024).