Crypto Portfolio App Documentation

<u>Project screenshots at the end</u>

Table of Contents

---

1. Overview

The **Crypto Portfolio App** is a full-stack platform designed to allow users to manage and track their cryptocurrency investments. The app integrates with the MetaMask wallet to fetch real-time asset data and provide users with insights into their portfolios. The platform is divided into frontend and backend components, with backend functionality separated into several microservices.

Key Features:

- **User Authentication:** Users can sign up, log in, and manage passwords securely.
- **MetaMask Integration:** Connect to MetaMask to fetch wallet details and show assets.
- **Portfolio Tracking:** View and track cryptocurrency portfolio, including real-time market data.
- **Charting:** Visualize price movements of cryptocurrencies using interactive charts.
- **Microservices Architecture:** Each service (user, portfolio, cryptocurrency, etc.) is isolated for scalability and modularity.

---

2. Architecture

The system is designed using a **Microservices Architecture**, where each core functionality (authentication, portfolio, cryptocurrency data, etc.) is handled by its respective microservice. The frontend communicates with these services via RESTful APIs.

Frontend (NextJS)

- **Frontend Framework:** NextJS
- **State Management:** Context API / React Query
- **UI Framework:** Material-UI for styling and component consistency
- **Charting:** Chart.js / D3.js / Victory (used for price movement visualization)
- **Wallet Integration:** MetaMask for wallet authentication and tracking assets

Backend (NestJS Microservices)

- **Backend Framework:** Nest.js
- **Microservices:** Each service is a standalone Node.js application that communicates via RESTful APIs.
  - **User Service:** Manages user authentication and profile.
  - **Portfolio Service:** Manages the portfolio data and calculates the portfolio's value.
  - **Cryptocurrency Service:** Fetches real-time cryptocurrency data.
  - **Notification Service:** Sends notifications (if implemented in the future).
  - **Workflow Service:** Handles business logic and integrates the services.

Database:

- **Database Engine:** MongoDB
- **Schema Design:**
  - **Users:** Stores user credentials, authentication tokens, and profile information.
  - **Portfolios:** Stores portfolio data, including asset types, amounts, and user IDs.
  - **Cryptocurrency Data:** Stores cryptocurrency information, including current prices and historical data.

---

3. Frontend (NextJS)

Features Overview

- **User Authentication:** Users can sign up, log in, and reset their password.
- **Wallet Integration:** Using MetaMask, the app fetches wallet information (like token balances) and displays them in the dashboard.
- **Dashboard:** Shows the top 10 cryptocurrencies with their real-time prices and price movements over time.
- **Portfolio Management:** Displays the user's portfolio with real-time updates on portfolio value based on connected wallet data.

Components

- **App.js:** The main component responsible for routing and layout.
- **Dashboard.js:** Displays real-time data, including cryptocurrency prices, portfolio value, and charts.
- **Signup/Login Forms:** Forms for user authentication and registration.
- **Portfolio.js:** Shows the current portfolio holdings.
- **CryptoChart.js:** A reusable component to visualize price movement using charting libraries.

API Integration

The frontend makes API calls to the backend microservices using the following utility function:

```
import axios from 'axios';

const apiClient = axios.create({
  baseURL: process.env.NEXT_PUBLIC_BACKEND_URL || 'http://localhost:3004',
});

export const setAuthToken = (token) => {
  if (token) {
    apiClient.defaults.headers.common['Authorization'] = `Bearer ${token}`;
  } else {
    delete apiClient.defaults.headers.common['Authorization'];
  }
};

export const getPortfolio = async (userId, walletAddress, providerUrl,
tokenAddresses) =>
  apiClient.get(`/workflow/portfolio/${userId}`, {
    params: { walletAddress, providerUrl, tokenAddresses },
  });

export const getTopCryptos = async () =>
apiClient.get('/workflow/crypto/top10');
```

The above example shows how to fetch portfolio data and top cryptocurrencies using the `apiClient`.

---

4. Backend (NestJS Microservices)

Microservices Overview

Each backend microservice is responsible for one core functionality:

User Service

- **Endpoints:**
  - **POST `/workflow/signup`**: User registration.
  - **POST `/workflow/login`**: User login, returns JWT.
  - **POST `/workflow/password`**: Password reset functionality.

Portfolio Service

- **Endpoints:**
    - **GET `/workflow/portfolio/{userId}`**: Fetch user portfolio by `userId`.
    - **GET `/workflow/portfolio/{userId}/value`**: Get the value of a user's portfolio in USD.

Cryptocurrency Service

- **Endpoints:**
    - **GET `/workflow/crypto/top10`**: Fetch top 10 cryptocurrencies with current market data.
    - **GET `/workflow/crypto/historical`**: Fetch historical data for a given cryptocurrency.

Notification Service (Future Enhancement)

- **Endpoints:**
    - **POST `/workflow/notification/send`**: Send notification to users about portfolio changes or market events.

Workflow Service

- This service integrates all the other services and handles the core logic of the application.

API Endpoints

Here are the main API endpoints for each microservice:

User Service:

- **POST `/workflow/signup`**: Registers a new user with credentials.
- **POST `/workflow/login`**: Logs the user in and returns a JWT token for authentication.
- **POST `/workflow/password`**: Allows a user to update their password.

Portfolio Service:

- **GET `/workflow/portfolio/{userId}`**: Fetches the user's portfolio by `userId`.
- **GET `/workflow/portfolio/{userId}/value`**: Fetches the value of the portfolio in USD.

Cryptocurrency Service:

- **GET `/workflow/crypto/top10`**: Fetches the top 10 cryptocurrencies, including their price, market cap, and volume.
- **GET `/workflow/crypto/historical`**: Fetches historical data (e.g., prices) for a specific cryptocurrency over a given range.

Database Design

- **Users Collection:** Stores user data, including:
    - `userId`

    - `email`
    - `passwordHash`
    - `authToken`
    - `createdAt`
- **Portfolios Collection:** Stores user portfolio data, including:
    - `userId`
    - `tokens`: Array of tokens owned by the user with their balances.
    - `totalValue`: Calculated total portfolio value in USD.
- **Cryptocurrency Data:** Stores cryptocurrency market data, including:
    - `symbol`: Cryptocurrency symbol (e.g., BTC, ETH).
    - `price`: Current price of the token.
    - `marketCap`: Market capitalization.
    - `historicalData`: Array of price data over time.

---

5. Deployment

Local Setup

1. Clone the repository and install dependencies as described in the **Setup** section.
2. Ensure that MongoDB is running locally, or use a cloud service like MongoDB Atlas.
3. Use `docker-compose` to run all services together, or run them individually in development mode.

Cloud Deployment

For cloud deployment, Docker is used to containerize the application. The app can be deployed on platforms like AWS, GCP, or Heroku. The following steps are typically followed:

1. Build the Docker images for each microservice.
2. Push the images to a container registry (Docker Hub, AWS ECR, etc.).
3. Deploy each service on a cloud provider using Kubernetes, ECS, or Docker Compose.

---

6. Testing

Unit Testing

Each microservice should have unit tests written to validate individual functions and logic. Tools like **Jest** and **Mocha** can be used for testing backend services, while **React Testing Library** can be used for testing React components.

Integration Testing

Integration tests should verify that the different services work together. Use **Postman** or **Supertest** to simulate API requests and validate responses.

Example Test for the Portfolio Service (Jest):

```
const request = require('supertest');
const app = require('../app');

describe('GET /workflow/portfolio/:userId', () => {
  it('should return portfolio data for the given userId', async () => {
    const response = await request(app).get('/workflow/portfolio/123');
    expect(response.status).toBe(200);
    expect(response.body).toHaveProperty('portfolio');
  });
});
```

---

7. Additional Notes

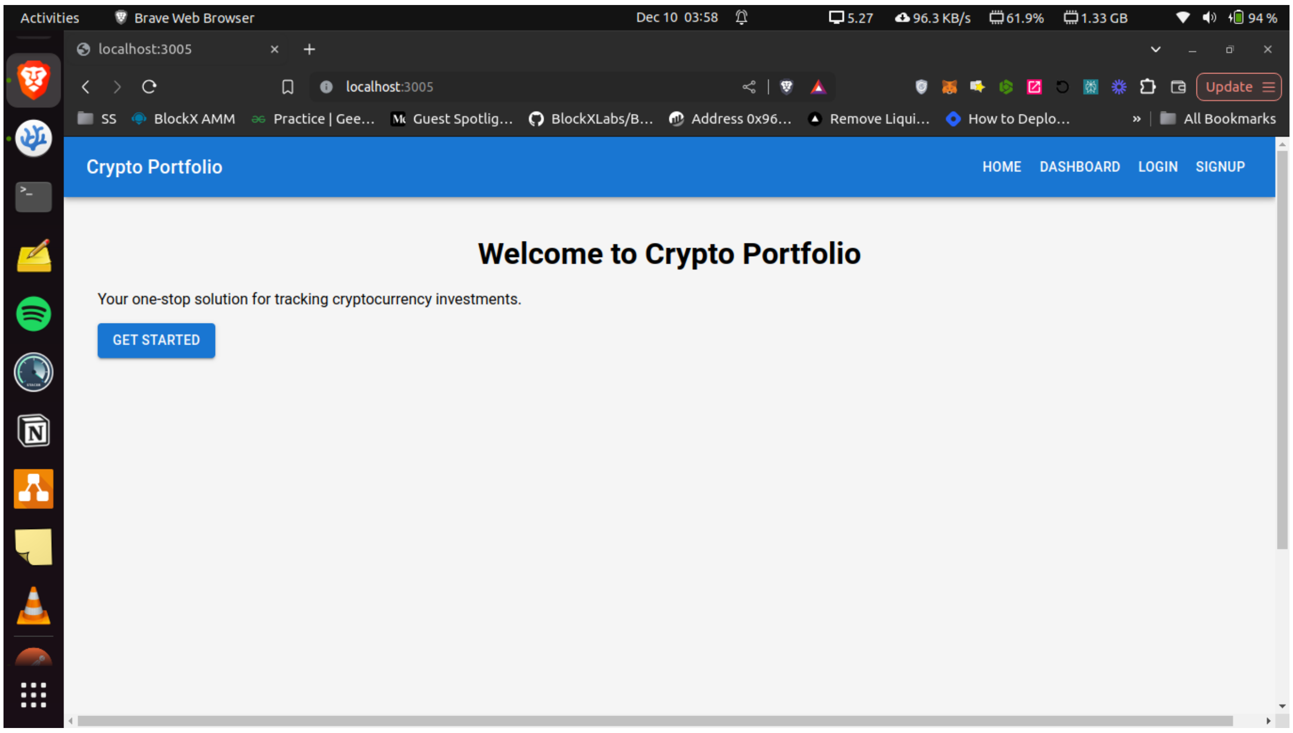
- **Error Handling:** All services should implement proper error handling for API requests. Return appropriate HTTP status codes (e.g., 400 for bad requests, 500 for internal server errors).
- **Authentication:** JWT tokens should be securely stored and verified to ensure that only authorized users can access their portfolios.

---

8. Future Enhancements

- **Push Notifications:** Implement push notifications to alert users of portfolio changes or significant market events.
- **Mobile App:** A mobile version of the app using React Native could be developed for cross-platform compatibility.
- **Advanced Charting:** Add more advanced charting features like candlestick charts, moving averages, etc.

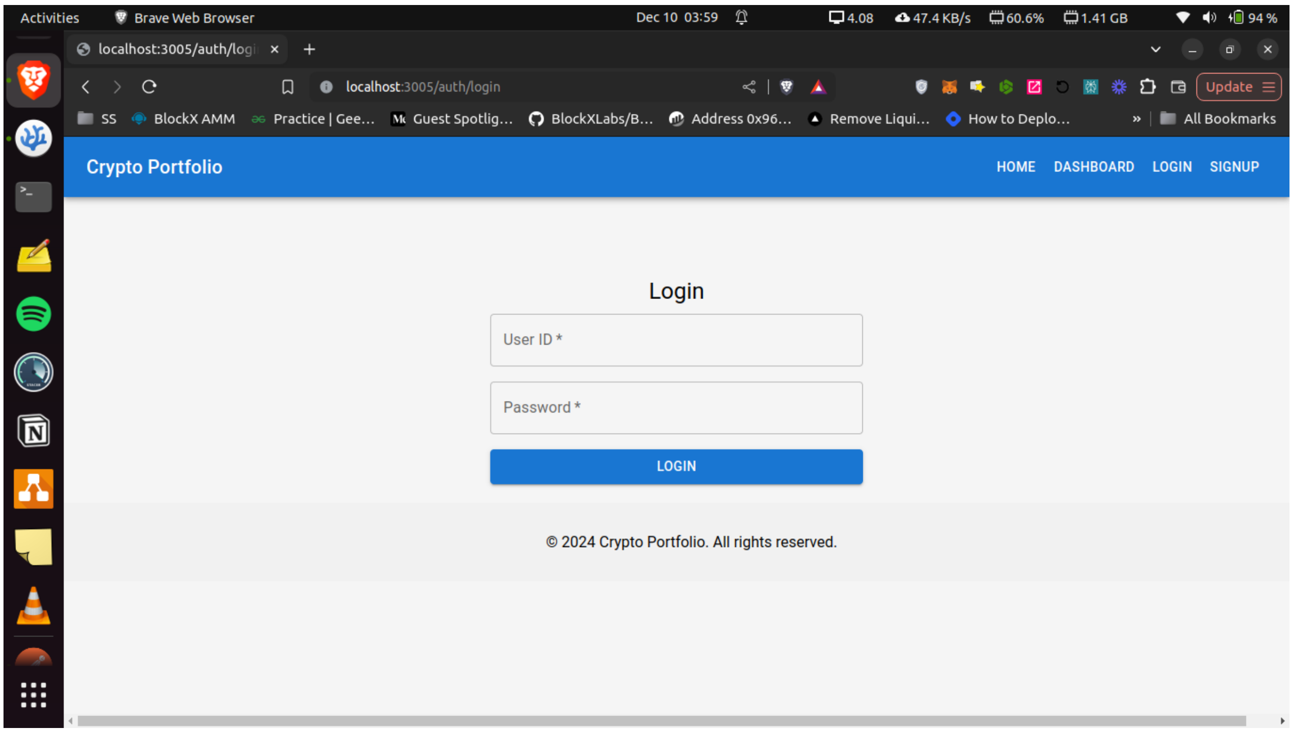
This documentation provides a comprehensive guide for understanding and using the Crypto Portfolio App. It includes detailed instructions on setting up the development environment, the microservices architecture, API endpoints, and deployment steps. Let me know if you need further clarifications!



Welcome to Crypto Portfolio

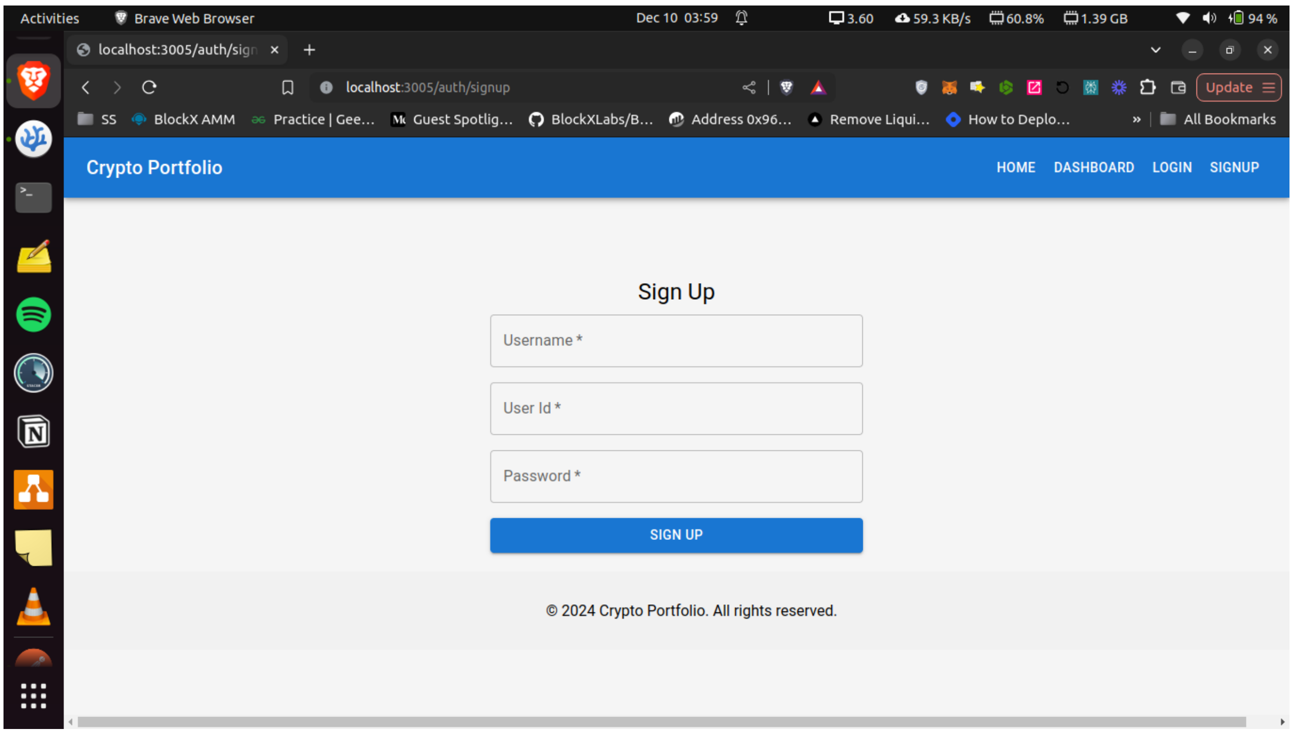
Your one-stop solution for tracking cryptocurrency investments.

<u>Landing page</u>



© 2024 Crypto Portfolio. All rights reserved.

<u>Login page</u>


Crypto Portfolio

HOME DASHBOARD LOGIN SIGNUP

Sign Up

Username *

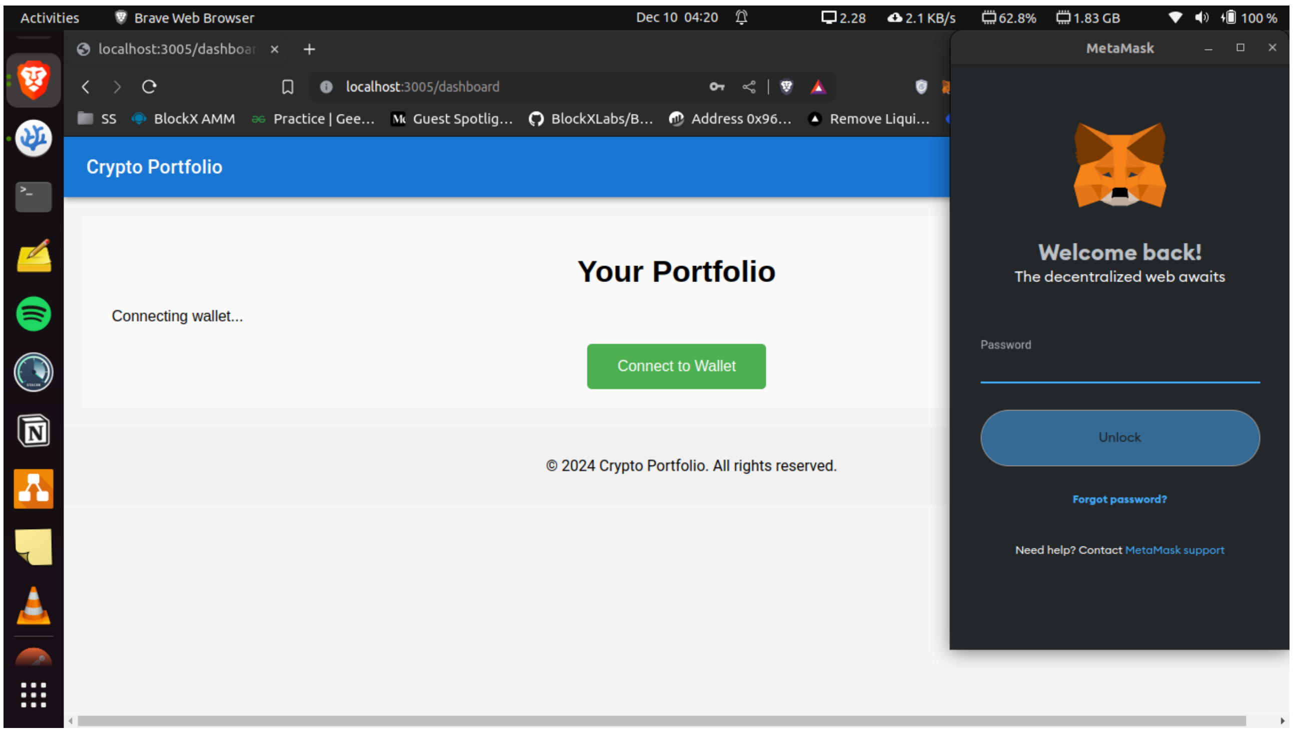
User Id *

Password *

SIGN UP

© 2024 Crypto Portfolio. All rights reserved.

Sign up


Crypto Portfolio

Your Portfolio

Connecting wallet...

Connect to Wallet

© 2024 Crypto Portfolio. All rights reserved.

MetaMask

Welcome back!

The decentralized web awaits

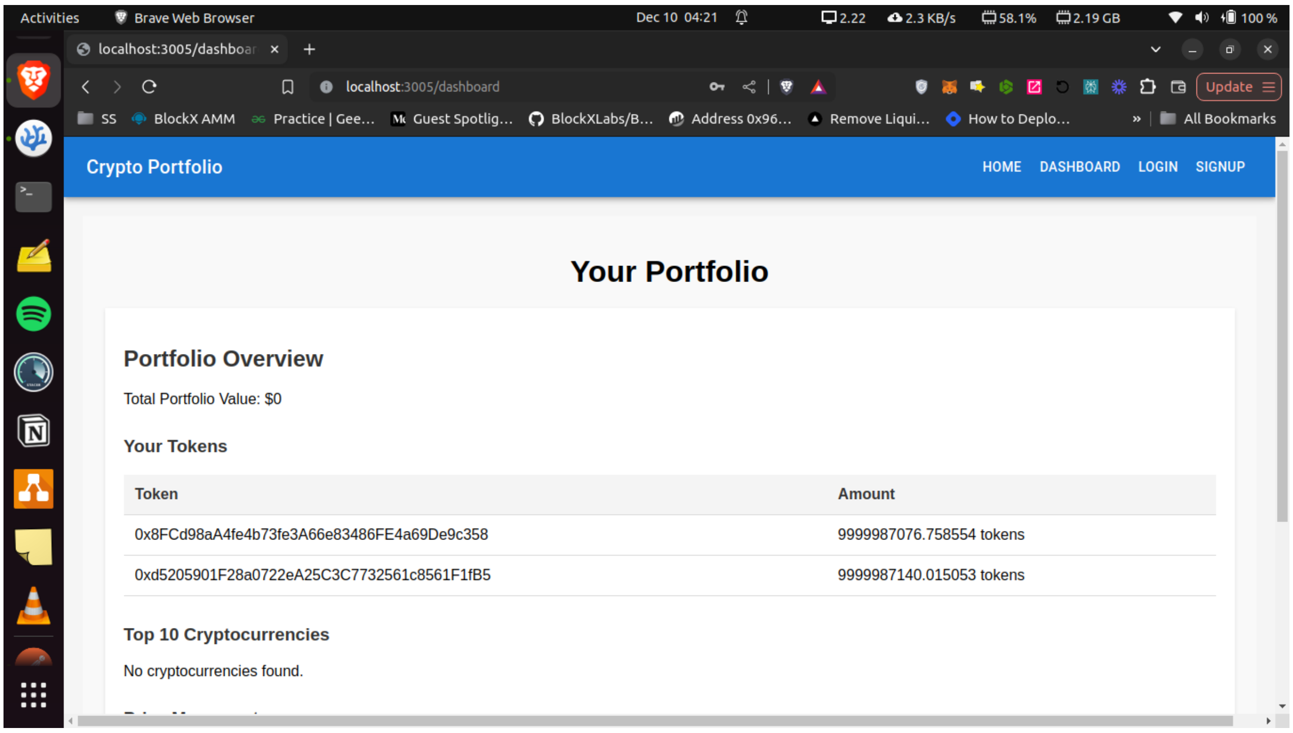
Password

Unlock

Forgot password?

Need help? Contact MetaMask support

Metamask integration



Your Portfolio

Portfolio Overview

Total Portfolio Value: $0

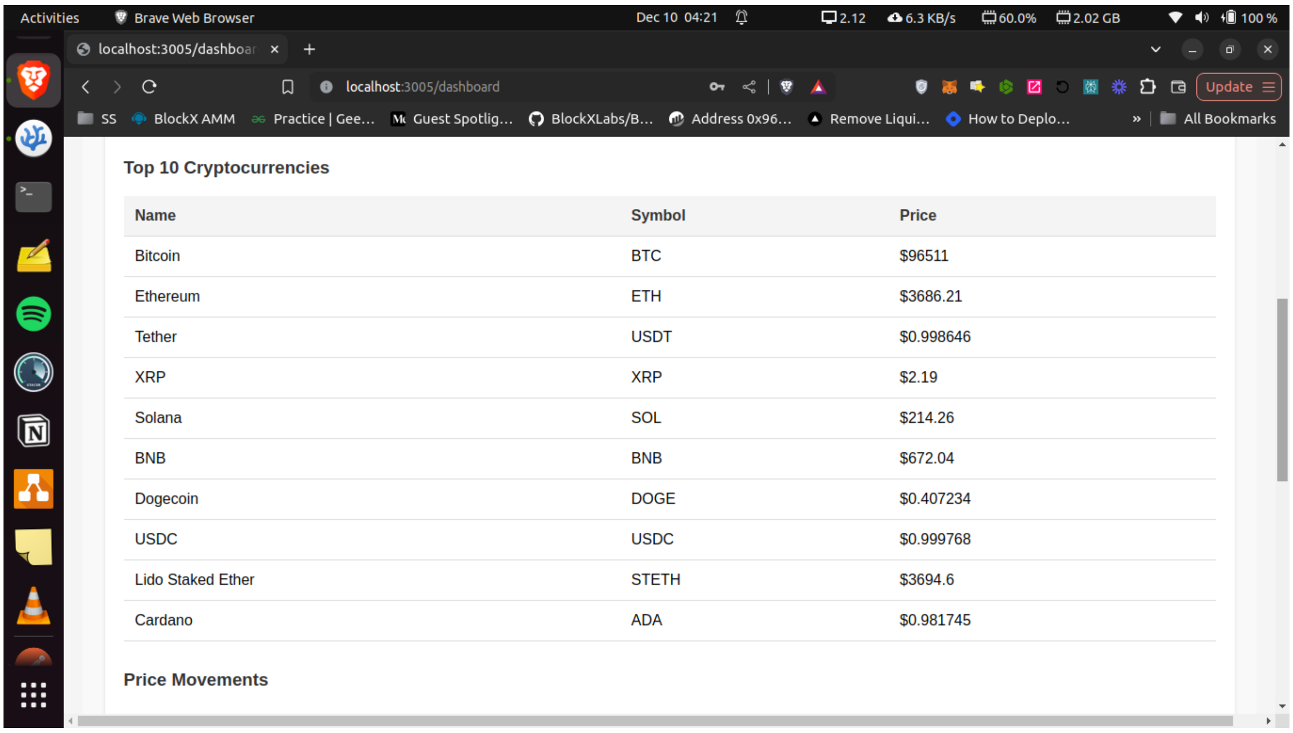
Your Tokens

Token	Amount
0x8FCd98aA4fe4b73fe3A66e83486FE4a69De9c358	9999987076.758554 tokens
0xd5205901F28a0722eA25C3C7732561c8561F1fB5	9999987140.015053 tokens

Top 10 Cryptocurrencies

No cryptocurrencies found.

Portfolios based on their MetaMask wallet data (chain : https://web3.blockxnet.com)

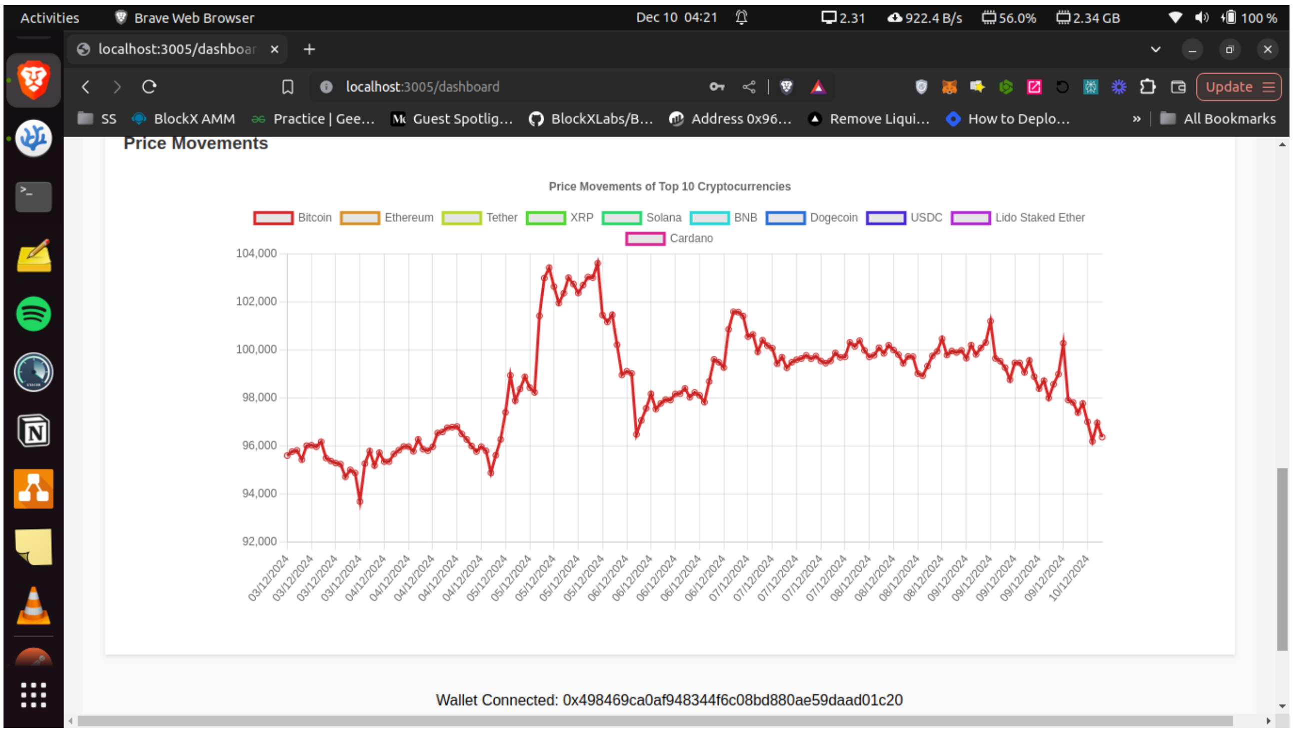


Top 10 Cryptocurrencies

Name	Symbol	Price
Bitcoin	BTC	$96511
Ethereum	ETH	$3686.21
Tether	USDT	$0.998646
XRP	XRP	$2.19
Solana	SOL	$214.26
BNB	BNB	$672.04
Dogecoin	DOGE	$0.407234
USDC	USDC	$0.999768
Lido Staked Ether	STETH	$3694.6
Cardano	ADA	$0.981745

Price Movements

Last price for the top 10 cryptocurrencies



Wallet Connected: 0x498469ca0af948344f6c08bd880ae59daad01c20

Price movements over time using a chart (Chart.js)