

# **Tracking Crypto Prices with BitNexus**

## **Introduction to Bitcoin (BCSE325L)**

### **DIGITAL ASSIGNMENT REPORT**

**Winter Semester 2024-2025**

Submitted by

**ISAM ABDUL AZIZ (22BKT0103)**

*in partial fulfillment for the award of the degree of*

**B. Tech**

In

**Computer Science Engineering**

**with Specialization in Blockchain Technology**



Vellore-632014, Tamil Nadu, India

**School of Computer Science and Engineering**

March, 2025.

# Tracking Crypto Prices with BitNexus

## Abstract

BitNexus is a comprehensive cryptocurrency platform designed to provide users with real-time data, transaction tracking, and various utilities centered around Bitcoin and other digital assets. Leveraging reliable APIs such as CoinGecko and BlockCypher, the website offers a range of features to enhance the user experience and accessibility of cryptocurrency-related information.

The core functionality of BitNexus includes the ability to list the top cryptocurrencies with detailed information such as current price, market capitalization, trading volume, and percentage changes over different time frames. Users can delve deeper into individual cryptocurrencies by clicking on them, where an interactive chart displays their price trends over the past 365 days. The time range of the chart can be adjusted, allowing users to analyze the asset's performance from days to months, providing flexibility for both short- and long-term analysis.

Additionally, the platform allows users to track Bitcoin transactions by entering a Bitcoin address. This feature shows the last 10 transactions associated with the address, with the option to load more transactions in increments of 10. A built-in balance tracker also displays the amount of Bitcoin held by the entered address, enhancing transparency and providing a clear overview of the wallet's holdings. To make the data more accessible, an export feature is available, allowing users to download their transaction history as a CSV file for further analysis or record-keeping.

Another key feature is the generation of a downloadable QR code for Bitcoin addresses, simplifying the process of receiving Bitcoin payments. The QR code feature helps users easily share their Bitcoin address with others.

BitNexus also offers multi-currency support, including the Indian Rupee, the Euro, and the US Dollar, allowing users to view cryptocurrency prices in their preferred currency. The website dynamically updates in real-time to reflect the selected currency, ensuring that all information remains accurate and relevant. BitNexus serves as a versatile and user-friendly platform, combining essential cryptocurrency tools and data into a seamless experience for users worldwide.

## Introduction

The rise of cryptocurrencies has significantly altered the financial landscape in recent years. These decentralized digital currencies, such as Bitcoin, Ethereum, and a variety of others, have seen rapid adoption worldwide. Their potential to provide users with financial sovereignty, alongside the promise of lower transaction fees, privacy, and security, has driven both retail and institutional interest. However, navigating the cryptocurrency ecosystem effectively requires access to accurate data, seamless transaction tracking, and simple tools for managing assets. BitNexus emerges as a comprehensive platform that aims to address these needs, offering users a sophisticated suite of tools designed to enhance their cryptocurrency experience.

At its core, BitNexus is focused on making cryptocurrency data accessible and actionable. With the integration of APIs such as CoinGecko for real-time price and market data, and BlockCypher for transaction tracking, BitNexus allows users to track the performance of top cryptocurrencies effortlessly. By providing detailed market metrics, including prices, market capitalization, and volume, BitNexus offers valuable insights for both novice and experienced traders, helping them stay informed about the market's fluctuations. With the ability to view detailed information on individual coins, users can explore trends, analyze historical data, and make more informed decisions based on the most up-to-date figures.

One of the standout features of BitNexus is its charting capabilities. The platform offers a dynamic chart that displays a cryptocurrency's price over the past 365 days. The time range is adjustable, giving users the flexibility to view price changes over a day, week, month, or year. This feature provides a powerful tool for traders and investors, offering them a visual representation of a cryptocurrency's price action and potential trends. The charting system also adds an extra layer of analysis, allowing users to better understand the price movements of their assets.

In addition to market data, BitNexus offers a unique transaction tracking feature designed specifically for Bitcoin. Users can input their Bitcoin address and view their last 10 transactions, providing a quick overview of their recent activity. This feature is highly beneficial for individuals who need to keep track of their Bitcoin transactions for personal, accounting, or tax-related purposes. For those with extensive transaction histories, BitNexus also offers a "load more" option, enabling users to access more transactions in increments of 10. This seamless interface allows for an uninterrupted review of transaction histories and provides the user with a clear view of their Bitcoin activity.

Furthermore, BitNexus includes a balance tracker for Bitcoin addresses. This tool allows users to quickly check the amount of Bitcoin stored in a particular wallet address. The balance tracker provides clarity and transparency, enabling users to stay updated on the state of their holdings. To further enhance its utility, BitNexus enables users to export their transaction history as a CSV file. This export feature ensures that users can easily save, share, or analyze their transaction data offline, making it a convenient option for those who wish to keep detailed records.

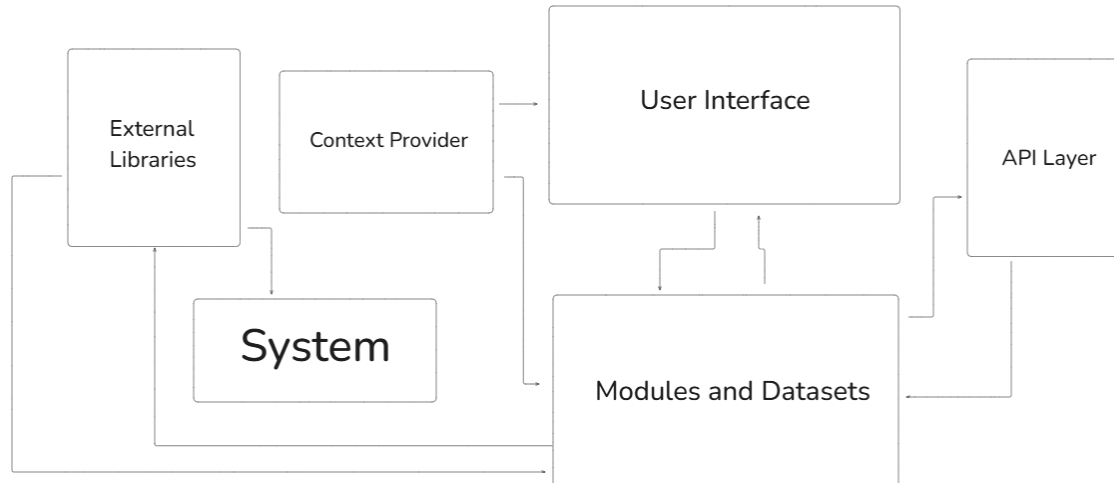
Another key feature of BitNexus is its Bitcoin QR code generator. By simply entering a Bitcoin address, users can instantly generate a QR code linked to that address. This QR code can be downloaded, making it easier to share Bitcoin payment information. Whether for

personal transactions, business use, or receiving donations, the QR code feature simplifies the process of receiving Bitcoin, enhancing the user experience by eliminating the need for manual address entry.

BitNexus also recognizes the importance of global accessibility. With support for multiple currencies, including INR (Indian Rupee), EUR (Euro), and USD (United States Dollar), users can view cryptocurrency prices in their preferred local currency. This multi-currency support ensures that BitNexus remains useful to a diverse global audience. The platform dynamically updates all prices in real-time based on the selected currency, allowing users to make more informed decisions regardless of their location.

In the fast-evolving world of cryptocurrencies, staying informed and having the right tools is crucial. BitNexus provides a holistic solution for anyone engaged in the cryptocurrency space, whether they are active traders, casual investors, or Bitcoin users looking to track transactions. By combining real-time market data, transaction tracking, charting tools, and Bitcoin address utilities into one user-friendly platform, BitNexus enables its users to access essential cryptocurrency information effortlessly. As the platform continues to evolve and grow, its commitment to providing a seamless, transparent, and efficient experience remains at the forefront, ensuring that BitNexus remains a trusted companion for anyone navigating the world of digital assets.

## Architecture Diagram



## Background Study

Cryptocurrency and blockchain technology have revolutionized the financial landscape, offering decentralized solutions for transactions, investments, and data storage. With the growing popularity of cryptocurrencies, the need for accessible and reliable tools to track market data, monitor wallet balances, and analyze transactions has also risen. Platforms like CoinGecko provide real-time data on market trends, prices, and coin statistics, while blockchain explorers enable users to track and analyze transactions. This website aims to address these needs by integrating various features such as fetching crypto data, viewing transaction histories, exporting data, and generating QR codes for wallet addresses. Through

these tools, users can engage with cryptocurrencies more efficiently, gaining valuable insights into their investments and transactions.

## Methodology:

### CoinContextProvider Module:

The `CoinContextProvider` component is responsible for fetching cryptocurrency data and managing the global state related to cryptocurrencies and the selected currency. It makes the data available to all child components through the React Context API.

#### 1. State Initialization:

- The component uses React's `useState` to create two pieces of state:
  - `allCoin`: Holds the list of cryptocurrencies and their associated market data.
  - `currency`: Stores the selected currency (e.g., USD, INR) and its corresponding symbol (e.g., "\$").

#### 2. Data Fetching:

- The `fetchAllCoin` function is an asynchronous function that fetches cryptocurrency market data from the CoinGecko API.
- The API request is made using the selected currency (`currency.name`), and the data is then stored in the `allCoin` state.

#### 3. Data Update:

- When the currency state changes (e.g., if the user switches the currency from USD to INR), the `useEffect` hook runs and triggers the `fetchAllCoin` function again to fetch new data based on the updated currency.
- The `allCoin` state is updated with the new data, ensuring that the list of cryptocurrencies reflects the selected currency.

#### 4. Context Provision:

- The component wraps its children components inside a `CoinContext.Provider`, passing the current values of `allCoin`, `currency`, and `setCurrency` as part of the context value. This allows child components to access and modify these values without having to pass them down as props.

### CryptocurrencyList Module:

The `CryptocurrencyList` Module is designed to display and filter a list of cryptocurrencies. It utilizes data fetched from the `CoinContext` and manages the display and filtering based on user input. The component uses React hooks to maintain local state for the displayed coins and user input.

### 1. Context and State Management:

- The component retrieves the list of cryptocurrencies (allCoin) and the current currency (currency) from the global state provided by the coinContext. It maintains its own local states:
  - displayCoin: Stores the filtered list of coins that will be shown to the user. Initially, it is set to an empty array but will be updated as the list is filtered or reset.
  - input: Tracks the value of the search input field where the user can type the name of a cryptocurrency to filter the list.

### 2. Filtering Logic:

- The inputHandle function updates the input state whenever the user types in the search field. If the input field is empty, it resets the displayed list (displayCoin) to show all available coins from the allCoin array.
- The searchHandle function is called when the user submits the search form. It filters the allCoin list by checking if the cryptocurrency name includes the search input (case-insensitive). The filtered list is then set to the displayCoin state.

### 3. Rendering the List:

- The useEffect hook ensures that whenever the allCoin data is updated (e.g., after fetching new data), the displayCoin list is reset to show all the cryptocurrencies. This ensures that the user sees the updated list of cryptocurrencies when the data changes.

## CoinDetails Module:

The CoinDetails Module is responsible for fetching and displaying detailed information about a specific cryptocurrency, including its current data and historical price charts. The module interacts with the CoinGecko API to retrieve this data and manage it within the component. It utilizes React hooks to manage state and lifecycle events such as fetching data and handling loading states.

### 1. Fetching Coin Data:

- The fetchCoinData function fetches detailed information about a specific cryptocurrency based on its unique coinId. The function uses the CoinGecko API to get information such as the cryptocurrency's name, symbol, current price, market cap, and other relevant details.
- The function sets a loading state (setLoading(true)) before making the API request and sets it back to false after the request completes. If the data is fetched successfully, it stores the response in the coinData state.

### 2. Fetching Historical Data:

- The `fetchHistoricalData` function fetches historical price data for the selected cryptocurrency. This data is used to render a chart showing the price of the coin over a certain period (e.g., the last 30 days, 365 days, etc.).
- The function takes a `selectedDays` parameter, which specifies how many days of historical data to fetch. It sends a request to the CoinGecko API's `/market_chart` endpoint, passing the `coinId`, `currency.name`, and the number of `selectedDays`.
- Like `fetchCoinData`, the loading state is managed to indicate that the data is being fetched, and the results are stored in the `historicalData` state.

### 3. Dynamic Data Fetching with `useEffect`:

- The `useEffect` hook is used to trigger data fetching whenever the `coinId`, `currency`, or `days` state changes.
- It first calls `fetchCoinData` to fetch the latest data for the selected cryptocurrency based on the `coinId`.
- After that, it calls `fetchHistoricalData` to fetch historical price data, passing in the number of days specified by the `days` state.

## Transactions Module:

The Transactions Module is responsible for managing and displaying the transactions related to a specific Bitcoin address. It fetches transaction details from the BlockCypher API and retrieves the current Bitcoin price from the CoinGecko API to allow users to see both the Bitcoin value and the equivalent currency value for each transaction. It also provides functionality to export transaction data to a CSV file.

### 1. State Management:

- `address`: Stores the Bitcoin address entered by the user.
- `transactions`: Holds the list of transactions associated with the provided Bitcoin address.
- `btcPrice`: Stores the current price of Bitcoin in the selected currency (USD, INR, EUR).
- `loading`: A boolean state that indicates whether data is being fetched.
- `error`: Stores any errors that occur while fetching data.
- `lastTransaction`: Keeps track of the last transaction's hash to load more transactions when required.

### 2. Fetching Bitcoin Price (`fetchBtcPrice`):

- This function fetches the current price of Bitcoin in the selected currency from the CoinGecko API. It updates the `btcPrice` state with the fetched value.

- It runs whenever the currency state changes, ensuring that the displayed Bitcoin price is up-to-date with the user's selected currency.
3. Fetching Transactions (fetchTransactions):
    - This function fetches the transaction data for a specific Bitcoin address using the BlockCypher API. It fetches all transactions for the address, and supports pagination to fetch additional transactions by using the before parameter, which is the hash of the last fetched transaction (lastTransaction).
    - The function updates the transactions state with the new transactions and appends them to the previously fetched data. It also updates the lastTransaction state to keep track of the last fetched transaction.
  4. Transaction Data Calculation:
    - The total amount of Bitcoin spent across all transactions is calculated by iterating over the transactions array. For each transaction, the sum of all output values is added, then converted from satoshis (1 BTC = 100,000,000 satoshis) to Bitcoin (BTC).
    - The total amount in the selected currency is calculated by multiplying the total Bitcoin spent by the current Bitcoin price (btcPrice).
  5. Exporting Data to CSV (csvData):
    - The csvData is generated by iterating over the list of transactions. For each transaction, it calculates the amount in both Bitcoin (BTC) and the selected currency. The data is then structured into an object with the transaction hash, amount in BTC, amount in the selected currency, the currency, the date the transaction was received, and the number of confirmations.
    - This structured data can later be exported to a CSV file, allowing users to download the transaction history.
  6. Balance Fetching for CSV (fetchBalanceForCSV):
    - The fetchBalanceForCSV function fetches the current balance of a Bitcoin address from the BlockCypher API. It stores the balance in Bitcoin, converting it from satoshis to BTC. The balance is used for purposes like exporting to CSV.

#### BalanceTracker Module:

The BalanceTracker Module, when integrated with the transactions page, allows users to track and display the Bitcoin balance of a given address. It interacts with the BlockCypher API to fetch the balance and manages user interactions, errors, and loading states to provide a seamless experience.

1. State Management:



- **address:** Stores the Bitcoin address entered by the user. This address is used to query the BlockCypher API for the balance associated with it.
- **balance:** Stores the current balance of the provided Bitcoin address. The balance is initially set to null and is updated with the result from the API call.
- **loading:** A boolean state used to indicate whether the balance is being fetched. It is set to true when the balance fetch is in progress and false when the process is complete.
- **error:** Holds any error messages that occur while fetching the balance (e.g., if the Bitcoin address is invalid or if the fetch fails).

## 2. Fetching Bitcoin Balance (fetchBalance):

- This function is triggered when the user enters a Bitcoin address and initiates a balance fetch.
- The function first checks if an address has been entered. If not, it returns early, preventing the API call.
- **Loading state:** The loading state is set to true when the fetch begins, providing a visual cue to the user that data is being loaded.
- **Error handling:** If the API responds with an error (e.g., an invalid Bitcoin address), the function sets the error state with an appropriate message ("Invalid Bitcoin address") and resets the balance state to null.
- **Successful response:** If the balance is successfully fetched, it is converted from satoshis (the smallest unit of Bitcoin) to Bitcoin (BTC) by dividing the balance by  $1e8$ . The balance is then stored in the balance state.
- **Error handling for fetch issues:** If there is a problem with the fetch (e.g., network error), the function sets the error state with a message ("Failed to fetch balance").

## QRcode Module:

The QRcode Module allows users to generate a QR code for a given Bitcoin address. It enables users to enter their Bitcoin address, generate a QR code based on that address, and download the QR code as an image file. The module uses `html2canvas` to capture the rendered QR code as an image, enabling the download functionality.

### 1. State Management:

- **bitcoinAddress:** This state holds the Bitcoin address entered by the user. The address is used to generate the corresponding QR code.
- **qrGenerated:** A boolean state that tracks whether the QR code has been generated. It determines whether the QR code should be displayed or not after the address has been provided.

- qrRef: A reference (useRef) to the QR code element in the DOM. This is used to capture the rendered QR code for the download functionality.
2. Handling Bitcoin Address Input (handleBitcoinAddressChange):
    - This function updates the bitcoinAddress state when the user types into the input field. It ensures that the entered address is captured and stored in the state for generating the QR code.
  3. Generating QR Code (handleGenerateQR):
    - This function is called when the user wants to generate the QR code. It checks if a valid Bitcoin address has been entered. If an address is provided, it sets the qrGenerated state to true, which triggers the display of the QR code.
    - The QR code itself would likely be rendered using a library such as qrcode.react, which converts the entered Bitcoin address into a QR code.
  4. Downloading QR Code (handleDownloadQR):
    - This function enables the user to download the generated QR code as a PNG image file. It uses the **html2canvas** library to capture the QR code (referenced by qrRef) as a canvas element.
    - Once the QR code is rendered as a canvas, the function creates a downloadable link with the canvas data as the image URL. The user can then click the link to download the QR code image.

## Proposed Model

The website is composed of the modules mentioned earlier. These modules are interconnected in a way that facilitates the seamless interaction of data and functionality on the website. Below is an explanation of how these modules work together, starting from the global state management to the specific functionalities like fetching cryptocurrency data, displaying transaction history, tracking Bitcoin balances, and generating a QR code for a Bitcoin address.

### 1. CoinContextProvider Module

#### Purpose:

The CoinContextProvider module is responsible for managing the global state of the cryptocurrencies and the selected currency (USD, INR, EUR). It fetches the cryptocurrency market data from the CoinGecko API and makes it available to the rest of the application through React context.

- **Global State Management:** This module manages the list of all coins (allCoin) and the selected currency (currency). It provides this data to other modules via React's useContext hook, allowing other components to access and update the global state.

- **API Fetching:** It fetches the cryptocurrency market data for the selected currency (USD, INR, EUR) and stores it in allCoin. Whenever the selected currency changes, the context is updated, and the data is refetched.
- **Integration with Other Modules:** Other modules like the CryptocurrencyListModule and CoinDetailsModule depend on the data provided by this context (e.g., for displaying cryptocurrencies and their details).

## 2. CryptocurrencyList Module

### Purpose:

This module displays a list of cryptocurrencies fetched by the CoinContextProvider. It allows the user to search for specific cryptocurrencies and view details like their price, market cap, and volume.

- **Data Access via Context:** This module uses useContext to access the allCoin data (which is managed by CoinContextProvider) and renders the list of cryptocurrencies on the page.
- **Search Functionality:** Users can search for cryptocurrencies by name. When the user types into the search input, the module filters the list based on the input and updates the displayed coins accordingly.
- **Currency Context:** The list of cryptocurrencies is displayed in the selected currency (USD, INR, EUR). When the user changes the currency, the CoinContextProvider fetches new data, and the list is automatically updated.

## 3. CoinDetails Module

### Purpose:

The CoinDetailsModule displays detailed information about a specific cryptocurrency when selected. This includes a price chart that shows the price history for the selected cryptocurrency over the past year, along with other coin-specific details like market cap, volume, etc.

- **Data Fetching:** This module fetches data from the CoinGecko API to get the details of a specific cryptocurrency (identified by coinId). The coinId is passed down as a prop or accessed through routing.
- **Historical Data:** It also fetches historical price data, enabling the user to adjust the time range for the price chart (e.g., 30 days, 90 days, or 365 days). This is again influenced by the selected currency.
- **Currency Context:** The price and historical data are adjusted based on the selected currency (USD, INR, EUR). This integration ensures that all data displayed to the user is consistent with their chosen currency.

## 4. Transactions Module

### Purpose:

This module allows users to enter a Bitcoin address and view the last 10 transactions

associated with it. It also allows users to load more transactions and provides a balance tracker.

- **Transaction Data Fetching:** This module fetches Bitcoin transaction data using the BlockCypher API. It displays the last 10 transactions and allows users to load more transactions in increments of 10.
- **Balance Tracker Integration:** It includes a balance tracker that queries the BlockCypher API to display the current balance of a given Bitcoin address.
- **Currency Context:** It fetches the current Bitcoin price using the CoinGecko API, adjusting the displayed value of Bitcoin transactions to the selected currency (USD, INR, EUR).
- **Export to CSV:** The module also includes a feature to export the transactions to a CSV file. The CSV data includes transaction hashes, amounts in BTC, and the selected currency.

## 5. BalanceTrackerModule

**Purpose:**

This module allows users to enter a Bitcoin address and check the current balance of the address. It uses the BlockCypher API to get this data and ensures the correct amount is displayed in Bitcoin.

- **Address Input:** The user enters a Bitcoin address, and the module fetches the balance for that address.
- **Loading and Error States:** The module manages loading and error states to inform the user if the balance is being fetched or if there's an issue (e.g., invalid address).
- **Currency Context:** Like the Transactions Module, this module also uses the selected currency (USD, INR, EUR) to display the balance in the correct format (BTC to USD/INR/EUR).

## 6. QRcode Module

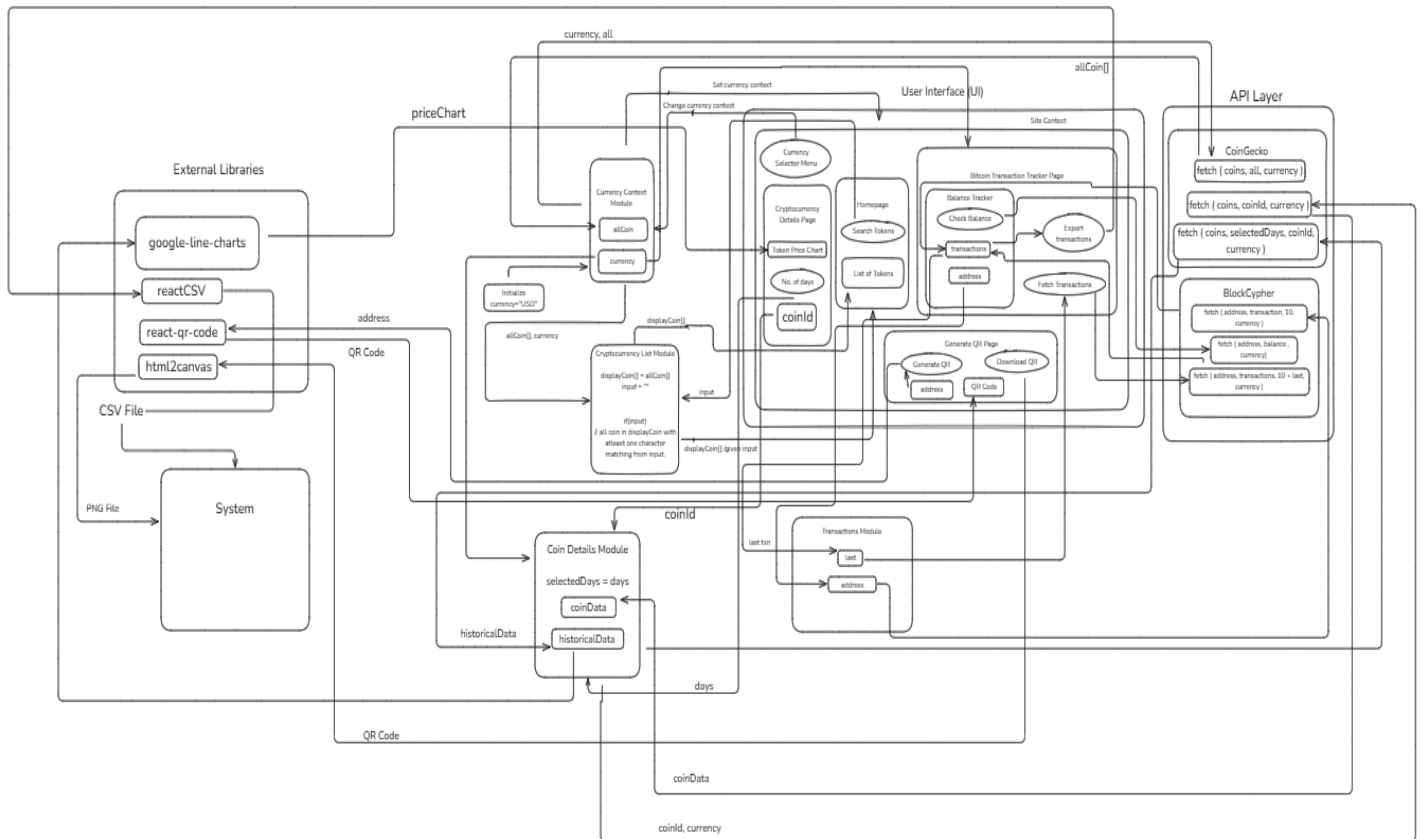
**Purpose:**

This module allows users to generate a QR code for a Bitcoin address. The QR code can be scanned by others to send Bitcoin to the specified address. The module also supports downloading the generated QR code.

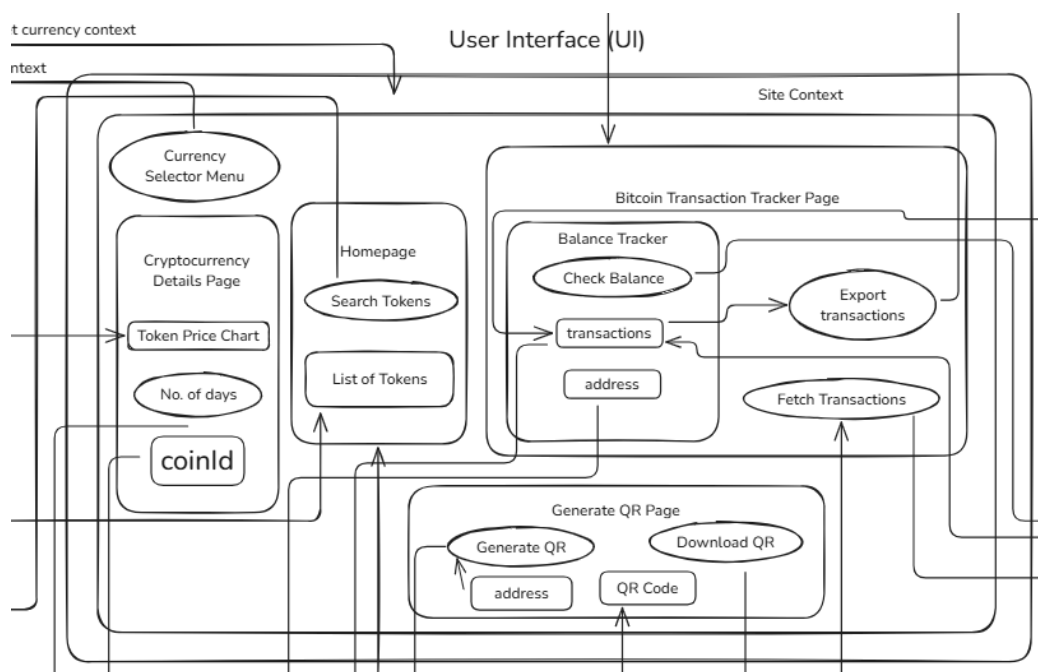
- **Address Input:** The user enters a Bitcoin address, and the QR code for that address is generated. The module utilizes html2canvas to capture the rendered QR code and provides a download link for the user to save the QR code as an image.
- **State Management:** The module manages the bitcoinAddress state for storing the entered address and the qrGenerated state for tracking whether the QR code has been successfully generated.
- **Transaction Context:** While not directly dependent on transaction history, the QR code feature complements the transaction flow by allowing users to easily share their

Bitcoin address via a QR code, which could be used in conjunction with the transaction page.

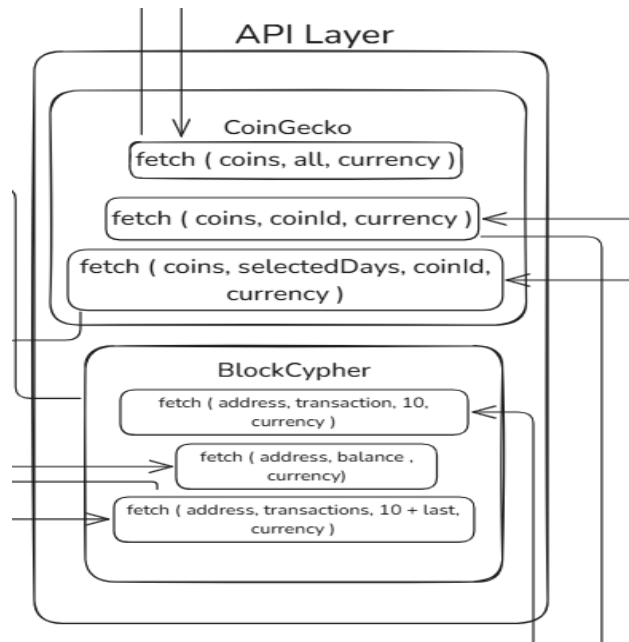
## Integrated Diagram:



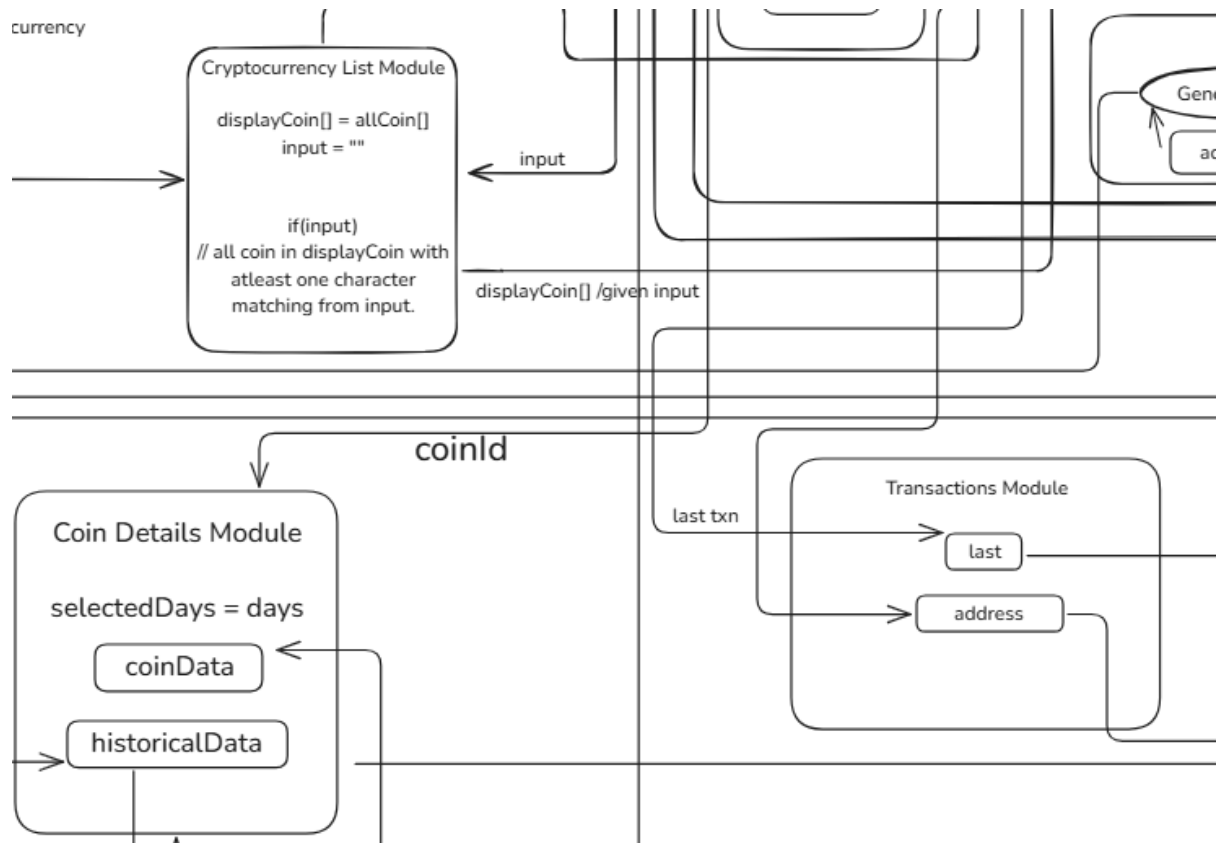
## User Interface:



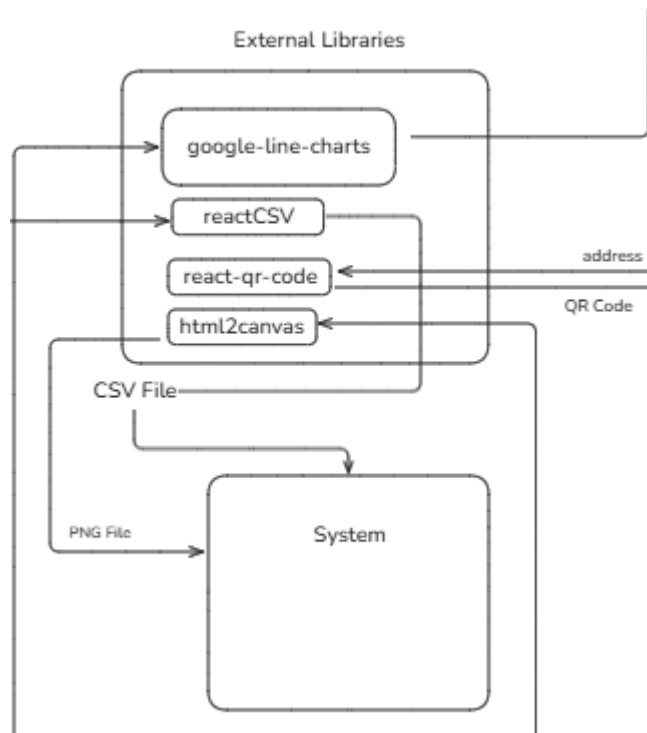
## API Layer:



## Utilities:



External Libraries:



## Results and Discussion

Website Live Link: <https://pricetrack-roan.vercel.app/>

Sample Source Code:

App.jsx:

```
import React from "react";
import Navbar from "../components/Navbar";
import Footer from "../components/Footer";
import { Routes, Route } from "react-router-dom";
import Home from "../pages/Home";
import Coin from "../pages/Coin";
import Bitcoin from "../pages/Bitcoin";
import About from "../pages/About";
import QRcode from "../pages/QRCode";

const App = () => {
  return (
    <div className="p-0 m-0 box-border">
      <div className="min-h-screen bg-gradient-to-b from-[#5b0f4e] to-[#002834] h-full w-full">
        <Navbar />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/coin/:coinId" element={<Coin />} />
        </Routes>
      </div>
    </div>
  );
};
```

```

        <Route path="/bitcoin" element={<Bitcoin />} />
        <Route path="/qrcode" element={<QRcode />} />
        <Route path="/about" element={<About />} />
      </Routes>
      <Footer />
    </div>
  </div>
);
};

export default App;

```

Coin.jsx:

```

if (loading) {
  return (
    <div className="grid place-self-center min-h-screen">
      <div className="w-[65px] h-[65px] place-self-center border-5 border-
[#bdbdbd] border-t-[#4500c6] rounded-full animate-spin"></div>
    </div>
  );
}

if (!coinData || !historicalData) {
  return <p>Please wait, fetching data for you.</p>;
}

return (
  <div className="p-2">
    <div className="flex flex-col items-center gap-5 my-10 mx-auto">
      <img src={coinData.image.large} alt="Coin Image" />
      <p className="text-amber-100 font-bold text-5xl">
        <b>
          {coinData.name} ({coinData.symbol.toUpperCase()})
        </b>
      </p>
    </div>
    <div className="m-auto h-[350px] max-w-[800px]">
      <LineChart historicalData={historicalData} />
    </div>
    <div className="flex justify-center my-5">
      <select
        className="p-2 bg-[#5b0f4e] text-white border border-[#002834]
rounded-lg focus:outline-none"
        value={days}
        onChange={(e) => setDays(e.target.value)}
      >
        {[7, 10, 30, 90, 180, 365].map((day) => (

```



```

        <option key={day} value={day}>
            {day} Days
        </option>
    )))
</select>
</div>
<div className="text-gray-100 max-w-[600px] mx-auto my-20 flex flex-
col">
    <ul className="flex flex-row justify-between px-10 py-0 my-3 border-b-
2 border-b-[#5f5d5f] list-none">
        <li>Crypto Market Rank</li>
        <li className="font-bold">{coinData.market_cap_rank}</li>
    </ul>
    <ul className="flex flex-row justify-between px-10 py-0 my-3 border-b-
2 border-b-[#5f5d5f] list-none">
        <li>Current Price</li>
        <li className="font-bold">
            {currency.symbol}{ " "}
            {coinData.market_data.current_price[currency.name].toLocaleString(
)}}
        </li>
    </ul>
    <ul className="flex flex-row justify-between px-10 py-0 my-3 border-b-
2 border-b-[#5f5d5f] list-none">
        <li>Market Cap</li>
        <li className="font-bold">
            {currency.symbol}{ " "}
            {coinData.market_data.market_cap[currency.name].toLocaleString()}
        </li>
    </ul>
    <ul className="flex flex-row justify-between px-10 py-0 my-3 border-b-
2 border-b-[#5f5d5f] list-none">
        <li>24 Hour High</li>
        <li className="font-bold">
            {currency.symbol}{ " "}
            {coinData.market_data.high_24h[currency.name].toLocaleString()}
        </li>
    </ul>
    <ul className="flex flex-row justify-between px-10 py-0 my-3 border-b-
2 border-b-[#5f5d5f] list-none">
        <li>24 Hour Low</li>
        <li className="font-bold">
            {currency.symbol}{ " "}
            {coinData.market_data.low_24h[currency.name].toLocaleString()}
        </li>
    </ul>
</div>
</div>

```

```
);  
};
```

Modules:

CoinContextProvider:

```
export const coinContext = createContext();  
const CoinContextProvider = (props) => {  
  const [allCoin, setAllCoin] = useState([]);  
  const [currency, setCurrency] = useState({  
    name: "usd",  
    symbol: "$",  
  });  
  
  const fetchAllCoin = async () => {  
    const options = {  
      method: "GET",  
      headers: {  
        accept: "application/json",  
        "x-cg-demo-api-key": "CG-1234567890",  
      },  
    };  
  
    try {  
      const response = await fetch(  
        `https://api.coingecko.com/api/v3/coins/markets?vs_currency=${currency.name}`,  
        options,  
      );  
      const data = await response.json();  
      setAllCoin(data);  
    } catch (error) {  
      console.error("Error occurred: ", error);  
    }  
  };  
  
  useEffect(() => {  
    fetchAllCoin();  
  }, [currency]);  
  
  const contextValue = {  
    allCoin,  
    currency,  
    setCurrency,  
  };  
  
  return (  
    <coinContext.Provider value={contextValue}>  
      {props.children}  
    </coinContext.Provider>  
  );  
};
```

CryptocurrencyList Module:

(Integrated within Home.jsx component)

```
const Home = () => {
  const { allCoin, currency } = useContext(coinContext);
  const [displayCoin, setDisplayCoin] = useState([]);
  const [input, setInput] = useState("");

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

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

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

BalanceTracker Module:

```
const API_URL = "https://api.blockcypher.com/v1/btc/main/addrs/";

export default function BalanceTracker() {
  const [address, setAddress] = useState("");
  const [balance, setBalance] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");

  async function fetchBalance() {
    if (!address) return;
    setLoading(true);
    setError("");

    try {
      const response = await fetch(
        `${API_URL}${address}/balance?token=`,
      );
      const data = await response.json();

      if (data.error) {
        setError("Invalid Bitcoin address.");
        setBalance(null);
      } else {
        setBalance(data.balance / 1e8); // Convert satoshis to BTC
      }
    } catch (err) {
      setError("Failed to fetch balance.");
    } finally {
      setLoading(false);
    }
  }

  return (

```

CoinDetails Module:

(Integrated within Coin.jsx)

```
const Coin = () => {
  const { coinId } = useParams();
  const [coinData, setCoinData] = useState(null);
  const [historicalData, setHistoricalData] = useState(null);
  const { currency } = useContext(coinContext);
  const [loading, setLoading] = useState(true);
  const [days, setDays] = useState(365);

  useEffect(() => {
    const fetchCoinData = async () => {
      setLoading(true);
      const options = {
        method: "GET",
        headers: {
          accept: "application/json",
          "x-cg-demo-api-key": " ",
        },
      };
      try {
        const response = await fetch(
          `https://api.coingecko.com/api/v3/coins/${coinId}`,
          options,
        );
        const data = await response.json();
        setCoinData(data);
      } catch (error) {
        console.error("Error fetching coin data: ", error);
      } finally {
        setLoading(false);
      }
    };

    fetchCoinData();
  }, [coinId]);
```

```
const fetchHistoricalData = async (selectedDays) => {
  setLoading(true);
  const options = {
    method: "GET",
    headers: {
      accept: "application/json",
      "x-cg-demo-api-key": " ",
    },
  };
  try {
    const response = await fetch(
      `https://api.coingecko.com/api/v3/coins/${coinId}/market_chart?vs_currency=${currency.name}&days=${selectedDays}`,
      options,
    );
    const data = await response.json();
    setHistoricalData(data);
  } catch (error) {
    console.log("Some Error has occurred, " + error);
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchHistoricalData(days);
}, [coinId, currency, days]);
```

## QRcode Module:

```
const QRcode = () => {
  const [bitcoinAddress, setBitcoinAddress] = useState('');
  const [qrGenerated, setQrGenerated] = useState(false);
  const qrRef = useRef(null);

  const handleBitcoinAddressChange = (e) => {
    setBitcoinAddress(e.target.value);
  };

  const handleGenerateQR = () => {
    if (bitcoinAddress) {
      setQrGenerated(true);
    }
  };

  const handleDownloadQR = () => {
    if(qrRef.current){
      html2canvas(qrRef.current).then((canvas) => {
        const link = document.createElement('a');
        link.href = canvas.toDataURL('image/png');
        link.download = "qrcode.png";
        link.click();
      });
    }
  };
};
```

## Transactions Module:

(Integrated within Bitcoin.jsx)

```
const Bitcoin = () => {
  const token = ' ';
  const [address, setAddress] = useState('');
  const [transactions, setTransactions] = useState([]);
  const [btcPrice, setBtcPrice] = useState(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState('');
  const [lastTransaction, setLastTransaction] = useState();

  const { currency } = useContext(coinContext);

  useEffect(() => {
    fetchBtcPrice();
  }, [currency]);

  const fetchBtcPrice = async () => {
    const options = {
      method: "GET",
      headers: {
        accept: "application/json",
        "x-cg-demo-api-key": ' ',
      },
    };
    try {
      const response = await fetch(
        `https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=${currency.name}`,
        options,
      );
      const data = await response.json();
      setBtcPrice(data.bitcoin[currency.name.toLowerCase()]);
    } catch (error) {
      console.error("Error fetching Bitcoin price:", error);
    }
  };
};
```

```

const fetchTransactions = async (startTx = "") => {
  setLoading(true);
  setError(null);

  const baseUrl = `https://api.blockcypher.com/v1/btc/main/addrs/${address}/full`;
  const url = startTx
    ? `${baseUrl}?before=${startTx}&token=${token}`
    : `${baseUrl}?token=${token}`;

  try {
    const response = await fetch(url);
    const data = await response.json();

    if (data.txs) {
      setTransactions((prev) => [...prev, ...data.txs]);
      setLastTransaction(data.txs[data.txs.length - 1].hash);
    }
  } catch (error) {
    setError("Error fetching transactions:", error);
  } finally {
    setLoading(false);
  }
};

```

```

const totalBitcoinSpent =
  transactions.reduce((sum, tx) => {
    return (
      sum + tx.outputs.reduce((txSum, output) => txSum + output.value, 0)
    );
  }, 0) / 1e8;







const totalCurrencySpent = btcPrice ? totalBitcoinSpent * btcPrice : 0;

const API_URL = "https://api.blockcypher.com/v1/btc/main/addrs/";
let balanceForCSV = 0;
async function fetchBalanceForCSV() {
  try {
    const response = await fetch(
      `${API_URL}${address}/balance?token=`,
    );
    const data = await response.json();
    if (data.error) {
      balanceForCSV = data.error;
    } else {
      balanceForCSV = data.balance / 1e8;
    }
    return balanceForCSV;
  } catch (error) {
    console.log(error);
  }
}

```

```
const csvData = transactions.map((tx) => {
  const amountBTC =
    tx.outputs.reduce((sum, output) => sum + output.value, 0) / 1e8;
  const amountCurrency = btcPrice ? amountBTC * btcPrice : 0;
  return {
    Hash: tx.hash,
    Amount_BTC: amountBTC.toFixed(8),
    Amount_Currency: amountCurrency.toFixed(3),
    Currency: currency.name.toUpperCase(),
    Time: new Date(tx.received).toLocaleDateString("en-GB"),
    Confirmations: tx.confirmations,
  };
});
```


List all coins:


#	Coins	Price	24H Change	Market Cap
1	 Bitcoin - btc	\$ 87,798	-0.5%	\$ 1,742,589,754,634
2	 Ethereum - eth	\$ 2,066.6	-0.91%	\$ 249,441,204,649
3	 Tether - usdt	\$ 1	0.01%	\$ 143,803,473,146
4	 XRP - xrp	\$ 2.46	-0.42%	\$ 142,909,926,642
5	 BNB - bnb	\$ 631.02	-0.83%	\$ 92,098,567,784
6	 Solana - sol	\$ 145.1	2.42%	\$ 74,178,663,725


Search coins with matching search text:


bitcoin




Search










#	Coins	Price	24H Change	Market Cap
1	 Bitcoin - btc	\$ 87,798	-0.5%	\$ 1,742,589,754,634
12	 Wrapped Bitcoin - wbtc	\$ 87,605	-0.9%	\$ 11,299,982,606
25	 Bitcoin Cash - bch	\$ 337.56	1.28%	\$ 6,703,737,004

Coin info with chart (upon clicking any coin in the list on home page):

# Bitcoin (BTC)

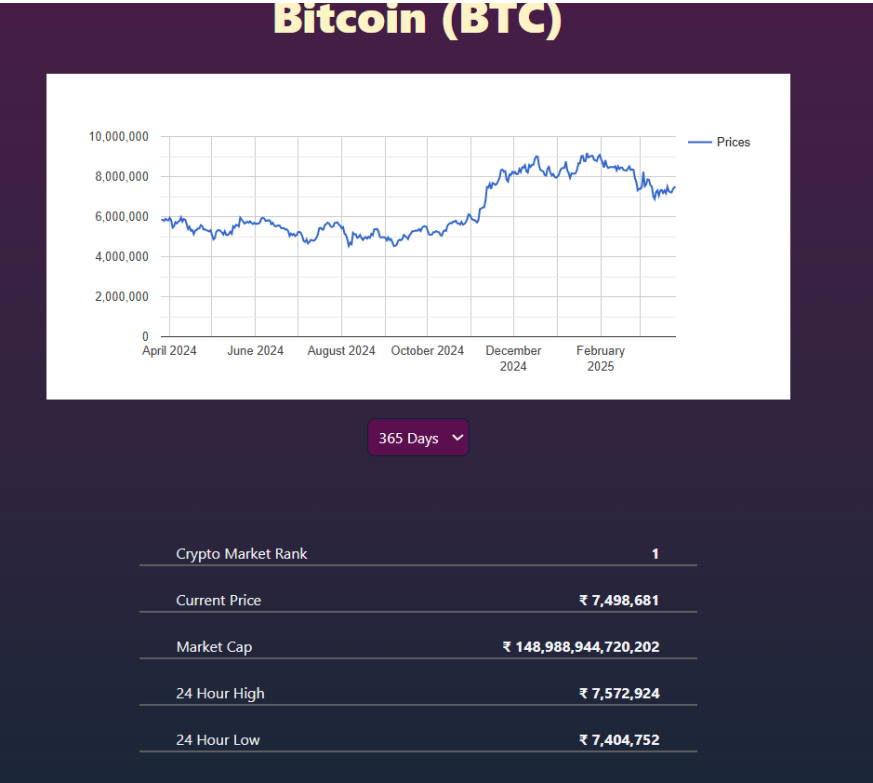


365 Days

Crypto Market Rank	1
Current Price	\$ 87,781
Market Cap	\$ 1,742,589,754,634
24 Hour High	\$ 88,510
24 Hour Low	\$ 86,358



Coin information in different currency:



Fetching transaction of given address:

Fetches 10 latest transactions at a time.

34xp4vRoCGJym3xR7yCVPFHocNxv4Twseo

Fetch Transactions

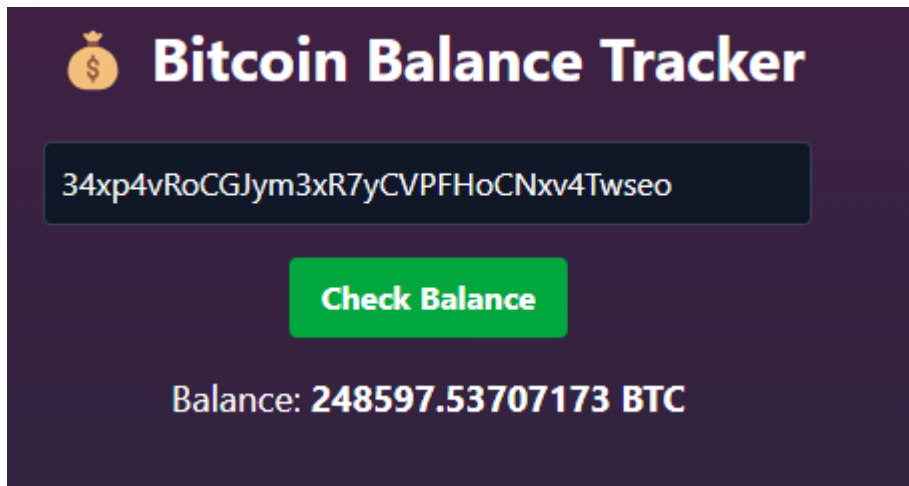
Transactions:

Transaction Hash	Amount (BTC)	Amount (USD)	Time	Confirmations
<a href="#">682bfd905b48a7e0925ed6fb...</a>	0.00201374	176.686 USD	23/03/2025	326
<a href="#">b288e0bc75df3d21c96b3f1a0...</a>	0.00048149	42.246 USD	21/03/2025	672
<a href="#">ca5254a53017e7bfd5079b3b2...</a>	67.77902695	5,946,931.825 USD	20/03/2025	850
<a href="#">841c49c277e15931bbda9b5c1...</a>	0.00001831	1.607 USD	17/03/2025	1324
<a href="#">596c1e44da755b5cae29c5f0d...</a>	0.00001638	1.437 USD	16/03/2025	1370
<a href="#">231f2bb24205caea9413d2cef...</a>	0.00003418	2.999 USD	16/03/2025	1396
<a href="#">7c81a56023a0683a6415a3c6c...</a>	0.00008202	7.196 USD	16/03/2025	1396
<a href="#">d6a0a84f43e5dd35af79aee99...</a>	0.00005278	4.631 USD	16/03/2025	1398
<a href="#">0e10fc1cc2288bb2df6dc15a2...</a>	0.00009680	8.493 USD	16/03/2025	1422
<a href="#">7833f503862911ed245f70a45...</a>	0.00006013	5.276 USD	11/03/2025	2099
Total Bitcoin Spent:		67.781882780 BTC	5,947,182.395 USD	

Export to CSV

Load More Transactions

Fetch balance:

A screenshot of a web application titled "Bitcoin Balance Tracker". It features a dark purple background. At the top left is a gold coin icon with a dollar sign. The title "Bitcoin Balance Tracker" is in large, bold, white text. Below the title is a dark blue rectangular box containing the Bitcoin address "34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo" in white text. Underneath this box is a bright green button with the text "Check Balance" in white. At the bottom, the text "Balance: 248597.53707173 BTC" is displayed in white.

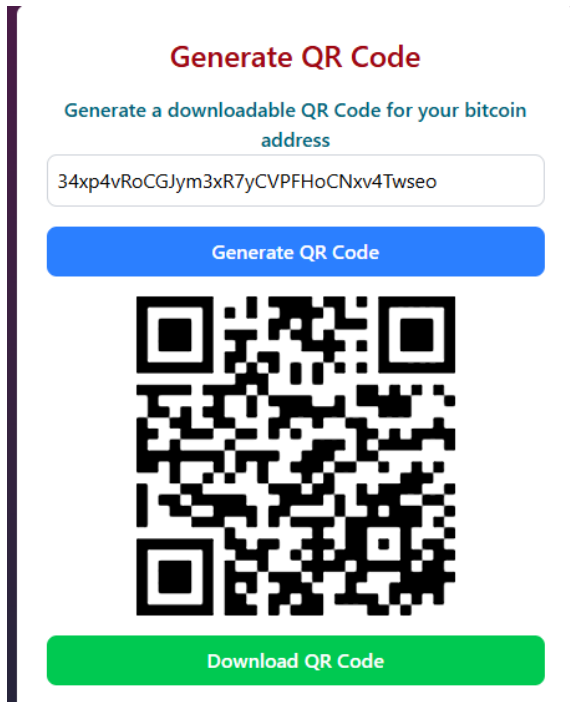
**Bitcoin Balance Tracker**

34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo

**Check Balance**

Balance: **248597.53707173 BTC**

Generate QR Code for given address


A screenshot of a web application titled "Generate QR Code". It has a light gray background. The title is in red text. Below it is a blue subtitle: "Generate a downloadable QR Code for your bitcoin address". There is a white input box containing the Bitcoin address "34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo". Below the input box is a blue button labeled "Generate QR Code". Underneath the button is a large QR code. At the bottom is a green button labeled "Download QR Code".

**Generate QR Code**

Generate a downloadable QR Code for your bitcoin address

34xp4vRoCGJym3xR7yCVPFHoCNxv4Twseo

**Generate QR Code**



**Download QR Code**

## Conclusion

In conclusion, the website developed is a comprehensive and user-friendly platform that provides an intuitive interface for interacting with the cryptocurrency world. By integrating real-time data from various sources and offering essential features such as cryptocurrency price tracking, transaction history, balance tracking, and QR code generation, the platform serves as a valuable tool for anyone involved in the world of digital assets.

This website offers users an in-depth look at the cryptocurrency ecosystem, empowering them to make informed decisions based on up-to-date market data and the status of Bitcoin

transactions. With functionalities such as multi-currency support, transaction exporting, and easy-to-use interfaces for tracking Bitcoin balances, the product bridges the gap between complex blockchain technology and user accessibility.

Blockchain technology, at the heart of cryptocurrencies like Bitcoin, offers secure, decentralized, and transparent transactions. This website is a glimpse into the potential of blockchain applications beyond just financial transactions. It showcases how blockchain technology can be leveraged to provide transparency, security, and accessibility for users, fostering a deeper understanding and adoption of decentralized technologies.

As cryptocurrencies and blockchain continue to evolve, this platform represents just one of many ways these technologies can enhance the user experience and offer new opportunities for financial interaction. It underscores the growing relevance of blockchain as a transformative force in the digital world, and with further advancements, similar platforms could expand their scope, enabling even more innovative applications in finance, data security, and beyond.