Student Name: Yazhen Han NUid:002950305

github username:ashisland11

Project Overview

The website showcases the "Latest Blocks" feature of a blockchain explorer interface that I developed. This feature is designed to present users with real-time data regarding the most recently mined blocks within a particular blockchain network.

Frontend Development

On the frontend, the interface is built using React.js, which allows for a dynamic and responsive user experience. React's component-based architecture is leveraged to create a table view that updates with new blocks as they are added to the blockchain. The styling and design elements are crafted using Ant Design, a comprehensive React UI library that provides a range of design components and tools to enhance the aesthetics and usability of the interface.

Backend Development

The backend is powered by Node.js, with Express.js serving as the web application framework. This combination allows for efficient handling of requests and seamless delivery of blockchain data to the frontend. The backend is responsible for interfacing with the blockchain, retrieving block data such as height, timestamp, the number of transactions, size, and weight, and then formatting this data to be sent to the frontend.

For the data source, the application taps into the rich data provided by the Bitquery API, a powerful tool that offers comprehensive blockchain data. Bitquery's API facilitates access to information across multiple blockchains, delivering a plethora of data points necessary for our "Latest Blocks" feature.

To query this data effectively, the backend utilizes GraphQL, a query language designed for APIs that provides a complete and understandable description of the data in the API. This allows clients to request exactly what they need and nothing more. The backend is responsible for crafting GraphQL queries that specifically request block heights, timestamps, transaction counts, sizes, and weights from the Bitquery API.

The use of GraphQL introduces efficiency in data retrieval, as it enables the backend to make precise data requests, avoiding over-fetching or under-fetching of data. This targeted approach to data retrieval means that the payload sent to the frontend is optimized, containing only the necessary block information required to render the "Latest Blocks" table.

The Node.js environment facilitates the execution of these GraphQL queries to the Bitquery API. The Express.js framework handles the incoming requests from the frontend, orchestrates the GraphQL queries, processes the API's responses, and formats the data to be suitable for frontend consumption.

The backend is also designed with error handling to manage any issues with the Bitquery API, ensuring that the frontend can gracefully handle any discrepancies or unavailability of data.

Data Flow and API Integration

Integration between the frontend and backend is achieved through a well-defined API, designed to provide the frontend with the necessary block data. This API is built using Express.js and is set up to handle asynchronous calls, ensuring that the frontend can receive updated data without reloading the page, providing a smooth user experience.

Engineering Challenges

The primary engineering challenge involved ensuring that the data flow between the backend and frontend was efficient and real-time. This required setting up WebSocket connections or polling mechanisms to push the latest block data to the frontend as soon as it became available. Additionally, careful state management in React was essential to ensure that the UI updates were performed optimally without unnecessary re-renders, which could lead to performance issues.