Building a Real-Time Ethereum Price and Gas Tracker USER GUIDE

author : Dzoan Steven tran email : Doansteventran@gmail.com

Goal

Building a Real-Time Ethereum Price and Gas Tracker This guide provides step-by-step instructions on how to build the application, encompassing both backend and frontend development.

Prerequisites:

Prerequisite	Version	Purpose	Notes
Node.js	16.x or 18.x	Runtime environment for running JavaScript on the server	LTS versions are recommended for stability and support.
npm	8.x	Package manager for JavaScript	Comes bundled with Node.js.
Git	Latest	Version control system	Necessary for source code management and collaboration
Docker	20.10.x or newer	Platform for developing, shipping, and running applications	Ensures consistent environments across development stag
TypeScript	4.x	Superset of JavaScript providing type safety	Optional but recommended for large projects.
Webpack	5.x	Module bundler for JavaScript applications	Optional, depends on project requirements.
React	17.x or 18.x	JavaScript library for building user interfaces	Choose the version based on compatibility with other too
Next.js	12.x or 14.x	React framework for production	Use the latest stable release for new projects.
ESLint	8.x	Linter tool to find and fix problems in JavaScript code	Helps maintain code quality and consistency.
Babel	7.x	JavaScript compiler	Transforms ECMAScript 2015+ code into backward compa version for current and older browsers.
Jest	27.x or 29.x	JavaScript testing framework	Provides a complete testing

Project Setup:

1- Create a GitHub Repository: Create a new private repository on GitHub named assignment-bs. Add any necessary collaborators (e.g., @kennysliding and @dark5tarx). Clone the repository to your local machine using Git:

git clone https://github.com/your-username/assignment-bs.git

2- Project Structure: Create the following directory structure within the cloned repository:

```
assignment-bs/
  backend/
    src/
      server.ts
      // ... other backend files (if needed)
    package.json
    tsconfig.json
  frontend/
    pages/
      index.tsx
    // ... other frontend files and components (if needed)
    package.json
    tsconfig.json
  README.md
```

3.Install Dependencies: Navigate to the backend directory and install the required dependencies:

npm install next react react-dom socket.io-client

Backend Development: Create server.ts: In the backend/src directory, create a file named server.ts and add the following code:

```
// ... (import statements and interface definitions - as shown in previous examples)

// ... (logger setup - optional)

const app = express();
const PORT = process.env.PORT || 3000;
const server = app.listen(PORT, () => console.log(`Listening on port ${PORT}`));
const wss = new WebSocket.Server({ server });

// ... (currentData object for data storage)

async function fetchCryptoData() {
 // ... (fetch ETH price and gas price data)

 // ... (data extraction, validation, and update currentData)

 return currentData;
}
```

```
// ... (broadcastData function)

// Fetch and broadcast data initially and periodically
fetchCryptoData().then(broadcastData);
setInterval(() => {
 fetchCryptoData().then(broadcastData);
}, 10000); // Adjust update interval as needed

wss.on('connection', (socket: WebSocket) => {
 console.log('New client connected');
 socket.send(JSON.stringify(currentData));
});
```

Implement Data Fetching: Choose your preferred method for fetching gas price data (Etherscan Gas Tracker scraping or ETH Gas Station API). Implement the logic for fetching data from the chosen source and extracting the necessary values within the fetchCryptoData function. Consider adding data validation and error handling as discussed previously.

Frontend Development: Create pages/index.tsx: In the frontend/pages directory, create a file named index.tsx and add the following code:

```
// ... (import statements)

function Home() {
 const [ethPrice, setEthPrice] = useState(null);
 const [gasPrice, setGasPrice] = useState({ standard: null, fast: null, instant: null });

 useEffect(() => {
  const socket = io('http://localhost:3000');

  socket.on('update', (data) => {
   setEthPrice(data.ethPrice.usd);
   setGasPrice(data.gasPrice);
  });

  // ... (cleanup logic)
 }, []);

 return (

   {/* Design your UI to display ethPrice and gasPrice */}

 );
}
```

```
export default Home;
```

Implement WebSocket Connection:

Use socket.io-client or the built-in WebSocket API to establish a connection to your backend server. Handle Data and Update State: In the useEffect hook, set up a listener for incoming messages from the WebSocket. Parse the JSON data and update the state variables ethPrice and gasPrice with the received values. Design the UI: Create the UI elements to display the Ethereum price and gas prices for different speeds. Use HTML elements, CSS styling, and potentially additional React components to create a clear and user-friendly interface.

Testing and Running the Application:

Start the Backend Server: In the backend directory, run npm start or yarn start to start your Node.js server. Start the Frontend Development Server: In the frontend directory, run npm run dev or yarn dev to start the Next.js

development server. Access the Application: Open your browser and navigate to http://localhost:3001 (or the port where your Next.js server is running) to access the application. Verify Functionality: Ensure that the Ethereum price and gas prices are being displayed correctly and updated in real-time as the backend sends new data. Test the application thoroughly and address any issues you encounter. so this is it for a local deployment and it should gives you this

```
Back End
https://websocketking.com/
18:58 20.02 { "ethPrice": 2991.07, "gasPrice": { "LastBlock": "19782287", "SafeGasPrice": "8",
"ProposeGasPrice": "9", "FastGasPrice": "11", "suggestBaseFee": "7.994966246", "gasUsedRatio":
"0.395591633333333,0.505830233333333,0.654276033333333,0.703146766666667,0.3398607" } }
17:29 14.27 { "ethPrice": 2944.91, "gasPrice": { "LastBlock": "19781844", "SafeGasPrice": "7",
"ProposeGasPrice": "7", "FastGasPrice": "9", "suggestBaseFee": "6.930921316", "gasUsedRatio":
"0.5845133,0.4052553,0.0566691,0.999673733333333,0.644321333333333" } }
17:28 14.22 { "ethPrice": 2944.91, "gasPrice": { "LastBlock": "19781839", "SafeGasPrice": "7",
"ProposeGasPrice": "7", "FastGasPrice": "9", "suggestBaseFee": "6.708466761", "gasUsedRatio":
"0.539835633333333,0.367459766666667,0.999733,0.503076966666667,0.478914933333333" } }
```

Front End http://localhost:3001/

```
Crypto Prices
```

```
Ethereum Price: 2991.07 USD

Gas Prices:
Standard: 9 Gwei

Fast: 12 Gwei

Instant: 10 Gwei
```

Troubleshooting

You might run like me into this error , the FE would not show the GAS price . so check the Gas value front your Back end :

```
"ethPrice": 2944.91, "gasPrice": { "LastBlock": "19781844", "SafeGasPrice": "7", "ProposeGasPrice": "7", "FastGasPrice": "9", "suggestBaseFee": "6.930921316", "gasUsedRatio": "0.5845133,0.4052553,0.0566691,0.999673733333333,0.644321333333333" }
```

and make sure the front end components Index.js , App. js are using the sames Gas key values : Standard: {gasPrice.standard} Gwei Fast: {gasPrice.fast} Gwei Instant: {gasPrice.instant} Gwei and not loose 2 hours debugging like me !

8 Deployement of Docker user guide is in the Docker Folder

9 Deployment on AWS/Amplify/Cloudflare user guide in the Cloud Folder