Sperax React Frontend Test Documentation

Project Overview

This project is a React-based frontend that interacts with an Ethereum blockchain to retrieve the balance of a wallet and send tokens to a provided address. The project leverages Web3.js to interact with smart contracts and manage wallet transactions.

Prerequisites

Before running or modifying this project, ensure you have the following installed:

- Node.js (v20.x or later)
- npm
- MetaMask or another Ethereum-compatible wallet (for testing in the browser)

Installation

1. Clone the repository:
   ```
   git clone https://github.com/techpilot/sperax-test.git
   ```

2. Navigate to the project directory:
   ```
   cd sperax-test/sperax-client
   ```

3. Install the dependencies:
   ```
   npm install
   ```

4. Start the development server:
   ```
   npm start
   ```

   The application will be available at `http://localhost:3000`.

Project Structure

The project follows a standard React structure:

```
src/
├── components/
│   ├── GetWalletBalance.tsx
│   ├── Footer.tsx
│   ├── Header.tsx
│   ├── SendToken.tsx
│   └── SendTokenModal.tsx
├── contexts/
│   ├── ContractProvider.tsx
│   └── Web3Context.tsx
├── contracts/
│   └── DesToken.json
├── App.tsx
└── index.tsx
```

Main Components

1. **GetWalletBalance.tsx**

   - **Description**: This component retrieves and displays the balance of the provided wallet address in the form
   - **Methods**:
     - `handleGetBalance()`: Retrieves the balance of the provided wallet and updates the state on submitting the form.

2. **SendToken.tsx**

   - **Description**: This component allows the user to send tokens to a specified address.
   - **Methods**:
     - `handleSubmit()`: Accepts recipient's address and amount through a form and opens a confirmation modal when the button is clicked

3. **SendTokenModal.tsx**

   - **Description**: This component is a modal that displays the details of transaction that is about to be performed for confirmation by the user before initiating the sending of the token.
   - **Methods**:
     - `handleTokenTransfer()`: Initiates the sending of the token to the provided recipient's address.

Interacting with the Smart Contract

Setting Up Web3.js with Context API

In this project, the Web3.js setup and smart contract interaction are handled using the Context API to make states and functions globally available across the application.

1. **Web3Context.tsx**: This file will set up the Web3 context hook, the interface for the exported states and functions used globally across the application.
2. **ContractProvider.tsx**: This file will set up the Contract provider, the connect wallet method, the smart contract instance, and other related states

Workflow

1. **Get Wallet Balance**:
   - When the Get`WalletBalance` component mounts, and a valid ethereal address is provided in the form, it retrieves the balance of the provided wallet using the `handleGetBalance()` method.
2. **Send Tokens**:
   - The user inputs the recipient's address and the token amount to send.
   - On clicking the "Next" button, the `handleSubmit()` method is invoked, opening the `SendTokenModal` component.
   - On clicking the "Send" button, the `handleTokenTransfer()` method is invoked, performing the token transfer to the provided wallet using the connected wallet.

Deployment

To deploy the React application:

1. Build the application:
   ```
   npm run build
   ```

2. Deploy the `build` folder to your preferred hosting service (e.g., Vercel, Netlify, GitHub Pages).

Additional Resources

- React Documentation
- Web3.js Documentation
- MetaMask Documentation