Develop a Basic Read-Only Wallet

Created	@August 25, 2021 4:20 AM
Tags	Client Libs, Full Stack
Updated at	@April 11, 2023 10:25 AM

> 👉 A take-home assignment is a work sample where candidates are asked to complete a task that is intended to showcase their practical skills and thought process. These tasks are accompanied by a due-date but are completed on the candidate's own time.

Evmos is on a mission to develop and ship the foundational tools necessary for building the cross-chain applications of the future, freeing developers from the confines of today's siloed blockchains. Our teams are building for a world where the next million Web3 users are simply non-crypto native people. Read our latest manifesto here

Assessment

You have a week to complete the below assignment. The team will follow up on your progress and will ask you to send over your solution and do a live run-through of your program during a 30-45 min call. You will be asked questions on your implementation choices as well as what you would have done differently given more time.

Prerequisites

- Node and `npm` / `yarn` / `pnpm`: https://nodejs.org/en/download/
- Languages and Frameworks

    - Next.js (recommended), React, Nuxt or Vue
    - TypeScript or JavaScript. If you use JS, we suggest using type checker like Flow.
- Libraries to use:
    - EvmosJS
    - Cosmos JS: cosm.js
    - Ethereum: ethers.js/web3.js
- GitHub account
- Coingecko API for the tokens prices

Instructions

For this assignment, you will:

1. Create an open-source (i.e public) GitHub repository to host your project
2. Create a read-only wallet as a web application that is able to display **all** user balances
3. Connect to a testnet (or mainnet) node using JSON-RPC (for ERC20 tokens) and gRPC/REST (for Cosmos coins). See the list of public available nodes that you can use to connect here.
    a. If you need tokens for your project, you can use the testnet faucet.

> 💡 Note: you'll need to use Port

4. Implement a UI in Next.js, React (or Vue / Nuxt) that reads the blockchain and token balances (ERC20 and Cosmos coins) for a given address
    a. The UI should satisfy the following **User Stories**:
        - As a wallet user, I want to be able to see my address in both Hex and Bech32 formats, so that I can use account easily with the available Ethereum and Cosmos tools and apps.

- As a wallet user, I want to see the balance from all my assets on Evmos: Cosmos (IBC) and Ethereum (ERC20s)
- As a wallet user, I want to see the value of fiat of each of the assets and see the total balance

b. The UI should contain the following views or pages:

- **Home**: Welcome page
- **Input address field**: To be able to display the balances relevant to that address
- **Balance**: Total balance of the user
  - Display Cosmos coins (native from the own chain and IBC coins) and Ethereum ERC20 tokens
  - Display the fiat (usd or eur) value for all the coins and tokens
  - Display the total balance in fiat (usd or eur)

5. Document your program with inline comments and a `README.md` file on the base directory of the GitHub repo.
6. (Bonus/Optional) Set up CI/CD pipelines in GitHub actions for Linter, unit tests, etc.
7. (Bonus/Optional) List the transaction linked to the user
   a. User Story: As a wallet user, I want to see all my own transactions (sent and received), so that I can see all my interactions with other addresses and contracts

Criteria

While demonstrating proficiency in React or Vue (Js / Typescript) is necessary, you may use any additional frameworks and libraries provided that you build a solid program with an emphasis on code quality, simplicity, readability, maintainability, and reliability, particularly regarding testing.

Be aware that the team, besides the items mentioned above, will mainly take into consideration the following evaluation criteria:

- How clean, organized and maintainable your code is (linting tools, type checking, good practices, etc)
- Design and UX of the web application
- How clear is the documentation is (eg. inline comments, function parameters, `README` instructions)

Other important notes:

- Add to the `README` file:

  (1) instructions to build, install necessary dependencies and run the code;

  (2) what were the main technical and design decisions you made and why you made them. If you used any references or guides, please list them at the bottom of the document;

  (3) relevant comments about your project and how each of the steps were performed;

  (4) challenges you faced during the assignment and how you solved them;

- You **must** use English in your code and also in your docs

This assignment should be doable in less than one week. We expect you to learn fast, **communicate with us**, and make decisions regarding its implementation & scope to achieve the expected results on time. If you are blocked for some reason, feel free to reach out to us:

Federico (Co-founder): federico@evmos.org

Sandoche (Full stack manager): sandoche@evmos.org

Consider another developer would get your project/repository to evolve and implement new features from exactly where you stopped.

Best,

Evmos Team 🌋