EigenLayer Marketplace Take Home Project

Senior Fullstack Engineer:

OVERVIEW

First off, **congratulations** on advancing to the project stage for the Senior Fullstack Engineer position here at EigenLayer! We are very excited that you are part of this process and are looking forward to seeing and discussing your project work.

This project is a way for us to:

1. gauge your interest in the position
2. assess your design, technical abilities and thought process related to this role
3. see how much you can get up to speed on the EigenLayer ecosystem

DESCRIPTION

The Marketplace team at EigenLayer is responsible for building out the apps for three core user personas: Stakers, Operators and AVSs. These applications are built using both traditional web2 development constructs like microservices, frontends and databases but also use web3 functionality like interactions with the blockchain/smart contracts, subgraphs, and wallets.

The goal of this project is to build a simple webapp that consumes an existing subgraph and presents some of its data in interesting and potentially useful ways. The actual data and how you wish to present it is completely up to you.

Consider this project as part of your EigenEducation. You'll become quite familiar with similar data and terminology in a full-time role.

Also consider what can be accomplished in just a few hours. This is not meant to be an exhaustive project. If something doesn't work after many attempts, document it and move on.

Design considerations

Frontend
Feel free to use an existing dashboard or frontend template/framework to build out your frontend. This is not an exercise in the look and feel of a dashboard, grid or webapp but rather what you learn and prioritize to present to us.

Subgraph
We've already implemented a subgraph for your use in this project. Feel free to implement a subgraph as well but that is not a requirement due to time constraints.

Alchemy API Subgraph Playground (w/ Schema)
https://subgraph.satsuma-prod.com/eigenlabs/eigen-graph-testnet-holesky/playground

GraphQL API Endpoint (HTTP)
https://subgraph.satsuma-prod.com/027e731a6242/eigenlabs/eigen-graph-testnet-holesky/api

Backend
It may be necessary to enrich the data from the subgraph in various ways before it's retrieved by the client/frontend. In this case, we highly suggest adding a simple backend built in NestJS to your solution.

Data
Several examples of interesting data:

- Beacon chain ETH deposited by block time
- Searchable table of EigenPods by owner or validator
- Paginated table of sortable Strategies
- List of deposits and/or withdrawals by address, filterable by token/amount/date
- Operator, Operator Set and AVS data (depending on data availability)

Feel free to review but ignore more lofty or concepts like: Checkpoints, Proofs, ThirdPartyTransfers and items not called out in our documentation.

REQUIREMENTS

We've set a few technical requirements but ideally you come to us with your opinions and any best practices.

- Use Typescript along with a major frontend framework like React or Vue.
- If you build any backend code, it's suggested to use the NestJS framework.
- A single Github monorepo containing your FE and any BE, subgraphs, docs, tests, etc.
- The rest is up to you. If you wish, you can run design or data considerations past us.

DELIVERABLES

Your project deliverables will include:

- A functioning Github repository containing your work.
    - Please consider sending this to us around 4 hours prior to your scheduled presentation.
    - Include instructions on how to get up and running quickly with your project.
- One hour presentation related to your work and findings.
    - (Optional) Slides and formal presentation
    - Additional questions to ponder:
        - What constraints might you encounter in bringing your webapp to the masses?

    - If you had more time, where would you take your dashboard?
    - What tradeoffs did you consider with the data you're presenting?
    - What did you identify in the data that you found interesting?
    - Where might a true backend API be useful over subgraphs?
    - What data or functionality is missing from the subgraph?

Feel free to research and implement additional functionality or a completely different architecture. Simply back up your ideas with a solid rationale. **Thank you for your time**.

RESOURCES

- EigenLayer
    - Eigenlayer Marketplace App
    - Documentation
    - Terms
    - Contract Addresses
    - Contract Source Code
- GraphQL
- TypeScript
- NestJS