TunnelVision Technical Interview

Below is my write up on design considerations as well as challenges for the technical assessment for both the frontend and backend stacks.

TunnelVision Write Up - Backend Considerations & Challenges

My first design consideration was the database I was going to use for the backend. Several of my recent clients have used MongoDB, and due to Mongo's easy use on node stacks, I decided to go ahead and use that. Next, I had to determine which Web3 provider I was using. I decided to use Alchemy, as I have used them in recent projects as well. Next, I had to decide which RESTful service I was going to use. I chose express, due to my familiarity with it.

NOTE: I did commit my .env files to the github since this is for interviewing purposes. I would never commit these files for an application that would be used in production, for obvious security reasons.

I wanted to make sure I was efficiently using the MongoDB database to avoid querying the Web3 provider I selected too often. My design pattern therefore became as follows:

1. Check to see if a record exists within MongoDB.
    a. If the record exists, return that record.
    b. If the record does not exist, query Alchemy and add the record for later retrieval.

Additionally, I had to add refresh timeouts to my Web3 provider calls in order to avoid overloading the provider. This brings to light the benefits of potentially using my own Web3 provider, but I didn't want to get too caught up in the weeds with having my own node running for this application. Instead, I set a delay of 150 ms per Web3 call to prevent any overloading issues.

Finally, I wanted to set up basic authentication for this app for those bonus points! I used a simple bearer token here with a client secret, and used jwt to sign the client secret with the IP of the requester. In the future, I would definitely tie this to some sort've email verification system or login system for more persistent API key usage. I enjoyed designing this backend, and made sure to document my functions as well so you may check how they work.

TunnelVision Write Up - Frontend Considerations & Challenges

I'll admit, my frontend skills are usually lesser than my backend ones. I find it very easy to display functional components and hook them up to the backend logically, but getting the displays to work the way I like is somewhat frustrating for me. That being said, I feel good about the state of the frontend for this app.

I used react's Router to set the different routes of the app (/blocks/XXXX for example will display block XXXX, and /tx/TXHASH will display tx TXHASH). Additionally, I had to think about how these different routes would query the backend. The /blocks/ route was fairly simple as I didn't include much supporting information beyond that of what was returned from the `/getBlock` endpoint, however /tx/ was a bit more work.

I wanted to make sure that my transactions were properly displaying things like token transfers, which required me to query the transaction receipts of the transactions as well as the raw transactions themselves. This allowed me to get logs of transfers as well as gas prices for the tx.

Overall, I still prefer to sit more toward the backend on these projects. However I feel confident that the functional components of this app are on display here.