Backend Web3 Integration Engineer (senior-level)

Coding Challenge

Hi, thank you for being part of our journey so far! Needless to say, we are a fast-growing company, with lots of opportunities, and if you're still with us so far, it means that we definitely see something special in you!

This is just a task for us to learn about your thought process, your working style, and the way you express your ideas.

This challenge is quite broad and open-ended. Do not worry! Although it would be awesome if you attempted all of them, we do not expect everyone to be experts in each and every field. This is more of a way for us to gauge your interests, and find a role that fits you best. If you have a better way of presenting yourself, we are open to hearing about it too, and change things around.

Part A: Web2 Backend

Task 1: Simple Backend Endpoint

Create a simple Python Django app that has the following features:

- a backend API endpoint called `get_score` with a simple dummy formula that could be anything e.g. `result = input + 1`
- a PostgreSQL database that the backend API uses to log the user ID and the score
- demonstrate that the endpoint is behaving as expected (how do we test it? how can we prove that it is working as expected?)

Task 2: Admin Panel

Create a simple admin panel where operations staff can use to manage the database:

- a non engineer should be able to view the SQL tables, and search/make queries
- even better if the staff can edit the entries too

Task 3: Database Migration

If we want to change the schema of the database, say we want to add, edit, or remove columns:

- demonstrate a process or script, or a demo video of the schema update process
- demonstrate that the endpoints and the services are not affected, and the tests are running fine

Part B: Web3

Web3 Integration: Reading

Our mobile app will allow users to interact with the web3 ecosystem, for the purpose of this challenge, you can use any EVM-compatible tools. Create functionalities and flows for the following use cases:

- create a functionality to verify that a user owns a wallet address
- allow users to connect their wallet and then display their token balances, and NFTs (even better if you can even retrieve the metadata/traits and media content of the NFTs)
- add filter functionality to only display NFTs from a list of contracts (for example, you can only choose to display Axie Infinity and Genopets NFTs)
- play to earn backend, that is, allowing the minting of tokens (when player wins) and burning of tokens (when player uses tokens to buy in-game items, for example)

S**mart Contracts**

There are a few smart contracts that we will be building, for these, you may use EVM standards such as ERC20, ERC721, ERC721A etc, we would like to hear from you on why you chose particular standards vs others! If you do not code smart contracts, let us know, or try to use existing smart contracts out there.

- create an ERC20 token smart contract, call it your name or a unique id such as $JOHN123
- create a staking smart contract, that is able to accept deposits of tokens and NFTs
- create a marketplace smart contract, that is able to allow users to trade and swap tokens and NFTs
- create a bridge contract
- deploy the smart contracts, write tests, and demonstrate that they work on testnet or mainnet

Web3 Integration: Writing

- backend functionality to allow users to transfer their NFT from their connected wallet, to a staking smart contract (aka “Omnibus” / “Vault”) to earn tokens
- backend functionality to allow users to trade their NFTs on the marketplace
- backend functionality to interact with a bridge contract (if available)

Part C: Infrastructure and Devops

Deployment to cloud

Demonstrate the tasks above can be deployed end to end on cloud (preferably on AWS)

- dockerise the app
- set up CI/CD pipelines or automated unit testing
- host the app on serverless hosting on AWS e.g. container / cluster hosting services
- connect to a database that is hosted on the cloud

Bonus Task: Data Pipeline

We are also creating data pipelines, for storing audio recordings, and also other wearable device data e.g. Fitbit, Oura Ring, Apple Healthkit

- create a data processing pipeline (ETL pipeline) where data stored in a cloud DataLake (e.g. s3 bucket) is processed by an ETL pipeline (e.g. PySpark) and finally ingested into a Data Warehouse (e.g. Amazon Redshift)
- even better if the entire pipeline is deployed to the cloud, and has a serverless, autoscaling architecture

Bonus Task: Infrastructure optimisation

Implement infrastructure features to the whole backend stack

- cybersecurity measures
- or some kind of monitoring tool / alert / disaster recovery method
- proxy servers, firewalls, load-balancers etc

Bonus Task: Open-ended!

If you have specific strengths in certain areas, feel free to demonstrate it. We are a company that incorporates games, data, and web3. There of opportunities to expand upon your work!

Task:

Prepare a presentation of your code and ideas.

Duration:

1 week (but do let us know if you need any extensions)

Submission

1. Please prepare a short presentation (PDF, powerpoint, google slides etc) outlining your solution.
2. Create a public GitHub repository and push your code there.
3. Please also include your public GitHub repository link, resume, phone number and preferred email address in the email.
4. Email your materials to:

james@pacer.gg

henry@pacer.gg

brian@pacer.gg

john@pacer.gg

apo@pacer.gg

After that, we will be scheduling a short meeting to allow you to present your idea to us, and we're good to go!

Again, thank you for participating, we're all gonna make it.