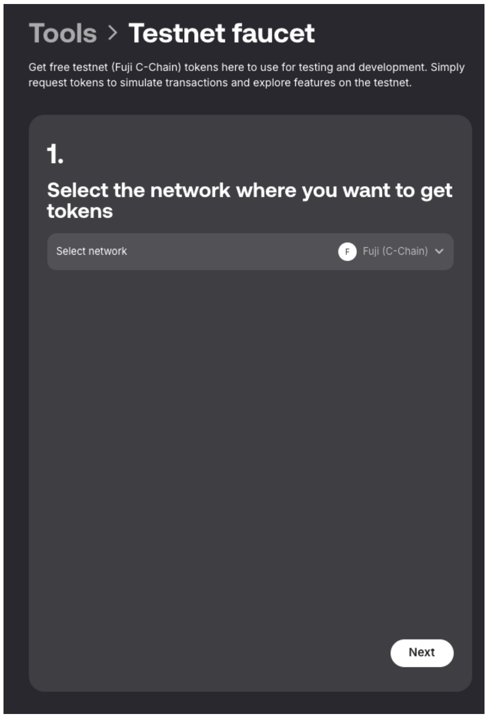
Take-Home Technical Interview Test: Web3 "Send Token" Form

Objective:

Develop a web application form that allows users to send a USDC ERC-20 token on Avalanche Fuji C-Chain (testnet) to another address. The application should interact with a blockchain to facilitate the transfer of tokens.

Before you get started:

- Aim to complete this assignment in less than 4 hours. Don’t worry if you’re unable to complete everything. Just get as far as you can.
- Wallet Setup:
    a. You may use whichever wallet provider you prefer, but if you need a wallet you can install [Core Extension]!
    b. Add Avalanche Fuji C-Chain network, add the USDC asset, and acquire USDC in your testnet account, all via our [Testnet Faucet]:



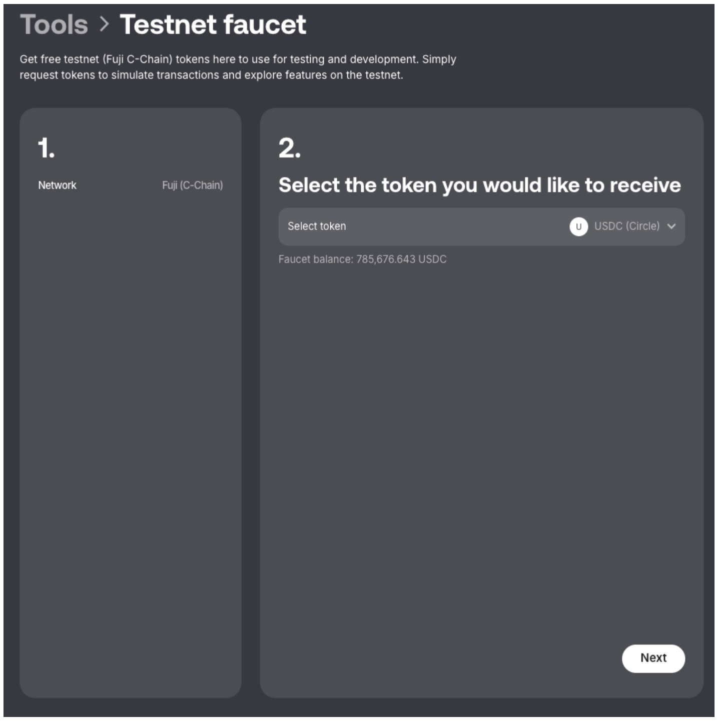
Tools > Testnet faucet

Get free testnet (Fuji C-Chain) tokens here to use for testing and development. Simply request tokens to simulate transactions and explore features on the testnet.

1\.

Select the network where you want to get tokens

Select network: F Fuji (C-Chain)



Tools > Testnet faucet

Get free testnet (Fuji C-Chain) tokens here to use for testing and development. Simply request tokens to simulate transactions and explore features on the testnet.

1\.

Network: Fuji (C-Chain)

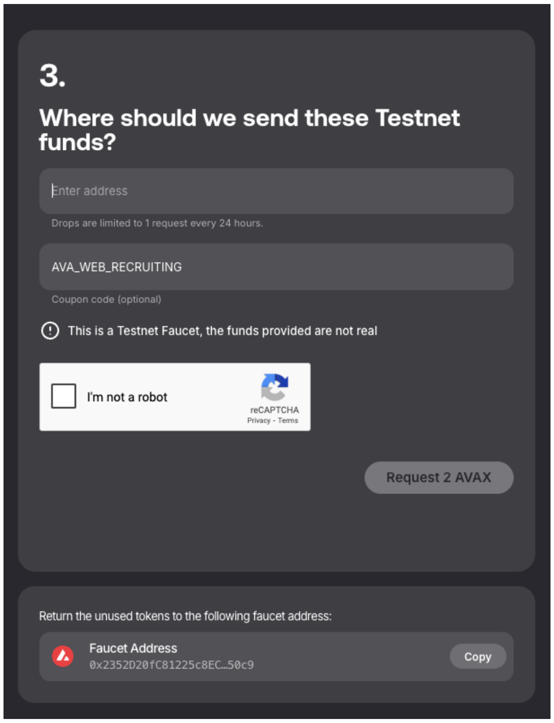
2\.

Select the token you would like to receive

Select token: U USDC (Circle)

Faucet balance: 785,676.643 USDC

c. Also, grab some AVAX from the faucet! You'll need it to pay for transaction fees. Use the Coupon Code: **`AVA_WEB_RECRUITING`**



3.

Where should we send these Testnet funds?

Enter address

Drops are limited to 1 request every 24 hours.

AVA_WEB_RECRUITING

Coupon code (optional)

This is a Testnet Faucet, the funds provided are not real

Return the unused tokens to the following faucet address:

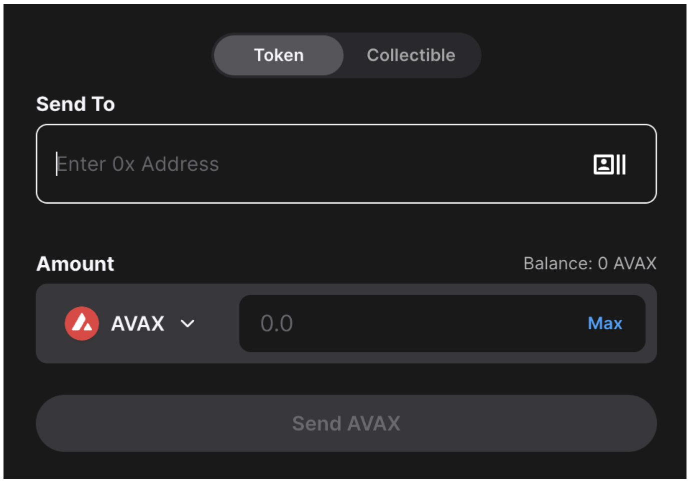
Faucet Address
0x2352D20fC81225c8EC…50c9

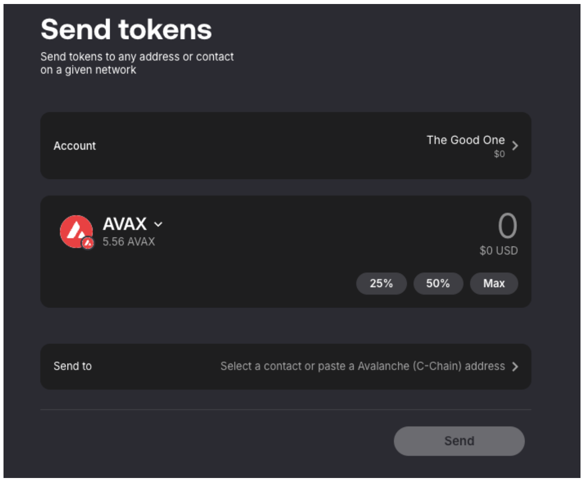
- Here is the USDC contract information on C-Chain (the token received from the testnet faucet):

```JavaScript
      {
        "address": "0x5425890298aed601595a70AB815c96711a31Bc65",
        "name": "USD Coin",
        "ercType": "ERC-20",
        "symbol": "USDC",
        "decimals": 6,
      }
```

Style Guide

You can use the Core Web "Send" tool's UI as an example. This is just to help provide guidance so please feel free to take liberties and deviate from Core's Send UI/UX.





Send tokens
Send tokens to any address or contact on a given network
Account
The Good One
$0
AVAX
5.56 AVAX
0
$0 USD
25%
50%
Max
Send to
Select a contact or paste a Avalanche (C-Chain) address
Send

Requirements:

1. Front-End Development:
   - Use React and TypeScript for the frontend
   - The form should have fields for:
     i. Recipient's wallet address
     ii. Amount of USDC to send
   - Implement form validation (e.g., address format, balance checks).
   - The UI should be responsive and user-friendly.
2. Blockchain Interaction:
   - Use the wagmi and viem Web3 libraries to connect to your wallet and interact with the blockchain.
     i. See https://wagmi.sh/core/chains to quickly integrate with Avalanche or AvalancheFuji
   - Implement logic to:
     i. Get the current balance of the sender's wallet.
     ii. Send tokens to the specified recipient address.
   - Handle errors and provide feedback to the user (e.g., insufficient balance, incorrect address format).
3. Testing:
   - Write unit tests for your code.
   - Document how to run the tests.
4. Documentation and Code Quality:
   - The code should be well-documented and easy to read.
   - Follow best practices for code structure and design patterns.
5. Bonus (Optional):
   - Add transaction status updates via blockchain event listening.

Evaluation Criteria:

- Functionality: Does the application meet the outlined requirements?
- Code Quality: Is the code well-organized, efficient, and maintainable?
- User Experience: Is the form intuitive and easy to use?
- Technical Skills: Proficiency in front-end development, blockchain concepts, and Web3 integration.

Submission Guidelines:

When turning in the assignment please do so through Stackblitz for the code portion. If the application isn't running inside of Stackblitz for whatever reason feel free to host the application on a server of your choice (Vercel, Netlify, etc.).

Please keep the Stackblitz and any reference to the material you create private so that future engineers aren't able to use what you produce for their own interview.

We **DO NOT** accept the code ourselves for security reasons so as such this will be a hard requirement.

Include a README file with a brief explanation of the approach and technologies used.

Deadline for submission: 24 hours after receiving this assignment.

Thanks for taking the time to complete the homework and we look forward to your submission!