

Hàm buy và sell trong vendor contract

```
contracts > Vendor.sol > Vendor
7  contract Vendor is Ownable {
18
19      function buyTokens() public payable {
20          require(msg.value > 0, "Send ETH to buy tokens");
21          uint256 tokensToTransfer = msg.value * tokensPerEth;
22          uint256 vendorBalance = yourToken.balanceOf(address(this));
23          require(vendorBalance >= tokensToTransfer, "Vendor has insufficient tokens");
24          yourToken.transfer(msg.sender, tokensToTransfer);
25          emit BuyTokens(msg.sender, msg.value, tokensToTransfer);
26      }
27
28      function sellTokens(uint256 tokenAmount) public {
29          require(tokenAmount > 0, "Specify an amount of tokens to sell");
30          uint256 ethToTransfer = tokenAmount * tokenPrice / 1 ether;
31          uint256 vendorETHBalance = address(this).balance;
32          require(vendorETHBalance >= ethToTransfer, "Vendor has insufficient ETH");
33          yourToken.transferFrom(msg.sender, address(this), tokenAmount);
34          payable(msg.sender).transfer(ethToTransfer);
35          emit SellTokens(msg.sender, tokenAmount, ethToTransfer);
36      }
37
38      function withdraw() public onlyOwner {
39          uint256 balance = address(this).balance;
40          require(balance > 0, "No ETH to withdraw");
41      }
42  }
```

file test buy and sell token

```
test > JS Vendor.test.js > ...
4  describe("Token Vendor", function () {
30      describe("Deployment", function () {
43          });
44      });
45      describe("Buy Tokens", function () {
46          it("Should buy tokens correctly", async function () {
47              const ethAmount = ethers.parseEther("1");
48              const expectedTokens = ethAmount * 100n;
49
50              await expect(() =>
51                  vendor.connect(user1).buyTokens({ value: ethAmount })
52              ).to.changeTokenBalance(yourToken, user1, expectedTokens);
53          });
54
55          it("Should emit BuyTokens event", async function () {
56              const ethAmount = ethers.parseEther("1");
57              await expect(vendor.connect(user1).buyTokens({ value: ethAmount }))
58                  .to.emit(vendor, "BuyTokens")
59                  .withArgs(user1.address, ethAmount, ethAmount * 100n);
60          });
61
62          it("Should fail if vendor has insufficient tokens", async function () {
63              const largeAmount = ethers.parseEther("1001"); // More than vendor has
64              await expect(
65                  vendor.connect(user1).buyTokens({ value: largeAmount })
66              ).to.be.revertedWith("Vendor has insufficient tokens");
67          });
68
69          it("Should fail with zero ETH sent", async function () {
70              await expect(
71                  vendor.connect(user1).buyTokens({ value: 0 })
72              ).to.be.revertedWith("Send ETH to buy tokens");
73          });
74      });
75
76      describe("Sell Tokens", function () {
77          beforeEach(async function () {
78              // ...
79          });
80      });
81  });
```

```

it("Should sell tokens correctly", async function () {
  const tokenAmount = ethers.parseEther("100");
  const expectedEth = tokenAmount / 100n; // 100 tokens = 1 ETH

  // Approve vendor to spend tokens
  await yourToken.connect(user1).approve(
    await vendor.getAddress(),
    tokenAmount
  );

  await expect(() =>
    vendor.connect(user1).sellTokens(tokenAmount)
  ).to.changeEtherBalance(user1, expectedEth);
});

it("Should emit SellTokens event", async function () {
  const tokenAmount = ethers.parseEther("100");
  const expectedEth = tokenAmount / 100n;

  await yourToken.connect(user1).approve(
    await vendor.getAddress(),
    tokenAmount
  );

  await expect(vendor.connect(user1).sellTokens(tokenAmount))
    .to.emit(vendor, "SellTokens")
    .withArgs(user1.address, tokenAmount, expectedEth);
});

it("Should fail if vendor has insufficient ETH", async function () {
  // First drain vendor's ETH
  await vendor.connect(owner).withdraw();

  const tokenAmount = ethers.parseEther("100");

```

```

describe("Sell Tokens", function () {
  it("Should emit SellTokens event", async function () {

    await yourToken.connect(user1).approve(
      await vendor.getAddress(),
      tokenAmount
    );

    await expect(vendor.connect(user1).sellTokens(tokenAmount))
      .to.emit(vendor, "SellTokens")
      .withArgs(user1.address, tokenAmount, expectedEth);
  });

  it("Should fail if vendor has insufficient ETH", async function () {
    // First drain vendor's ETH
    await vendor.connect(owner).withdraw();

    const tokenAmount = ethers.parseEther("100");
    await yourToken.connect(user1).approve(
      await vendor.getAddress(),
      tokenAmount
    );

    await expect(
      vendor.connect(user1).sellTokens(tokenAmount)
    ).to.be.revertedWith("Vendor has insufficient ETH");
  });

  it("Should fail with zero tokens", async function () {
    await expect(
      vendor.connect(user1).sellTokens(0)
    ).to.be.revertedWith("Specify an amount of tokens to sell");
  });
});

```

kết quả

✓ Should transfer the funds to the owner (71ms)

Token Vendor

Deployment

- ✓ Should set the right owner
- ✓ Should assign the token contract correctly
- ✓ Should have tokens in vendor contract

Buy Tokens

- ✓ Should buy tokens correctly
- ✓ Should emit BuyTokens event (45ms)
- ✓ Should fail if vendor has insufficient tokens
- ✓ Should fail with zero ETH sent

Sell Tokens

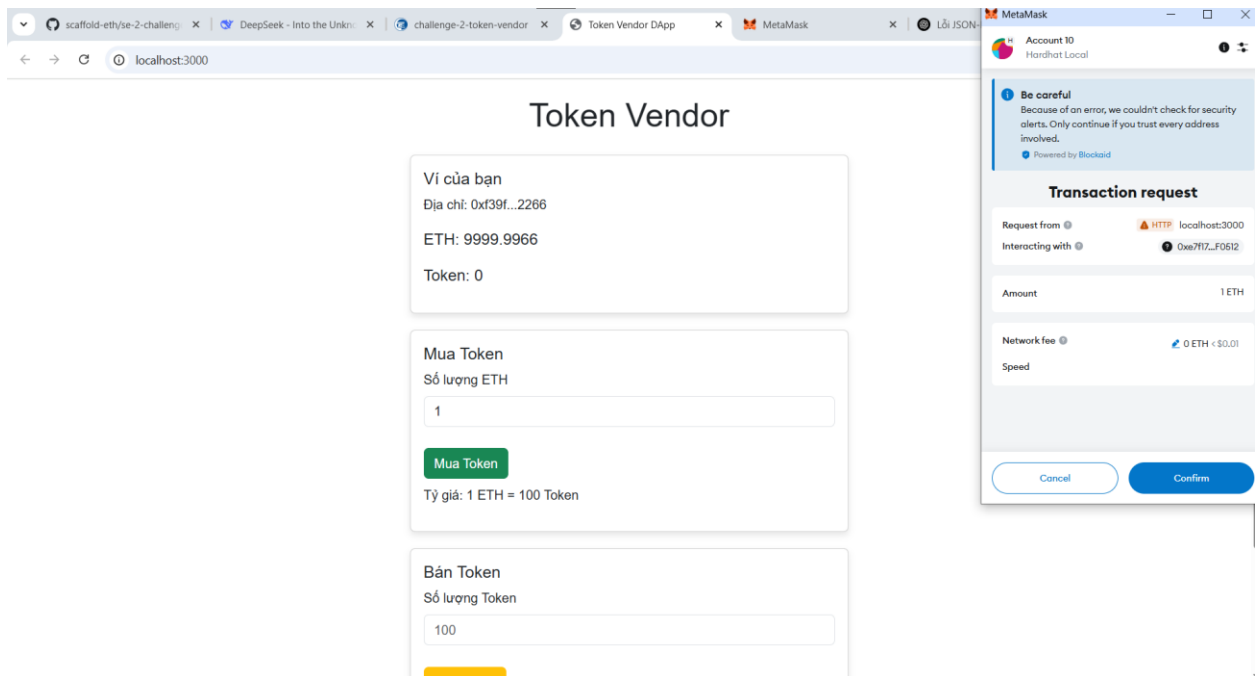
- ✓ Should sell tokens correctly
- ✓ Should emit SellTokens event (41ms)
- ✓ Should fail if vendor has insufficient ETH
- ✓ Should fail with zero tokens
- ✓ Should fail if vendor has insufficient ETH
- ✓ Should fail if vendor has insufficient ETH
- ✓ Should fail with zero tokens

Owner Functions

- ✓ Should allow owner to withdraw ETH
- ✓ Should prevent non-owners from withdrawing
- ✓ Should allow owner to deposit tokens
- ✓ Should prevent non-owners from depositing tokens

Edge Cases

- ✓ Should handle maximum token amounts correctly
- ✓ Should handle small amounts correctly





Account 10
Hardhat Local



Be careful

Because of an error, we couldn't check for security alerts. Only continue if you trust every address involved.

✓ Powered by [Blockaid](#)

Spending cap request

This site wants permission to withdraw your tokens

Estimated changes ?

You're giving someone else permission to spend this amount from your account.

Spending cap <0.000001 ? 0x0B306...97016

Spender ? ? 0x95992...007B1

Request from ? HTTP localhost:3000

Network fee ? 0 ETH < \$0.01

Speed

Cancel

Confirm