

Front End User Documentation

Yuang Huang

[3116337144@qq.com\(hyahhhhjj@gmail.com\)](mailto:3116337144@qq.com(hyahhhhjj@gmail.com))

Table of Contents

Overview.....	1
Installation.....	2
Usage.....	3
Note.....	9

1 Overview

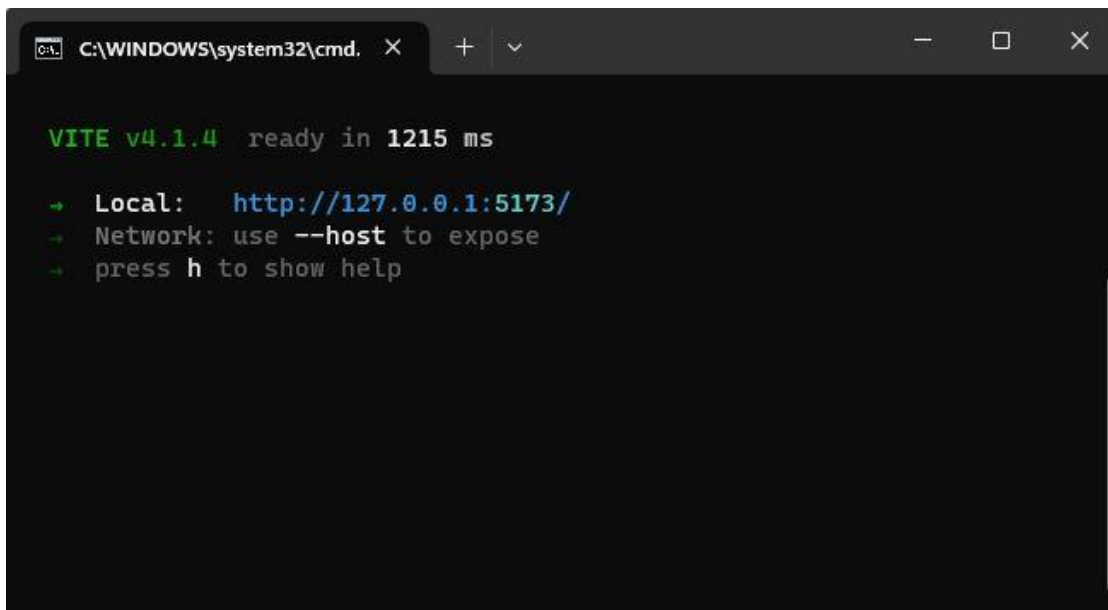
LiquidX Protocol frontend is mainly designed to facilitate the interaction between users and LiquidX Contracts and is developed based on Vite and React. Currently, it includes two interfaces. The first one is the Stake Page, whose interaction interface mainly corresponds to the LiquidXStakePool contract. The second one is the Manage Page, whose interaction interface mainly corresponds to the ManagerAccount contract. Please note that the frontend functions are still in early development and do not represent the final product quality.

2 Installation

To run the app, you need to make sure you have installed Node.js.

1. Open the command prompt (cmd) on your computer.
2. Navigate to the LiquidXFrontendV2-master folder where the project is located.
3. Type the command `npm run dev` to start the app.

That's it! The app should now be running and you can access it in your browser by visiting <http://localhost:3000>.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The terminal output displays 'VITE v4.1.4 ready in 1215 ms' in green. Below this, three lines of instructions are shown: '→ Local: http://127.0.0.1:5173/' in blue, '→ Network: use --host to expose' in grey, and '→ press h to show help' in grey.

```
C:\WINDOWS\system32\cmd. X + v - □ X  
  
VITE v4.1.4 ready in 1215 ms  
→ Local: http://127.0.0.1:5173/  
→ Network: use --host to expose  
→ press h to show help
```

3 Usage

The first interface that you should see when you open the website is this page:

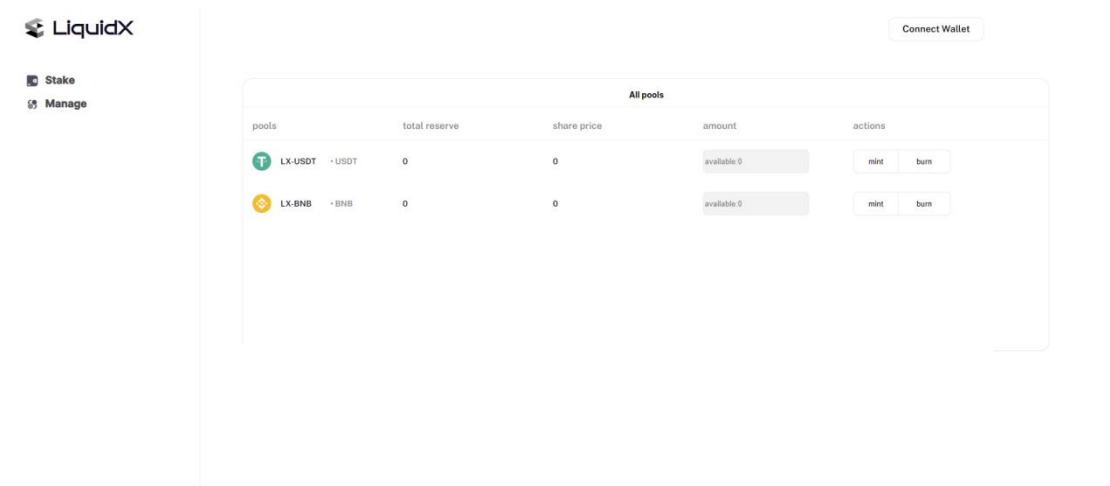


Figure 1 Stake Page

To log in, your wallet should pop up automatically similar to most other Web3 websites. If it doesn't, you can manually connect by clicking the "Connect Wallet" button.

To mint or burn your pool share token. You can type the amount of native token you want to transfer in or transfer out here:

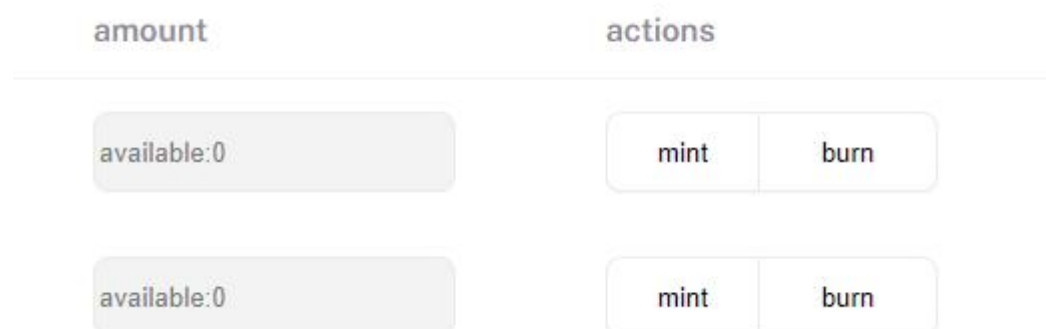


Figure 2 type amount

And click the “mint” or “burn” button.

Next, I will show you how to interact with the Manage page. Click the “Manage” button you will see this:



Figure 3 Create Account Button

Click it and you will set up your personal LiquidX Protocol manager account.

And it should be like this now:

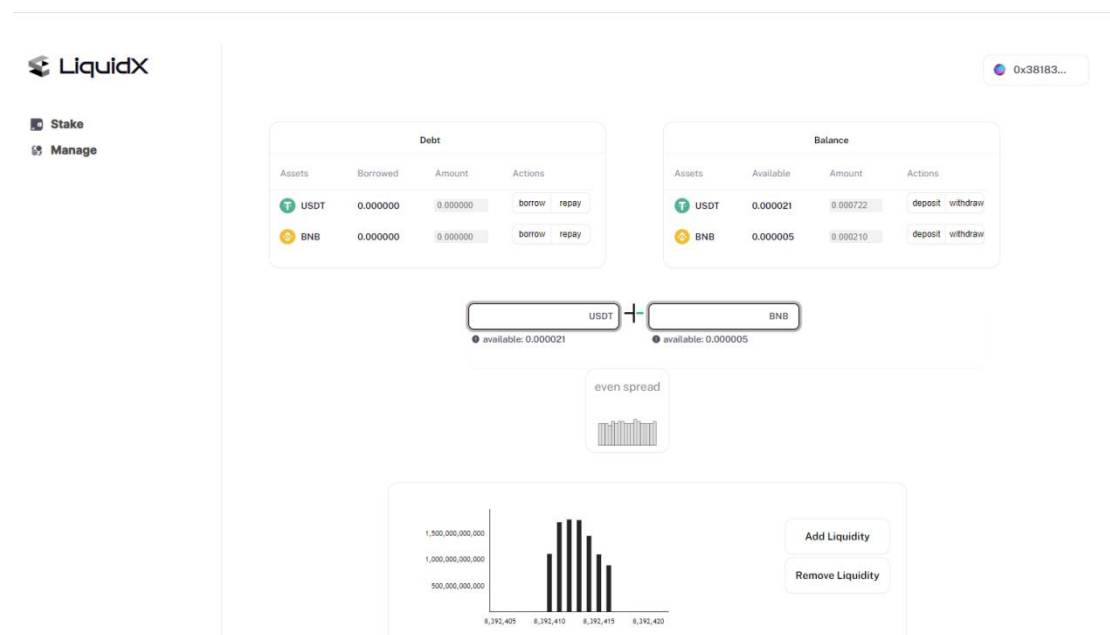


Figure 4 Manage Page

I will introduce the functions of this page section by section to you. The first one is section Debt:



Debt			
Assets	Borrowed	Amount	Actions
 USDT	0.000000	<input type="text" value="0.000000"/>	<button>borrow</button> <button>repay</button>
 BNB	0.000000	<input type="text" value="0.000000"/>	<button>borrow</button> <button>repay</button>

Figure 5 Section Debt

This section allows you to borrow a certain amount of native assets to add liquidity. To do so, simply input the desired amount and click on the “borrow” button. If you wish to repay your debt, click on the “repay” button. The corresponding functions in the “ManagerAccount.sol” contract are “borrow” and “repay”, respectively.

Please note that for a new account, the available amount of assets for borrowing would always be zero. And the corresponding mechanism has already been introduced in lite paper¹.

Now comes to section Balance.

¹ https://github.com/hyahhhhjj/LiquidXContracts/blob/master/documentation/LiquidXProtocol_LitePaper.pdf



Balance			
Assets	Available	Amount	Actions
 USDT	0.000021	<input type="text" value="0.000722"/>	<button>deposit</button> <button>withdraw</button>
 BNB	0.000005	<input type="text" value="0.000210"/>	<button>deposit</button> <button>withdraw</button>

Figure 6 Section Balance

This section allows you to deposit/withdraw a certain amount of tokens. The tokens deposited in the manager account can be used not only to directly add liquidity to the LB pair but also as margin when borrowing funds.

In the “Operating with Liquidity” section, we will focus on delivering a highly flexible experience for personal market makers through our custom liquidity provision tool.

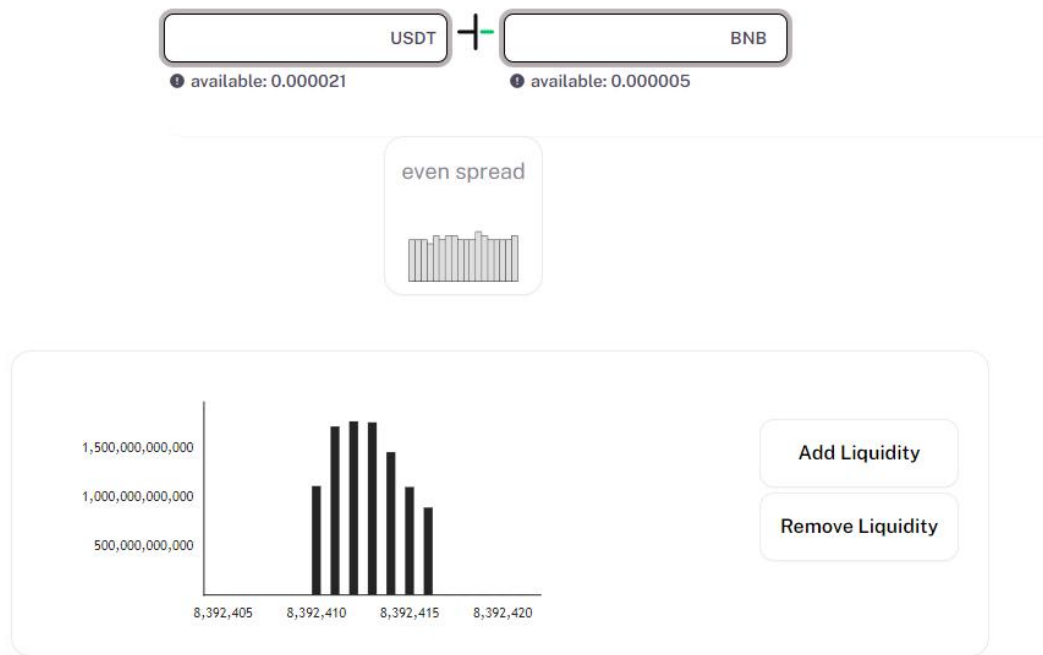


Figure 7 Section operating with liquidity

Please note that this section is still in development. The basic idea of this section is quite similar to one that in TraderJoeV2's website², but with more flexibility and customization providing to market maker. You can input different amounts of tokens into two input boxes, and the system will calculate a uniform distribution of liquidity for you to add. After adding liquidity, the chart will be updated with the added liquidity. You can further customize the liquidity by dragging the chart to remove liquidity. The updated liquidity will be displayed on the chart after each operation. In the future, we plan to add more liquidity distribution options and custom liquidity addition functionality.

² <https://traderjoexyz.com/>

To read the liquidity distribution chart, you must understand the working principles of the LB pair, which are described in LBPair.sol3.

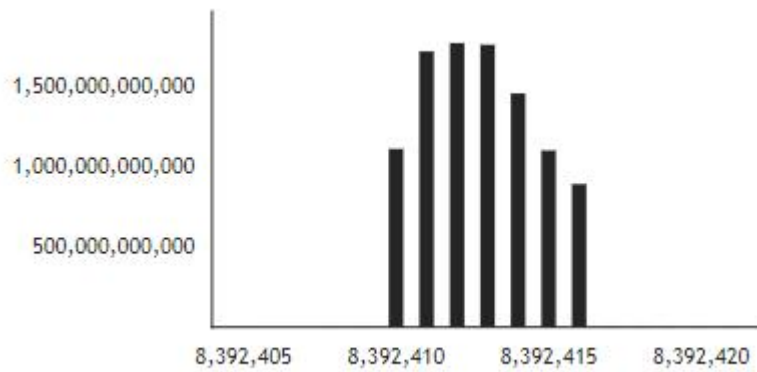


Figure 7 Liquidity Distribution

Generally speaking, x represents the bin ID introduced in Liquidity Book V2 and can be thought of as alternative pricing. Y represents the amount of LB token in a specific bin. Although we understand that this may not be very comprehensible, we aim to release a user-friendly update for the UI soon.

³ <https://github.com/traderjoe-xyz/joe-v2/releases/tag/v2.0.0>

4 Note

Before you start, you should get some tokens to test with.

```
(TBUSD):0x6658081AbdAA15336b54763662B46966008E8953
```

To get this token, you need manually call

```
function swapExactAVAXForTokens(
    uint256 amountOutMin,
    uint256[] memory pairBinSteps,
    IERC20[] memory tokenPath,
    address to,
    uint256 deadline
)
```

Using ILBRouter.sol interface. This file can be found in TraderJoeV2(Not V2.1).

If you are using Brownie, you can copy these code:

```
router = interface.ILBRouter("0xf7C6d73336f333b63144644944176072D94128F5")
router.swapExactAVAXForTokens(0, [15],
["0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd",
"0x6658081AbdAA15336b54763662B46966008E8953"],
"your address here", 1693399470,
{"from": get_account(), "value": amount})
```

And run `brownie run scripts/deploy.py --network bsc-test`. Then you will get some of these tokens.

If you are not using Brownie, here's some information you need:

```
LBRouter:0xf7C6d73336f333b63144644944176072D94128F5
Pair: 0x5f79ABacC763A61AD7ffEaa01a8b6Fd9F1856C2e
tokenX(TBUSD): 0x6658081AbdAA15336b54763662B46966008E8953
tokenY(WBNB): 0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd
binstep:15
```

The second token is

```
(WBNB):0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd
```

If you send some of testnet BNB to this address, you will automatically receive some of WBNB.