**How stability is achieved:**

Stability is done using volume controlled action. The volume/total supply of the token will always be between a particular limit settled by you and even owner will not be able to mint/burn against that volume. That gives the coin stability in the market against price manipulation.

**How volume is maintained?**

1. Every Time Minting is Done.

The token Minted transfers to the Owner Address Only.

If OWNER Try to MINT above maximum amount than only MAXIMUM Limit is minted to OWNER ADDRESS. Rest will be burned.

1. Every Time Burn is Done ...

The token's are burned from Owner Account only. if OWNER TRIES of BURN below minimum amount ONLY MINIMUM Limit AMOUNT will be left as total Supply and rest other coins burned.

This way the total supply will remain between min and max limit

**If there is anything we need to care about (for example liquidity pool:**

**Yes,**

To trade the tokens on any swapping platforms you have to add Liquidity pairs of your tokens in atleast 2-3 currencies according to the price ratio you need.

That way your token price would be defined.

Keep in mind, whatever amount you will keep in liquidity pool, majorly 10% of that will be tradable in one go. This is the feature of the Defi platforms which restricts whales to do bulk trading or manipulation.

**How to deploy the contract and how to keep price stable:**

1. First fill the .env file with your wallet’s private key and min and max volume limit you need (in 18 decimals).
2. Then in staked.sol , provide initial minting quantity in constructor.
3. Then run the following commands steps by step:
4. Test the contracts using : npx hardhat test
5. Compile your contracts using : npx hardhat compile
6. Deploy your contract using: npx hardhat run scripts/sample-script --network polygon (or your choice of network).

**We prefer price stability to be inside the contract so we don't need to call an externally contract to keep the price stable(externally mean outside the chain):**

The stability is defined inside the contract itself.Once deployed it will execute automatically.

The private key in .env file is just for contract deployment purpose.

**Describe how we need to create liquidity pools to keep prices stable for example how to provide liquidity for BNB/RUSD, BUSD/RUSD, etc:**

For this I have attached a document with the code named : Swap Model. Which includes how you can add liquidity and perform swap operations.

**Use the branch you already created QuestGLT:**

Yes we are using the same branch

**Write unit/integration tests:**

Yes, Integrated tests are written in the test/sample-test.js

**Don't edit or replace ERC20Rewards contract:**

It was creating conflicts with ERC20 token parameters like variable names , declarations are different in latest ERC20 and old ones

**Main modifications should happen on RUSD.sol contract (need to add stability to it 1:1 with USD)**

There was no file named RUSD.sol found in the repo. We did changes on Gold.sol but then you recommended we should do them in staked.sol.

About the price, it’ll be defined when you add liquidity in your price ratio on swapping platforms.