ChillieWallet Mobile Application

The purpose of this document is to explain the functionality of the ChillieWallet mobile application, and its associated ChillieWallet utility token. If you are reading this White Paper, that means the Wallet and Token contracts are ready for launch, but here is still much to get done on the accompanying mobile app.

I cannot provide solid delivery dates when things will be complete on the Application side. I am a professional Software Engineer, specializing in Android development, and I'm taking this Project on completely by myself. If the launch of the token on BSC goes well, I may be able to direct all my time towards development of the App. If things don't go so well, my time to develop the app will become scarcer. No matter what the case, the wallet is on the way, and you can get all the latest development updates on our Discord Server, and big news will always be available on our website!

The ChillieWallet utility token is used to Fuel an application that gives any crypto trader the ability to easily use Limit Orders on any Decentralized Exchange that is based on Ethereum's network. This Wallet will be scalable to use all EVM block chains such as BNB, ETH, MATIC, AVAX, etc. Limit Orders will have the ability to buy a token once it dips below a certain price, and then automatically sell when the token has risen in price. Stop Loss functionality will also allow the user to sell tokens in a crashing market. I intend to provide this functionality for an extremely low fee, and only collect taxes when the wallet has successfully made the user a profit. – This is the motivation behind the app and the fueling token!

What are the features of Chillie Wallet mobile application?

Privacy is number one, ChillieWallet will never capture any personal information about you such as Email, Real Name, etc.

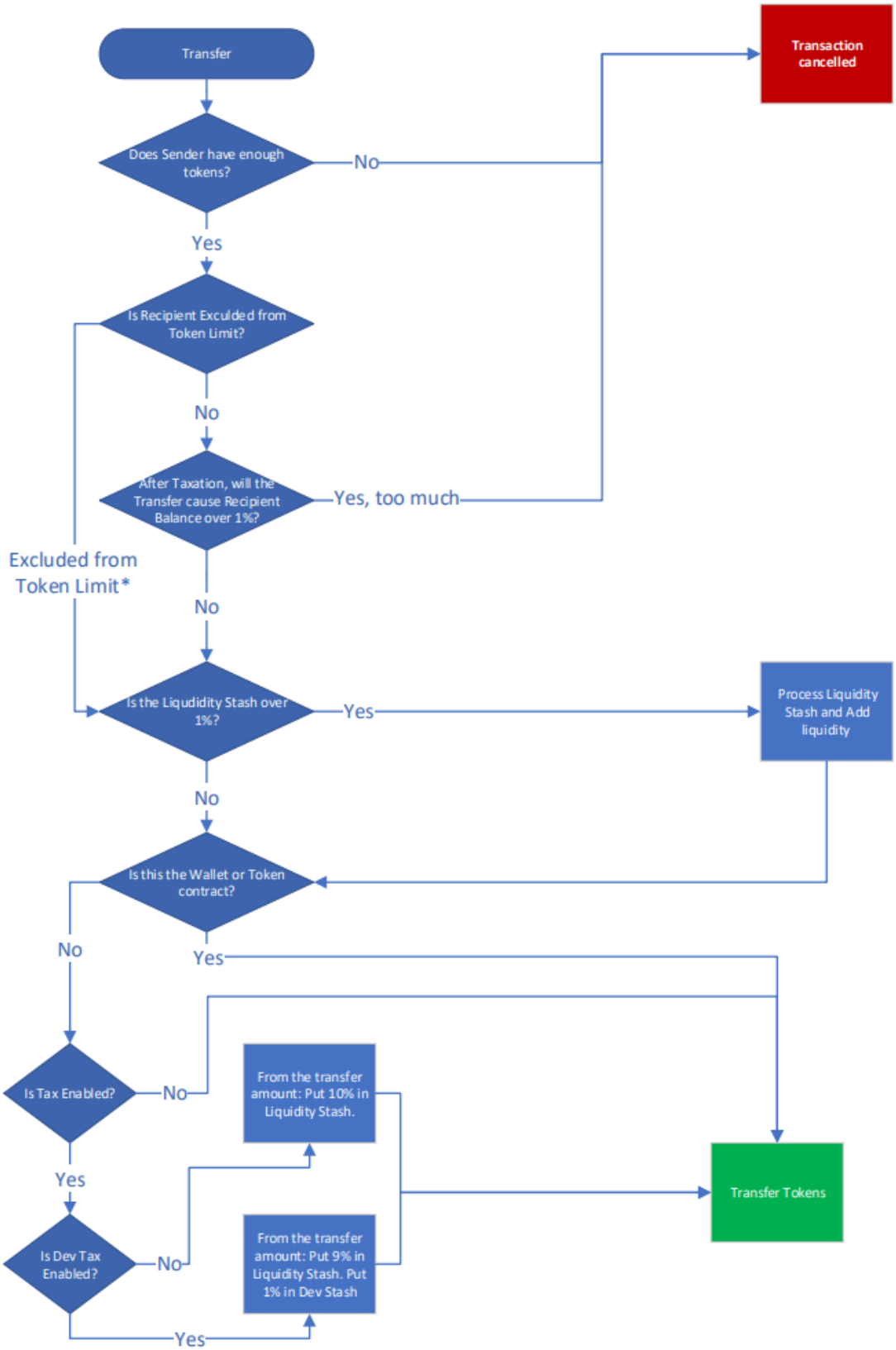
The idea of Chillie Wallet is to turn your phone into your own, full-time, day trader. The focus area is Tokens that are listed on a Decentralized Exchange such as UniSwap or PancakeSwap. One of the most annoying things about these exchanges, is there is no meaningful method of creating Limit Orders! It also can be annoying using the website interface of the DEX, especially when using a phone. With the ChillieWallet, you can be faster than the competition because the App is communicating directly with the Block Chain and Smart Contracts and will use the preprogrammed logic that you teach it to use.

Wouldn't it be nice if you could tell your phone to Buy a certain Token once the price dips to your target? Wouldn't it be nice to have a Stop Loss on your tokens, that way your exit strategy can automatically be executed when the market is crashing? Wouldn't it be nice to give your phone precise instructions on how it should Flip a certain token, and it just does it for you? And wouldn't it be nice to use this functionality on any DEX on a blockchain, even ones that you can manually add yourself? ChillieWallet is the answer to these questions!

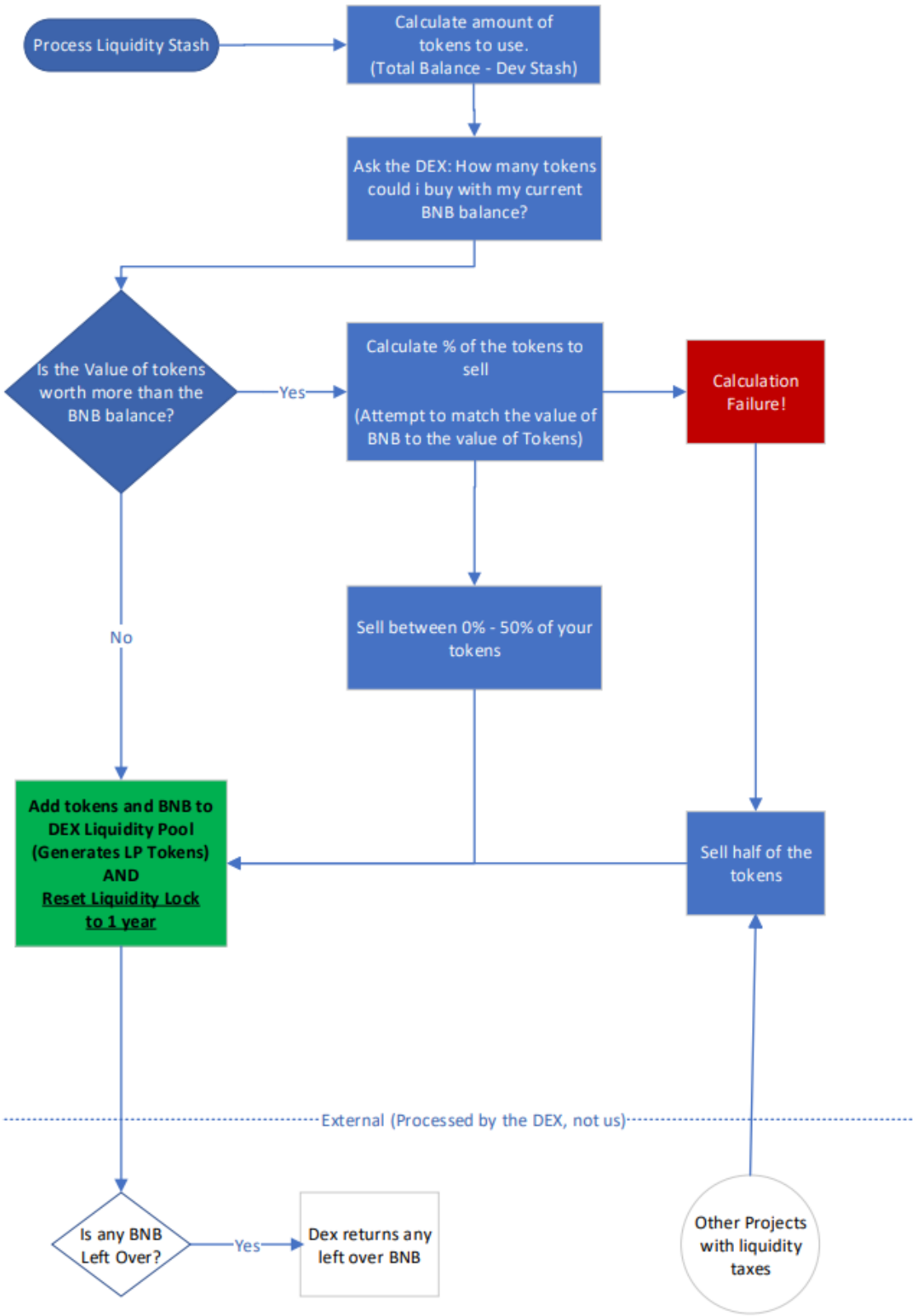
Another big plan of implementation is to have robust vetting of any token that is purchased using ChillieWallet. The ChillieWallet v2 will be focused on detecting common scams, and clearly warn the user before they buy something that they will most likely not be able to sell back. I have been scammed, rug pulled, honey potted, and lied to. I will provide a tool that can mitigate this damage.

Much more information to come on the App! Keep your eyes on https://chillieman.com for continuous updates on the ChillieWallet. I have not figured out what will be my main method of communication as it relates to Development of the Chillie Wallet, but I will make it obvious on the site so you can stay up to date!

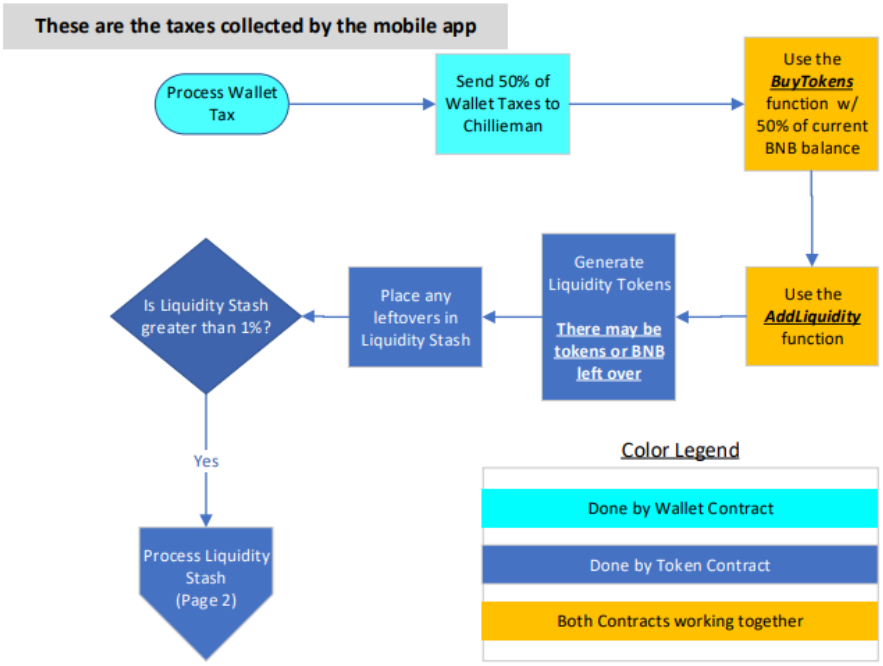
Token Transfers:



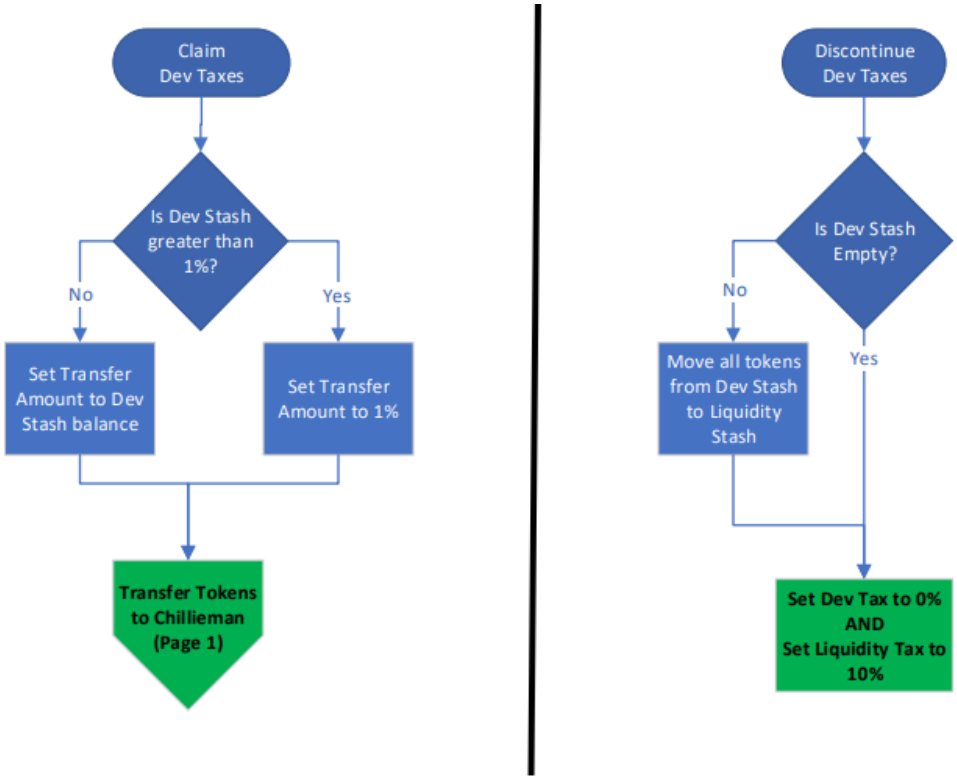
Process Liquidity Stash:



Process Wallet Taxes:



Development Stash:



ChillieWallet's Utility Token:

ChillieWallet token is an ERC-20 token that is inspired by DeFi projects that automatically add liquidity by taxing the transfers of the token. My starting point was based on Open-Source code found in https://github.com/OpenZeppelin/openzeppelin-contracts. I took inspiration from other popular and successful ERC-20 token. I like the tokens based on adding liquidity to constantly raise the bottom line, but I'm not a huge fan of the ones that revolve around Reflection and Burn protocols.

***Note that some tokens achieve an automatic burn by sending 50% of initial supply to a NULL address, causing half of the Reflections to end up burnt. If you think about that… that means that over 50% of the fees that were supposed to be collected for Reflections (for the community) end up in the trash...* **cough* lazy programming *cough**

I started by making a simple function for handling token transfers and implemented a straightforward function for collecting taxes. I made sure an account can never go above 1% of the total supply, or the transfer will fail. I believe this functionality will be a deterrent for whales, and stop people from causing massive volatility if they chose to sell a massive portion of the supply. The only exception to this rule is reserved for the Smart Contract, Chillieman, and any Exchanges. The Owner will have the ability to add exchanges to the list that is excluded from the max token limit. Any time an Exchange is added to the list, the Token will emit a **ExchangeAdded** event to retain full transparency. **ExchangeRemoved** event will be emitted if an exchange was removed from this special status.

I then started the code that will be responsible for adding liquidity. I took a look around other existing token with this Liquidity functionality and found a huge problem. The liquidity tokens were being purchased with the Owner listed as the recipient, giving the Owner instant access to all that liquidity. I analyzed the web and found another huge problem. UniCrypt wants 15,000 USDC to create a dynamic vault for the Liquidity Tokens… Since I'm a programmer, it just wouldn't feel right paying all that money when I can just implement a tiny bit of extra logic to get the Smart Contract to lock its liquidity.

To resolve this Liquidity issue, the token assigns itself as the receiver of the liquidity tokens, not the Owner. With each new batch of liquidity, a timer is set (or reset) to 1 year until any liquidity can be claimed. The only way to get the liquidity out is to not have meaningful token activity for a full year – Which I hope never happens, so that Liquidity should never see the light of day! If the liquidity is ever claimed, the Contact will emit a **LiquidityUnlocked** event to retain full transparency. Liquidity is added after the Liquidity stash (collected from transaction taxes) has grown larger than 1% of total supply (10 billion). Liquidity is also added when the Chillie Wallet Fee stash is processed.

I stripped out any code that gave the Owner the ability to change the Fee Rates, so fees can never be raised to any value the owner wants to set. I added new logic that allows Chillieman to turn the fees on and off globally (for everyone) which may come in handy in the future for events, holidays, and possible migrations. I added a lot of functionality that is designed to be used by the Wallet contract. A function for paying ChillieWallet Fees, and a function for processing those Fees, further fueling the liquidity of Chillie Coin. The two projects will constantly fuel the success of the other, causing a self-sustainable system, if people are using the Chillie Wallet! I will do everything in my power to make Chillie Wallet irresistible to ensure the success of the Token.

--- Flash Forward --

I enhanced the function that is called when the Wallet is processing Taxes. In rare cases where the Wallet would Inject much more ETH than exists in the liquidity pool, ensure a bunch of tokens art left in the Token Address - Process Liquidity both ways when adding a massive amount of ETH to handle it gracefully. Any excess ETH will stay in the Token Contract and be used the next time liquidity is generated. This allows the case where the Token Contract will not need to sell any tokens to add liquidity, avoiding the price ever from falling from a sale.

Furthermore, the calculation of how many tokens to sell when generating liquidity is very robust. Each time liquidity is generated, the ETH balance of the token will creep to 0, and in some case even reaching 0!! This is extremely exciting because there will never be the case where ETH get lost forever because they were accidently sent to the Token address. If you look

out there, there are Millions if not Billions of dollars' worth of ETH and BNB just sitting in a Tokens Balance, never to be used. - This contract solves that issue.

I used a fancy try / catch 🤓 in the Liquidity Generation function, which attempts to calculate a prime ratio of tokens to sell, but if the fancy logic does not work for whatever reason, the execution reverts to being simple and just selling Half of your tokens and provide liquidity with that. Apart of that calculation is to see if the $ value of ETH in the balance is greater than the $ value of the tokens that have been taxed to the liquidity stash. If the value of ETH is greater, then you don't need to sell any tokens, and simply add everything you have!

Oh, and I attempted to make the code as readable as possible to aspiring Token developers can have a solid example to go by! =D

…So, um… why should I trust you?

If it's not clear by now, I want to make this explicit: **This token and wallet are designed to be as fair as I can possibly make it.** Trust is one of the hardest things to gain, especially in the crypto space, as the number of scams out there is just ridiculous. Even though I have gone to great lengths to make the Smart Contract code as fair and safe as possible, I understand that it may not be easy, for those who aren't a geek like I am, to *see* that the Contract is fair and safe.

I will fight to gain your trust; any reasonable way I can. I am making it a priority to get a contact Audit as soon as I can, but I assume that's going to be a bit expensive and may have to wait until some Token Taxes (Or Donations) are generated so I can pay for it. I will keep up communication with the community, constantly providing Dev Updates and Sneak Peeks. I will embrace feedback from everyone, even my biggest critics, and address any issues that arise. I am genuinely interested in helping other people make money – **So let's do this!**

Token-omics:

1. Binance Smart Chain
    a. Name: **ChillieWallet**
    b. Symbol: **CHLL**
    c. Decimals: **18**
    d. Total Supply of Token: **1 trillion (1,000,000,000,000)**
    e. Initial Supply to PancakeSwap: **100%** + **1 BNB**
    f. Contract Address: **0x86b09825416612809e00947D0fEE05EC5853f62B**
2. Taxes: **10%**
    a. 9% will be taken to automatically provide liquidity
        i. Liquidity will automatically be locked in the Chillie Token contract for 1 Year. The 1 Year timer is reset every time the Liquidity stash is converted into LP Tokens.
    b. 1% will be taken to continue product development
        i. This will be turned off once the Chillie Wallet launches!
3. Liquidity production on a new level:
    a. The Wallet and Token are both designed to funnel as much liquidity into the token as possible.
    b. Aggressive recycling algorithms ensure that money is not being lost or stuck in the Wallet or Token address.
    c. The Generation of liquidity is elegant – a prime rate of tokens to purchase is calculated in a way that the BNB balance of the token will always creep to zero.
    d. Having the BNB balance creep to zero (or reach 0) means that the Token contract is using every bit of BNB it has and adding it to the liquidity – there is no waste!!!
    e. ChillieWallet generates fees for activation of the best features of the app, which will get added directly to the liquidity stash by the Wallet contract.
    f. ChillieWallet app generates taxes when it helps the user make a profit – These taxes are used by the Wallet Contract to buy tokens, and then add liquidity with it – Raising the price of the token!

4. Maximum Token Amount: **1% of Total Supply**
    a. I have designed ChillieWallet token in such a way that standard accounts cannot hold more than 1% of the total supply of tokens.
    b. While a standard account cannot purchase over 1%, they still can receive Token Rewards by using the ChillieWallet. There is no limit to the amount of Token Rewards you can earn by using the ChillieWallet!
    c. Exchanges are an exception, and must be able to hold more supply, as they typically hold the Tokens for their customers, until the customer decides to Withdrawal their tokens to a personal address.
    d. Exchange Exemptions can be removed or added by Chillieman as needed, which will allow the address to hold more than 1% of supply. The first Exchange will always be the UniSwap V2 clone related to blockchain which the ChillieWallet token was deployed. (i.e., PancakeSwap)
5. Burn: **None**
    a. The only tokens to be burned will come from my personal development address, and will manually be sent to 0x000000000000000000000000000000000000fAdED
    b. I strongly encourage you to NEVER burn your personal tokens for any reason
6. Whale Protection:
    a. First line of defense is the maximum token amount that a standard address (not an exchange) can hold. While this doesn't make it impossible to control, a Whale would need 100 different addresses to hold 10% of the total supply, which seems like more trouble than its worth.
    b. As the token is constantly funneling new Liquidity into the original exchange, it becomes increasingly harder to control the Token Price by adding and removing personal Liquidity.
    c. Providing and removing Liquidity is taxed at double the rate as typical transfers.
    d. Whales cannot Add/Remove liquidity at ridiculous amounts to manipulate the price of the token.

e. Adding a ton of liquidity and then removing it all in one fell swoop is how a Rug Pull is performed. This contract prevents anyone from doing that because someone can only Add liquidity that matches 10 Billion Tokens at a time (1% of supply) – and they can only remove liquidity that matches 10 Billion Tokens at a time (1% of the supply).

7. Functions that can only be called by Chillieman:
    a. **chillieToStandardAccount -**
        i. When ChillieWallet token is created, Chillieman will be exempted from Taxes by default (so no tokens are lost when adding the Initial Supply)
        ii. Once all Tokens have been supplied, this function will be called, and then Chillieman will be taxed on all future transactions, and must abide by the max token limit, just like everyone else.
        iii. Emits a **ChilliemanIsNowLimited** event to prove this function was called.
    b. **chillieSuspendTaxes** - Turn off All Taxes for Everyone
    c. **chillieResumeTaxes** - Turn on All Taxes for Everyone
    d. **chillieAddExchange** - Add an Exchange Address (Can hold more than 1%)
    e. **chillieRemoveExchange** - Remove an Exchange Address (Cannot hold more than 1%)
    f. **claimDevelopmentTax** - Claim the Development stash to fund development, site improvements, and marketing.
    g. **chillieDiscontinueDevTaxes** -
        i. Stop collecting 1% tax for the Dev Stash and set the full tax to be sent to the Liquidity Stash.
        ii. **Yes, I intend to stop collecting Dev Taxes on the Token once the Wallet is released**
    h. **unlockLiquidity** – A Failsafe function. If the token and app are a complete failure, Liquidity can be unlocked 1 full year after the last auto generation of liquidity
    i. **chillieSetWalletAddress** – Set the wallet address
8. Functions that can only be called by the Wallet address:

a. **walletAddToLiquidityStash** – Adds tokens from the Wallet to the Token -> directly into the liquidity stash.
b. **walletBuyTokens** – Wallet uses the Token contract to purchase tokens from PancakeSwap
c. **walletAddLiquidity** – Wallet uses the tokens bought from PancakeSwap, and uses it to generate Liquidity.