1. Trend Token Introduction

Trend Tokens allow for anyone to gain exposure to a dual momentum trading system acting on a portfolio of tokens, simply by holding the Trend Token in their Metamask wallet. Each Trend Token is 100% backed by its underlying assets. To purchase a Trend Token, supply one of the assets in the portfolio to the Trend Token pool. Send a Trend Token back to the pool to redeem a portfolio token of your choice.

If a token, such as BNB, is in a strong uptrend and therefore desired in the Trend Token portfolio, there will be a reward for supplying this asset and a fee for redeeming it. If the market turns and BNB is no longer desired, there will be a reward for redeeming this asset and a fee for supplying it. This creates a potential arbitrage opportunity that incentivizes market participants to keep the pool's allocations consistent with the dual momentum trading strategy.

An automated dual momentum trading system will update the deposit and redeem fees or rewards based on its strategy. Rather than the pool of assets incurring trading fees internally by interacting with a DeFi protocol such as PancakeSwap, the fees are passed on to the users via the rewards for depositing and redeeming assets as desired.

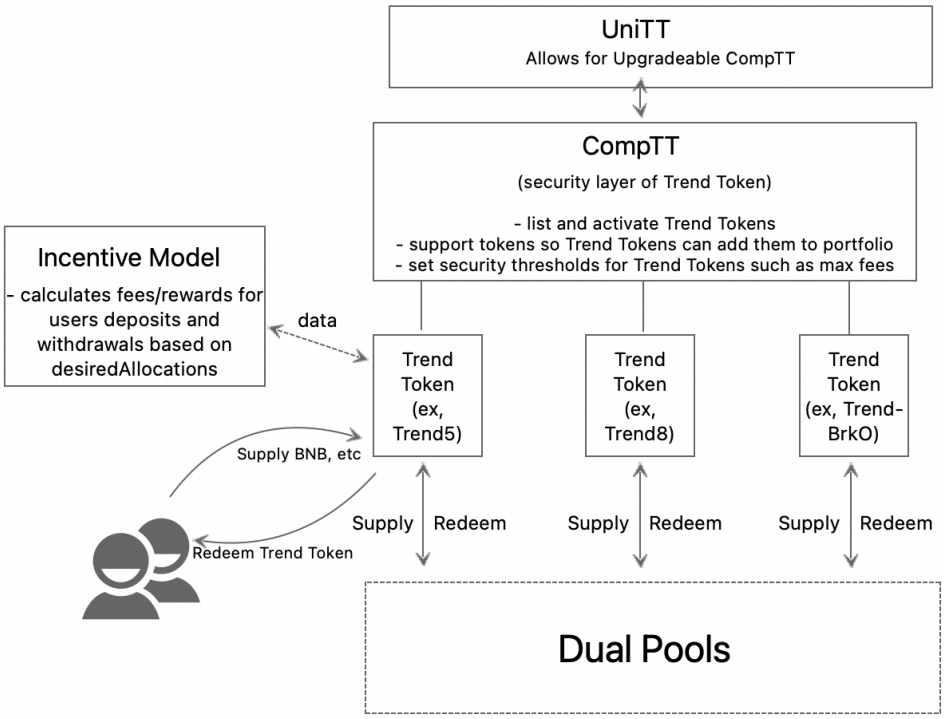
2. Portfolio Options

There will be types of Trend Tokens, each with a different portfolio theme. This is similar to the different types of portfolio options offered for Trendbot and Marginbot. To begin, three Trend Tokens will be offered.

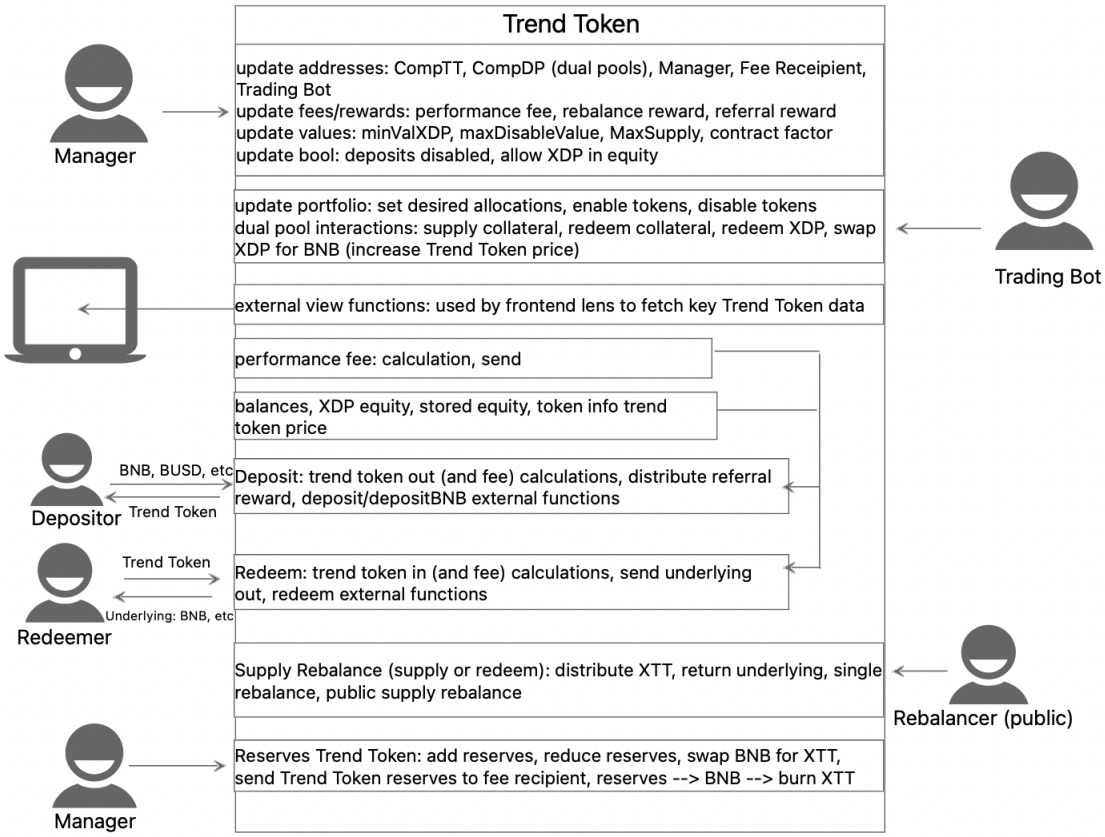
1) **TREND5:** Portfolio of BUSD, BNB, and 3 top tokens by liquidity on PancakeSwap.
2) **TREND8:** Portfolio of BUSD, BNB, and 5 top tokens by liquidity on PancakeSwap.
3) **Trend-BrkO:** Portfolio of BUSD, BNB, and 0-5 tokens making new all time highs out of the highest 10-20 liquid tokens on PancakeSwap.

3. Architecture

3.1 Overall Architecture



3.2 Trend Token Architecture



4.0 Deploy Instructions

4.1 Comptroller

Governs all deployed Trend Tokens and ensures safety parameters

1. Deploy CompTT
2. Deploy UniTT
    - Example: 0x20e0827B4249588236E31ECE4Fe99A29a0Ec40bA (verified)
3. Configure CompTT and UnitTT
    - _setPendingImplementation() in UniTT using CompTT address
    - _become() in CompTT using UniTT address
    - From now on, interact with CompTT using UniTT address but CompTT ABI
4. Add price Oracle to CompTT using _setPriceOracle()
    0x4aA176cCD63fC2C6FB881C568FE214889Cff4c6C (from Dual Pools)

5. Support tokens/dTokens using _setToken(underlying, dToken), dTokens from Dual Pools
    a. BNB: 0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd,0x243fF2E429B4676d37085E7b5a1e1576f11508f3
    b. BUSD: 0x8301F2213c0eeD49a7E28Ae4c3e91722919B8B47,0x2a98C6E2BD140513df99FFCC710902a2faFb3bb7
    c. BTC: 0xA808e341e8e723DC6BA0Bb5204Bafc2330d7B8e4,0x67DAB885c014FBB42a73f15F87953EE9c619910a

4.2 Trend Tokens

1. Deploy XTT from XTTgov.sol
2. Add XTT and XDP in TrendTokenStorage.sol under xtt and xdp variables
    XTT: 0x4D0E7Cd2A4f6D45d72B7936DDb8652aa3216A51e (verified)
    XDP: 0x4Dfc503D91A098AFB30056f6b9e38D40e4b9d714 (from Dual Pools)
3. Deploy IncentiveModelSimple from IncentiveModelSimple.sol
    Example: 0xc2b0706227D1c991D508AAE81b86253E86DeF30B (verified)
4. Deploy TrendToken
    - _wbnb, _compTT, _compDP, _pancakeRouter
    - 0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd,0x20e0827B4249588236E31ECE4Fe99A29a0Ec40bA,0x022d21035c00594bdFBdAf77bEF76BBCe597d876,0xd99d1c33f9fc3444f8101754abc46c52416550d1
5. Support Trend Token in CompTT
6. Add incentive model to Trend Token in _updateContracts(0x00, 0x00, addr)
    - 0x0000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000,0xc2b0706227D1c991D508AAE81b86253E86DeF30B
7. Add Portfolio
    - [0xae13d989daC2f0dEbFf460aC112a837C89BAa7cd,0x8301F2213c0eeD49a7E28Ae4c3e91722919B8B47],[1000000000000000000,0]

5.0 Test Cases

5.1 Deposits (all good)

Deposit an underlying asset such as BNB for Trend Tokens. Token that was deposited must be included in the portfolio but also not in depositsDisabled. If Trend Token desires more of an asset, it will reduce the protocolFeeDeposit or even exceed it for a net reward. If Trend

Token desires less of an asset, it will add a fee in addition to protocolFeeDeposit. The dToken associated with the underlying asset being deposited must also be enabled.

If a depositor holds XTT, they will get a discount on protocolFeeDeposit up to 80%. If _referrer parameter in deposit function is not the zero address, then referralReward% of remaining protocolFeeDeposit will be sent to the referrer and referralReward% sent to the referral.

If Trend Token makes an all time high, the performance fee will be sent to the feeRecipient. **This may be removed for tax purposes,**

As governed by the Trend Token comptroller,, the amount being deposited must be greater than 0, mint guardian cannot be paused, and the trend token must be listed and active.

5.1.1 Tests

Deposit a token that is not in the portfolio (good)
- Expect error
- Result: error

Deposit token that is in depositsDisabled (good)
- Expect error
- Result: error, set back to false and worked

Try depositing 0 (good)
- Expect error
- Result: error

Deposit token that is in demand (good)
- Expect fee lower than protocolFeeDeposit
- Trend Token price should go up if protocolFeeDeposit > -feeOrReward
- Result: reward was distributed and trend token price dropped a little

Deposit token that is not in demand (good)
- Expect higher fee than protocolFeeDeposit
- Trend Token price should go up more than last deposit
- Result: fee of 0.30% (0.20 + 0.10) and trend token price increased proportionally

Add referral (good)
- Expect referralReward% of protocolFeeDeposit to go to referrer and referral
- 40% of fee was distributed to referral and referrer

In CompTT, set mintGuardianPaused to True and active to false (good)
- Expect error

Set fees to 0% and deposit (good)
- Performance fee, referral reward,
- Make sure no error

5.2 Redeem (all good)

Redeems allow users to redeem their Trend Tokens for an underlying asset of their choice. The underlying asset must be enabled and have a sufficient balance for what the amount they wish to redeem. If Trend Token desires less of an asset, it will reduce the protocolFeeRedeem or even exceed it for a net reward. If Trend Token desires less of an asset, it will add a fee in addition to protocolFeeRedeem. The dToken associated with the underlying asset being redeemed must also be enabled.

There is up to a 80% discount on protocolFeeRedeem based on how much XTT the user holds. However, there is no referral reward. The protocolFeeRedeem is charged based on the amount of Trend Tokens deposited and goes directly to reserves or burned based on trendTokenRedeemBurn%

If Trend Token makes an all time high, the performance fee will be sent to the feeRecipient. **This may be removed for tax purposes,**

As governed by the Trend Token comptroller, the amount being redeemed must be greater than 0, mint guardian cannot be paused, and the trend token must be listed and active.

1) Redeem a token that is not in the portfolio (good)
    - Expect error
    - Result: error
2) Try redeeming 0 (good)
    - Expect error
    - Result: error
3) Redeem token with not enough balance (good)
    - Expect error
    - Result: error
4) Redeem token that is not in demand (good
    - Expect fee lower than protocolFeeRedeem
    - Trend Token price should go up if protocolFeeDeposit > -feeOrReward
    - Result: reward was distributed and trend token price dropped a little
5) Redeem token that is in demand (good)
    - Expect higher fee than protocolFeeRedeem
    - Trend Token price should go up more than last redeem
    - Result: fee of 0.30% (0.20 + 0.10) and trend token price increased proportionally
6) In CompTT, set mintGuardianPaused to True and active to false (good)
    - Expect error
7) Set fees to 0% and deposit (good)
    - Performance fee
    - Make sure no error

5.3 Trade (all good)

Swaps one underlying asset (such as BNB) for another (such as BUSD). If you sell an asset Trend Token wants and buy an asset the Trend Token does not want, there would be a large reward.

1. Sell one token for another, both in portfolio (good)
    - Works for sell BNB and token
2. Sell one token for another, one not in portfolio (good)
    - Expect error
    - Result: error
3. Buy token not enough balance (good)
    - Expect error
    - Result: error
4. Sell amount 0 (good)
    - Expect error
    - Result: error
5. Return amount beyond minOut (good)
    - Expect error
    - Result: error

5.4 Rebalance (all good)

1. Rebalance token not in portfolio (good)
    - Expect rebalance but no action
    - Result: rebalance but no action
2. Rebalance token with no balance but in portfolio (good)
    - Expect rebalance but no action
    - Result: rebalance but no action
3. Rebalance token in portfolio with balance (good)
    - Expect rebalance successfully
    - Result: rebalance successful
4. Move contract factor to 0% (good)
    - Expect rebalance to 100% supplied
    - Result: 100% supplied
5. Move contract factor to 100% (good, fixed)
    - Expect 100% to contract
    - Result: error
        - Might need to adjust how it behaves when contract factor > 50%
        - At 0.50, 0.80 fails but 0.70 succeeds

5.5 Manager Actions

- Make sure fees are sent to the right person?
- Retest redeem XDP and XDP going to right person

5.5.1 _updateContracts(address _compTT, address _compDP, IIncentiveModelSimple _incentiveModel) (all good)

Updates key addresses that Trend Token contract interacts with

1. Add random addresses and check stored addresses
    - Expect: stored addresses

5.5.2 _updateAddresses(address _manager, address payable _feeRecipient, address _tradingBot) (all good)

Updates manager, feeRecipient, and tradingBot

1. _reduceTrendTokenReservesToRecipient(uint redeemAmtTrendToken)
    - Expect: redeemAmtTrendToken Trend Tokens to fee recipient
2. _redeemXDP()
    - Expect: accruedXDPtoFeeRecipient% XDP to fee recipient

5.5.3 _newPerformanceFee(uint _performanceFee) (all good)

Update performance fee, percentage of all time high that goes to trend token reserves

1. _distributePerformanceFee() (good)
    - Expect performance fee distributed if new ATH made
2. Set performance fee beyond trendTokenMaxPerformanceFee() (god)
    - Expect: error
3. Set performance fee to 0 and redeem (good)
    - Expect: no distribution

5.5.4 _updateFeeDistribution(uint _trendTokenRedeemBurn, uint _accruedXDPtoFeeRecipient) (all good, except xdp, RETEST)

Adjusts the amount of trend tokens from redeems that get burn (instead of going to reserves, and percentage of accrued XDP that goes to fee recipient (instead of staying in pool)

1. _redeemXDP() (had to fix xtt to xdp)
    - Expect: accruedXDPtoFeeRecipient% XDP to fee recipient
2. Set _trendTokenRedeemBurn to 0 (good)
    - Expect: All Trend Token fee upon redeem goes to reserves

3. Set _trendTokenRedeemBurn to 100% (good)
    - Expect: No increase in reserves, higher increase in Trend Token price

5.5.5 _setReferralReward(uint _referralReward) (all good)

Sets the percentage of deposit protocol fee that goes to referrer and referral, remaining goes to Trend Token holders

1. Set _referralReward to 60% (good)
    - Expect error
2. Set _referralReward to 0% (good)
    - No referral reward
3. Set _referralReward to 50% (good)
    - Referral reward highest

5.5.6 _maxDisableValue(uint _maxDisableTokenValue) (all good)

Sets the maximum value that can be held in an underlying asset before it can be disabled. Otherwise could disable an asset that has a high balance and would drop Trend Token price

1. Disable BNB (good)
    - Expect: error
2. Set max value to 0, try to remove an asset with >0 balance
    - Expect: error
3. Set max value above 10000
    - Expect: error, exceed compTT
4. Set max value to 100000e18, try to remove an asset (good)
    - Expect to be able to remove, but Trend Token price fall a lot
5. Enable token back (good)
    - Expect: Trend Toke price to bounce back

5.5.7 _setMaxSupply(uint _maxSupply) (all good)

Limits the maximum supply of Trend Tokens. Will prevent deposits otherwise

1. Set max supply below current supply and deposit (good)
    - Expect error
2. Set max supply just above current and deposit to exceed (good)
    - Expect error
3. Set max supply far above current and deposit to not exceed (good)
    - Expect successful deposit

5.5.8 _setContractFactor(uint _contractFactor) (all good)

Sets the percentage of token balances that stay in the contract. The remainder get supplied to Dual Pools

1. Vary contract factor from 0-100% and public rebalance
    - Expect to redeem/supply accordingly (required a fix)

5.6 Trading Bot Actions

Trend Token reserves may be: burned, swapped for XTT and burned, or send to trading bot

- Redeploy for _swapXDPforBNB
- Maybe remove _swapXDPforBNB and _redeemTrendTokenReservesBurnXTT

5.6.1 _depositsDisabled(address underlying, bool _state) (all good)

Enables or disables underlying asset from being deposited

1. Disable a token and try to deposit (good)
    - Expect: unable to deposit
2. Enable token form above (good)
    - Expect: able to enable

5.6.2 _pauseTrendToken(bool _pause) (all good)

Pauses deposit, redeem, rebalance, and trade

1. Pause Trend Token and try to deposit, redeem, rebalance, and trade
    - Expect: error
2. Unpause Trend Token and try to deposit, redeem, rebalance, and trade
    - Expect: able to execute

5.6.3 _setDesiredAllocations(uint[] memory _allocations) (all good)

Sets new allocations according to the portfolio list. Must be same length

1. Sum of allocations to 0
    - Expect: error
2. Sum of allocations above 100%
    - Expect: error

3. Set allocation of improper length
4. Vary allocations and supply/rebalance to check accuracy of fees

5.6.4 _enableTokens(address[] calldata _tokens, uint[] calldata _allocations) (all good)

Enables new portfolio and sets associated allocations

1. Enable tokens that arent supported in CompTT (good)
    - Expect: error
2. Enable tokens that are supported in CompTT (good)
    - Expect: enabled successfully
3. Enable same token twice (good)
    - Result: successful, but not shown twice in portfolio
4. Deploy new ETH and USDT dTokens, enable them (should be good)
    - Successful rebalance

5.6.5 disableToken(IERC20 _bep20, uint[] calldata _allocations) (all good)

Disables a token and sets new associated allocations

1. Disable a token and set new allocation of improper length (good)
    - Expect: error
2. Disable a token and set new portfolio allocations (good)
    - Expect: success

5.6.6 _swapXDPforBNB(uint _sellAmountXDP, uint _minOut) (retest)

Swaps contract XDP for BNB and adds to portfolio values

1. Sell XDP balance beyond current
    - Expect: error
2. Sell XDP but set minOut extremely high
    - Expect: error
3. Sell XDP with sufficient balance
    - Expect: success

5.6.7 _reduceTrendTokenReservesToRecipient(uint redeemAmtTrendToken) (all good)

Sends trend token reserves (from redeems and performance fees) to fee recipient

1. Redeem Trend Token amount beyond balance (good)
    - Expect: error
2. Redeem Trend Token amount below balance (good)
    - Expect: success

5.6.8 _supplyCollateral(IERC20 _depositBep20, uint supplyAmt) (all good)

Manually supplies collateral

1. Supply token that isnt in portfolio
    - Expect: error
2. Supply token in portfolio with insufficent balance
    - Expect: error
3. Supply token in portfolio with sufficient balance
    - Expect: success

5.6.9 _redeemCollateral(IERC20 _redeemBep20, uint _redeemAmt) (all good)

Manually redeems collateral

1. Redeem token that isn't in portfolio
    - Expect: error
2. Redeem token in portfolio but not enough supply
    - Expect: error
3. Redeem token in portfolio with enough balance
    - Expect: success

5.6.10 _redeemXDP() (need to restest) (all good)

1. Redeem (might need to supply XDP to Dual Pools) (good)
    - Expect: XDP sent to Trend Token, proper amount to fee recipient

5.5 Update Contracts

Only Comptroller is upgradeable right now

5.5.1 Comptroller

1. Deploy new comptroller
2. "At Address" with existing CompTT and UniTT ABI
3. _setPendingImplementation() in UniTT
4. _become() in CompTT

6.0 Notes

- TrendToken.sol doesn't have enough contract room to emit events