Token Vesting Project (BSC)

Requirements:

1. Token contract with set max supply
2. Verification of contract with no bugs or any errors
3. Token must have option to be sent frozen to investors, this freeze will be able to get unfreeze AUTOMATIC with specific percentages every month and in specific dates we will agree on
4. Token must have the function of being sent Manually to private sale investors and have the same freeze / unfreeze option and schedule with it.
5. Token must be eligible to purchased from launch pads for public sale and have all requirements required from launchpads
6. The total tokens purchased will be sent to Investors on first purchase but will not be controlled by him until contract condition is done
7. Manual function to unfreeze any amount in specific wallet if the owner wants to do this immediately
8. Functionality to handle multiple vesting and simultaneous **purchase from DEX**
9. Token must have burn feature

Contract overview: (additional functions on top of basic erc-20 functions)

1. Function to manually add vesting
    0. Caller passes address of beneficiary, amount to be vested
    1. Set percentage to be unlocked every month
    2. Set percentage to be unlocked on specific date and also the date
    3. Optional set time to unfreeze all assets

2. Function to see how many tokens are frozen in a specific account, how many vesting schedules are there for a specific wallet, how many they can spend, and how many have already been vested.
    0. Caller passes address of the user

1. Function to set launchpad vesting details (Can handle multiple launchpads and offerings)
    0. Caller passes launchpad presale address
    1. Set percentage to be unlocked every month
    2. Optional set time to unfreeze all assets
2. Function to manually unfreeze amount in specific wallet

a. Caller passes address of wallet and also the amount they want to unfreeze

The transfer function will be overridden in such a way that it can detect frozen assets and assets purchased through a DEX. It will allow only the transfer of unfrozen assets. There will be a mapping that maintains the books for multiple vesting per address.

The timeline will be two weeks, and the deliverables will be the following:

1. Smart Contract (Bug Free, without any severities)
2. Rigorous local testing
3. Fork the bsc chain and test with Pinksale
4. Fork the bsc chain and test with Pancakeswap
5. Test on testnet
6. Document with instructions on how to deploy on mainnet and how to use the contract

Smart Contract (contract + local testing) - 7-10 days ($400)
Testnet testing - 3 days ($200)
Document - 1 day ($300)

Key Features Contained in Smart Contract

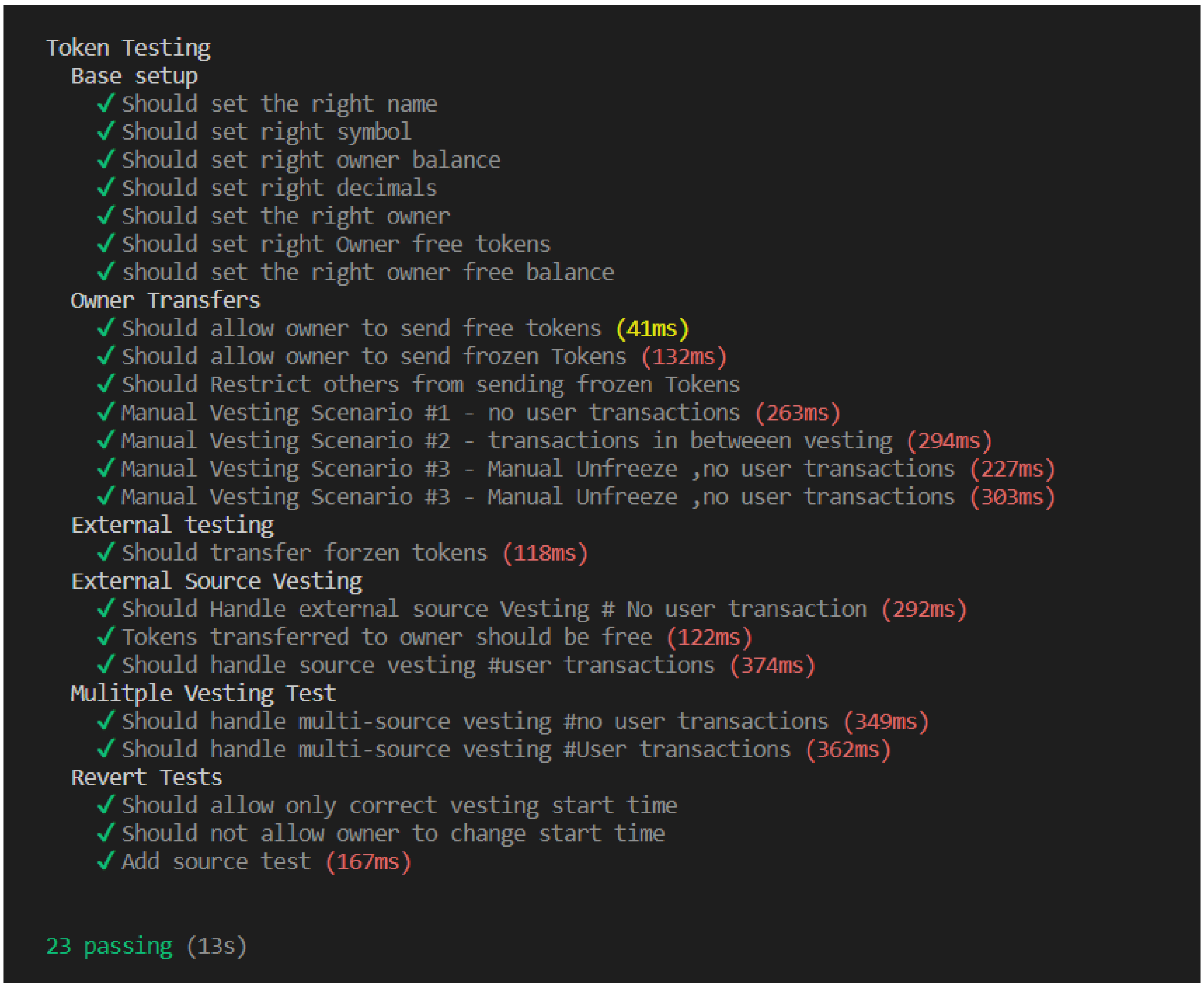
Table 1.1 View Only Functions in contract:

	Name	Description	Status	Access
1.	allowance(owner,spender)	Returns the amount that the spender is still allowed to withdraw from the owner.	✓	public
2.	balanceOf(user)	to get total tokens owned by the user.	✓	public
3.	startTime()	Get Vesting start time	✓	public
4.	checkSource()	Check if a particular source is added	✓	public
5.	decimals()	Returns decimals, which refers to how divisible a token can be.	✓	public
6.	getAmountVested(user)	get total amount of tokens that has been vested to user	✓	public
7.	getFreeTokens(user)	get amount of tokens the user can spend	✓	public
8.	getFrozenTokens(user)	get amount of tokens a user has but cannot be used yet.	✓	public
9.	getVestingCycles(user)	Number of vesting cycles user is yet to receive	✓	public
10.	name()	name of the token	✓	public
11.	Owner()	Current contract Owner	✓	public
12.	symbol()	Returns the symbol by which the token contract should be known, for example “ABND”. It is usually 3 or 4 characters in length.	✓	public
13.	totalSupply()	Returns the number of total supplied tokens, including the total minted tokens (minus the total burned tokens).	✓	public

Table 1.2: **Key state changing functions contained in the contract**

	Name	Description	Status	Access
1.	constructor(name,symbol ,initialSupply)	Used during Token deployment	✓	-
2.	SetStartTime(date)	to set vesting start time; // (3rd September)	✓	onlyOwner
3.	addSource(address ,initialReleasePercentage, monthlyReleasePercentage)	To add new source from where users can get Token and set the vesting standard and allowance.	✓	onlyOwner
4.	unfreezeAmount(user, minAmount)	to unfreeze tokens for any user at any point. * this sets the next release time as current time + vesting Period	✓	onlyOwner
5.	approve(spender,amount)	Approve an address to spend on another's behalf	✓	public
6.	burn(amount)	allows a user to burn their token.	✓	public
7.	burnFrom()	Burn tokens of the address that you have a spend allowance for	✓	public
8.	increaseAllowance()	Increase spend allowance	✓	public
9.	decreaseAllowance()	Decrease spend allowance	✓	public
10.	transferOwnership()	Transfer ownership of the contract to specified address. Only the owner can call this.	✓	onlyOwner
11.	renounceOwnership()	Reassigns ownership of the contract to null address. Only the owner can call this.	✓	onlyOwner
12.	sendFrozen(to,amount,initialReleasePercentage,monthlyReleasePercentage)	for owner to send frozen tokens	✓	onlyOwner
13.	transfer(address,amount)	to transfer tokens. • When used by owner, transfers free tokens. • When used by a listed source, transfers frozen tokens.	✓	public
14.	transferFrom()	transfer tokens from a source user has allowance for.	✓	public

Unit Testing:



```
  Token Testing
    Base setup
      ✓ Should set the right name
      ✓ Should set right symbol
      ✓ Should set right owner balance
      ✓ Should set right decimals
      ✓ Should set the right owner
      ✓ Should set right Owner free tokens
      ✓ should set the right owner free balance
    Owner Transfers
      ✓ Should allow owner to send free tokens (41ms)
      ✓ Should allow owner to send frozen Tokens (132ms)
      ✓ Should Restrict others from sending frozen Tokens
      ✓ Manual Vesting Scenario #1 - no user transactions (263ms)
      ✓ Manual Vesting Scenario #2 - transactions in betweeen vesting (294ms)
      ✓ Manual Vesting Scenario #3 - Manual Unfreeze ,no user transactions (227ms)
      ✓ Manual Vesting Scenario #3 - Manual Unfreeze ,no user transactions (303ms)
    External testing
      ✓ Should transfer forzen tokens (118ms)
    External Source Vesting
      ✓ Should Handle external source Vesting # No user transaction (292ms)
      ✓ Tokens transferred to owner should be free (122ms)
      ✓ Should handle source vesting #user transactions (374ms)
    Mulitple Vesting Test
      ✓ Should handle multi-source vesting #no user transactions (349ms)
      ✓ Should handle multi-source vesting #User transactions (362ms)
    Revert Tests
      ✓ Should allow only correct vesting start time
      ✓ Should not allow owner to change start time
      ✓ Add source test (167ms)


  23 passing (13s)
```

- **BSC Testnet Token Address:** 0xfB0bD9759a22334390B5980ceB2B89F0AF091754 (Link)

- **PinkSale Launchpad:** 0xC6068c114291B0AceB139ca1Dd82032e3Cc8954D {Link)

Steps to test:

Constructor Parameters – name, symbol, max supply.

1. **Testing with Pinksale:**
    1. Deploy on Testnet.
    2. Set vesting start time.
    3. Create Launchpad on pinksale (This step will transfer some tokens to pinksale contract)
    4. Add pre-sale contract address as listed source along with release percentages using *[addSource ()]* function. (Must be done before finalizing pre-sale contract).
    5. Contribute to the pre-sale using different metamask address.
    6. Once the sale has completed, Finalize the pre-sale on pinksale using owner's address.
    7. Claim the tokens using contributor address.
    8. Use [*getFreeTokens (), getFrozenTokens ()*] function to confirm right amount of tokens are released at right time. Note: [*balanceOf ()*] will display total tokens, as required.

2. **Testing [*sendFrozen ()*] functionality:**

    1. Deploy on Testnet.
    2. Set vesting start time.
    3. Use [*sendFrozen ()*] to transfer frozen tokens to an address.
    4. Use [*getFreeTokens (), getFrozenTokens ()*] function to confirm right amount of tokens are released at right time by the recipient. Note: [*balanceOf ()*] will display total tokens, as required.
    5.

Note to Owner:

- Contract Deployer is the default owner.
- For testing purpose, vesting period has been set as 5 minutes. Before actual deployment to main-net, **this must be changed to desired value at line number 15 in contract code.**
- Owner's transactions done using [*transfer ()*] function will send Free tokens. Owner must use dedicated function to send Frozen token. This also includes [*transferFrom ()*] functions where amount is being deducted from owner.
- Owner must set Vesting start date before any transfer of Frozen Tokens is done; otherwise, the transaction will be reverted. This includes transfer from pre-sale contracts.
- Once set, vesting start date cannot be changed.
- Owner must add pre-sale contract's address as a listed source before any token is transferred from it; else, these tokens will be free and receiver will be able to use them immediately.

- Monthly release percentage for any source must divide (100 – initial release percentage). This helps contract to work with using minimum blockchain storage and hence minimizes gas consumption.
- In case of ownership transfers, owner must also take care of owner’s tokens.
- [*unfreezeAmount()*] works by advancing monthly vesting. The function takes in minimum Amount (say, x) to be released and releases tokens until – tokens released >= x.